

# SYMBIAN OS

Горнаков С. Г.

## Программирование мобильных телефонов на C++ и Java 2 ME

Компакт-диск к книге



# SYMBIAN

ДЛЯ ПРОГРАММИСТОВ

Горнаков С. Г.

SYMBIAN OS

**ПРОГРАММИРОВАНИЕ  
МОБИЛЬНЫХ ТЕЛЕФОНОВ  
НА C++ И JAVA 2 ME**

---

Москва, 2005

**УДК 004.438**  
**ББК 32.973.26-018.2**  
Г26

Горнаков С. Г.  
Г26 Symbian OS. Программирование мобильных телефонов на C++ и Java 2 ME. — М: ДМК Пресс, 2005. - 448 с: ил.

**ISBN 5-94074-030-8**

Создание мобильных приложений для операционной системы Symbian - сложная и трудоемкая задача. Эта книга познакомит вас с основами программирования для Symbian OS на языке программирования C++, а одна из глав посвящена программированию Java 2 ME приложений. Темы, рассматриваемые в книге весьма разносторонние - это интегрированные среды программирования CodeWarrior for Symbian, C++ BuilderX Mobile Studio, инструментальные средства разработчика SDK от Symbian, Sony Ericsson и Nokia для платформ UIQ, серии 60, серии 80 и серии 90. Большой объем информации освещает вопросы, связанные с программной архитектурой операционной системы, основными идиомами программирования в Symbian OS, структурой и созданием GUI приложения, локализацией, работой с меню, элементами пользовательского интерфейса, графикой, изображениями, созданием инсталляционного пакета.

Книга будет интересна широкому кругу читателей, желающим самостоятельно изучить программирование для операционной системы Symbian на языке C++.

**УДК 004.438**  
**ББК 32.973.26-018.2**

Все права защищены. Любая часть этой книги не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами без письменного разрешения владельцев авторских прав.

Материал, изложенный в данной книге, многократно проверен. Но поскольку вероятность технических ошибок все равно существует, издательство не может гарантировать абсолютную точность и правильность приводимых сведений. В связи с этим издательство не несет ответственности за возможные ошибки, связанные с использованием книги.

ISBN 5-94074-030-8  
2005

© Горнаков С. Г., 2005  
© Оформление, ДМК Пресс,

# Содержание

Предисловие .....	14
Структура книги .....	15
Что вы должны знать .....	16
Программное обеспечение .....	16
Исходные коды .....	16
Благодарности .....	17
<b>Глава 1. Знакомство с Symbian OS.....</b>	<b>18</b>
1.1. Работа в Symbian OS .....	20
1.2. Навигация .....	22
1.3. Интернет .....	23
1.4. Java-приложения .....	25
1.5. Программы наC++ .....	27
1.6. Обзор программ для Symbian OS .....	28
1.6.1. Файловые менеджеры .....	29
1.6.2. Веб-браузеры.....	29
1.6.3. Мультимедиа .....	30
1.6.4. Игры .....	32
<b>Глава 2. Среда программирования IDE Metrowerks</b>	
<b>CodeWarrior for Symbian OS .....</b>	<b>33</b>
2.1. Установка CodeWarrior for Symbian Personal v2.8.3 .....	35
2.2. Знакомство с Metrowerks CodeWarrior for Symbian Personal v2.8.3 .....	38
2.2.1. Меню File .....	40
2.2.2. Меню Edit .....	40
2.2.3. Меню View.. .....	41
2.2.4. Меню Search .....	42
2.2.5. Меню Project .....	43

2.2.6. Меню Debug .....	44
2.2.7. Меню Window .....	45
2.2.8. Меню Help .....	46
2.2.9. Панель инструментов .....	46
2.2.10. Окно Workspace .....	47
2.2.11. Текстовый редактор .....	49
2.3. Настройка Metrowerks CodeWarrior .....	51
2.3.1. Группа General .....	52
2.3.2. Группа Editor .....	56
2.3.3. Группа Debugger .....	61
2.4. Создание проекта .....	63
2.5. Импорт проекта .....	65
2.6. Компиляция проекта .....	67
2.7. Создание установочного пакета .....	69
<b>Глава 3. IDE C++ BuilderX Mobile Studio.....</b>	<b>72</b>
3.1. Установка IDE C++ BuilderX Mobile Studio .....	73
3.2. Изучаем C++ BuilderX .....	75
3.2.1. Меню File .....	76
3.2.2. Меню Edit .....	77
3.2.3. Меню Search .....	78
3.2.4. Меню View .....	78
3.2.5. Меню Project .....	79
3.2.5. Меню Run .....	79
3.2.6. Меню Team .....	80
3.2.7. Меню Wizards .....	80
3.2.8. Меню Tools .....	80
3.2.9. Меню Window .....	80
3.2.10. Меню Help .....	81
3.2.11. Панель инструментов .....	81
3.2.12. Панель Project .....	82
3.3. Подключение SDK .....	83
3.4. Создание проекта .....	84
3.5. Импорт проекта .....	86
3.6. Компиляция проекта .....	87
3.7. Создание установочного пакета .....	88
<b>Глава 4. Инструментальные средства разработчика.....</b>	<b>89</b>
4.1. Программные средства компании Sony Ericsson .....	90
4.1.1. Установка SDK .....	91

4.1.2. Эмуляторы телефонов Sony Ericsson.....	94
4.2. Программные средства компании Nokia.....	95
4.2.1. Серия 60.....	97
4.2.2. Серия 80.....	98
4.2.3. Серия 90.....	100
4.2.4. Программа SISAR.....	102
<b>Глава 5. Архитектура Symbian OS.....</b>	<b>105</b>
5.1. Аппаратная архитектура.....	106
5.2. Системные библиотеки.....	107
5.3. Программная архитектура.....	108
5.3.1. Ядро и аппаратная часть системы.....	109
5.3.2. Базовые сервисы.....	109
5.3.3. Сервисы операционной системы.....	110
5.3.4. Пользовательские сервисы.....	112
5.3.5. Инфраструктура пользовательского интерфейса.....	113
5.4. Файловая система.....	113
5.4.1. Диск Z.....	114
5.4.2. Диск C.....	114
5.4.3. Диски.....	115
5.4.4. ДискE.....	115
5.4.5. Оперативная память.....	116
<b>Глава 6. Основы программирования в Symbian OS.....</b>	<b>117</b>
6.1. Классы.....	118
6.1.1. КлассыC.....	118
6.1.2. Классы R.....	118
6.1.3. КлассыT.....	118
6.1.4. КлассыM.....	119
6.1.5. Статические классы.....	119
6.2. Функции.....	119
6.2.1. Уходящие функции.....	119
6.2.2. Неуходящие функции.....	120
6.2.3. Функции LC.....	120
6.2.4. Функции Set.....	120
6.2.5. Функции Get.....	120
6.3. Структуры.....	120
6.4. Макросы.....	121
6.5. Имена переменных.....	121
6.6. Простые типы данных.....	121
6.7. Рекомендации.....	122

<b>Глава 7. Структура приложений в Symbian OS</b> .....	124
7.1. Системные классы .....	124
7.1.1. Платформа i9000 .....	126
7.1.2. Серия 60 .....	126
7.2. Базовая составляющая приложения .....	127
7.2.1. Класс Application .....	127
7.2.2. Класс Document.....	128
7.2.3. Класс AppUI .....	128
7.2.4. Класс App View .....	128
7.3. Первая программа .....	129
7.3.1. Работа системы .....	130
7.3.2. Класс CTestApplication .....	131
7.3.3. Класс CTestDocument .....	133
7.3.4. Класс CTestAppUi .....	136
7.3.5. Класс CTestAppView .....	140
7.3.6. Oa\nTest_Main.cpp .....	144
7.3.7. Файл Test.pan .....	145
7.3.8. Файл Test.hrh.....	145
7.3.9. Файл Test_Caption.rss.....	146
7.3.10. Файл Test.rss .....	147
7.3.11. Файл bld.inf .....	149
7.3.12. Файл Test.mmp .....	149
7.3.13. Файл Test.pkg .....	151
7.4. Уникальные идентификаторы UID.....	154
7.4.1. Идентификатор UID1 .....	155
7.4.2. Идентификатор UID2 .....	155
7.4.3. Идентификатор UID3 .....	155
7.4.5. Идентификаторы платформы.....	156
7.5. Добавляем иконку в приложение .....	157
7.5.1. Добавление AIF ресурсов в C++ BuilderX.....	162
7.6. Сборка проекта компилятором .....	163
7.7. Создание установочного пакета SIS .....	166
<b>Глава 8. Интерфейс пользователя</b> .....	167
8.1. Платформа UIQ.....	167
8.1.1. Панель Application Picker .....	167
8.1.2. Панель Menu bar .....	167
8.1.3. Клиентская область экрана .....	168
8.1.4. Панель ToolBar .....	168
8.1.5. Панель Status bar .....	168

8.2. Серия 60 .....	168
8.2.1. Панель Status Pane .....	168
8.2.2. Панель Main Pane .....	169
8.2.3. Панель Control Pane .....	169
8.3. Ресурсы .....	169
8.4. Меню .....	170
8.5. Локализация .....	181
8.6. Получение данных от пользователя .....	186
8.7. Списки .....	187
8.7.1. Вертикальный список .....	188
8.7.2. Список Grid .....	192
8.7.3. Список Setting .....	193
8.7.4. Демонстрационный пример Setting List .....	199
<b>Глава 9. Программирование графики .....</b>	<b>223</b>
9.1. Рисование линий .....	224
9.2. Рисуем прямоугольник .....	230
9.3. Рисуем эллипс .....	235
9.4. Рисуем часть круга .....	238
9.5. Текст и шрифт .....	242
9.6. Работа с изображениями .....	251
<b>Глава 10. Программирование Java приложений .....</b>	<b>255</b>
10.1. Платформа Java 2 ME .....	255
10.1.1. Конфигурация CLDC .....	256
10.1.2. Профили MIDP .....	257
10.2. Мидлет .....	258
10.2.1. Структура работы мидлета .....	259
10.2.2. Экранная навигация .....	263
10.3. Высокоуровневый пользовательский интерфейс .....	265
10.3.1. Класс TextBox .....	265
10.3.2. Класс List .....	265
10.3.3. Класс Alert .....	266
10.3.4. Класс Form .....	266
10.3.5. Класс ChoiceGroup .....	267
10.3.6. Класс StringItem .....	267
10.3.7. Класс TextField .....	267
10.3.8. Класс DateField .....	267
10.3.9. Класс Spacer .....	268
10.3.10. Класс ImageItem .....	268



1.20. Bluetooth Security Manager .....	296
1.21. Bluetooth Service Discovery Agent .....	296
1.22. Bluetooth Service Discovery Database .....	297
1.23. Bluetooth Sockets .....	298
1.24. Bluetooth UI .....	300
1.25. Calendar Conversion .....	301
1.26. Certificate Manager .....	302
1.27. Character Conversion .....	306
1.28. Character Conversion Plug-In Provider .....	306
1.29. Character Representation of Real Numbers .....	306
1.30. Circular Buffers .....	307
1.31. Client/Server .....	307
1.32. Clipboard .....	307
1.33. Clock .....	307
1.34. Command Line Parsing .....	308
1.35. CommDb .....	309
1.36. Connection Management .....	310
1.37. Contacts Model .....	311
1.38. Contact Views .....	314
1.39. Converter Architecture .....	315
1.40. Cookies Support .....	316
1.41. Critical Sections .....	317
1.42. Data Application Model .....	317
1.43. Date and Time Handling .....	319
1.44. DBMS Columns, Column Sets and Keys .....	320
1.45. DBMS Database Incremental Operations .....	321
1.46. Interface to DBMS Databases .....	321
1.47. DBMS Rowsets .....	322
1.48. DBMS Sharing Databases .....	323
1.49. Descriptor Arrays .....	323
1.50. Descriptors .....	324
1.51. Device drivers .....	326
1.52. Dial .....	327
1.53. Dynamically Loading Link Libraries .....	328
1.54. Dynamic Arrays .....	328
1.55. Dynamic Buffers .....	330
1.56. ECom Plug-in Architecture .....	330
1.57. Embedding .....	331
1.58. Encrypted Streams and Stores .....	332
1.59. Environment Change Notifier .....	332

1.60. ETel Core.....	332
1.61. Extended Notifier Framework.....	332
1.62. Fax Client.....	333
1.63. Fax Client MTM.....	333
1.64. Fax Header Line.....	334
1.65. Fax Store.....	334
1.66. File Logging.....	335
1.67. File Server Client Side.....	335
1.68. Writing a file system.....	337
1.69. Filestores.....	339
1.70. Fixed Size Arrays.....	339
1.71. Fonts.....	340
1.72. Fonts and Bitmaps.....	341
1.73. Front End Processors.....	341
1.74. FTP Engine.....	342
1.75. Graphics.....	343
1.76. Graphics Foundations.....	344
1.77. Grid Foundations.....	345
1.78. Grid.....	345
1.79. Hardware Abstraction Layer (HAL).....	346
1.80. Handles.....	346
1.81. Hardware Accelerator.....	347
1.82. Help Model.....	349
1.83. HTTP Client.....	350
1.84. HTTP Message.....	353
1.85. HTTP Transport Layer.....	353
1.86. Image Converter.....	354
1.87. Incremental Matcher.....	355
1.88. Interface to Resource Files.....	355
1.89. Internet Mail.....	356
1.90. Internet Protocol Utility.....	360
1.91. Interrupt architecture.....	364
1.92. IPSec.....	364
1.93. IrDA Sockets.....	365
1.94. InfraRed Transfer Picture Protocol.....	365
1.95. Lexical Analysis.....	366
1.96. Literal Descriptors.....	366
1.97. Locale Settings.....	367
1.98. Log Engine.....	369
1.99. Maths Services.....	370

1.100. Media Server Common Classes.....	370
1.101. Memory Streams .....	371
1.102. Message Scheduled Sending .....	371
1.103. Message Window .....	373
1.104. MMS MTM Client .....	373
1.105. MMS Utilities.....	374
1.106. MultiMediaCard.....	376
1.107. NetDial .....	379
1.108. Notification Services .....	379
1.109. Onboard Camera .....	379
1.110. Open Font System.....	379
1.111. Package Buffers .....	380
1.112. PC Connect Device-side BAL.....	380
1.113. PhoneBook Synchroniser .....	381
1.114. Power management framework.....	381
1.115. Print Framework .....	381
1.116. Print Preview.....	382
1.117. Printing.....	382
1.118. Raw Memory.....	383
1.119. Recognizers.....	383
1.120. Reference counting objects .....	384
1.121. Security .....	384
1.122. Semaphores.....	385
1.123. Serial Protocol Module.....	385
1.124. SIM Application Toolkit .....	385
1.125. SMS GSM Utilities .....	386
1.126. SMS Utilities.....	388
1.127. Sockets Client.....	389
1.128. Sound Device .....	390
1.129. Stores.....	391
1.130. Store Streams .....	392
1.131. System Agent.....	392
1.132. System Sounds.....	393
1.133. TCP/IP .....	393
1.134. Test Console.....	395
1.135. Text and Text Attributes.....	395
1.136. Text Views .....	397
1.137. Timers and Timing Services .....	399
1.138. Transfer Buffer .....	399
1.139. To-do List.....	399

1.140. UID Manipulation.....	400
1.141. UI Control Framework .....	400
1.142. UI Graphics Utilities.....	403
1.143. Uikon Core .....	404
1.144. Uikon Resources.....	406
1.145. USB Client.....	408
1.146. WAP Messaging.....	410
1.147. WAP SMS Protocol Module .....	411
1.148. WAP Stack.....	411
1.149. Window Server Client Side .....	412

## **Приложение 2. Техническая документация**

<b>телефонов Symbian OS.....</b>	<b>413</b>
ArimaU300 .....	413
BenQP30 .....	414
FomaF880iES.....	414
FomaF900i .....	415
FomaF900it.....	415
FomaF901ic.....	416
FomaF2051 .....	416
LenovoP930.....	417
Motorola A920.....	417
Motorola A925.....	418
Motorola A1000.....	418
Motorola A1010.....	419
Nokia N-Gage .....	419
Nokia N-Gage QD .....	420
Nokia 3230.....	420
Nokia 3650/3600.....	421
Nokia 3660/3620.....	421
Nokia 6260.....	422
Nokia 6600.....	422
Nokia 6620.....	423
Nokia 6630.....	423
Nokia 6670.....	424
Nokia 6680.....	424
Nokia 6681 .....	425
Nokia 6682.....	425
Nokia 7610.....	426
Nokia 7650.....	426

Nokia 7710.....	427
Nokia 9210.....	427
Nokia 9290 .....	428
Nokia 9300 .....	428
Nokia 9500 .....	429
Panasonic X700.....	429
Panasonic X800.....	430
SendoX .....	430
SendoX2 .....	431
Siemens SX1 .....	431
Sony Ericsson P800.....	432
Sony Ericsson P900.....	432
Sony Ericsson P900i.....	433
Nokia N70 .....	-...433
Nokia N90.....	434
Nokia N91.....	434
<b>Приложение 3. Интернет ресурсы .....</b>	<b>435</b>
Компания Symbian Ltd.....	435
Инструментальные средства разработчика .....	435
Интегрированные средства разработки приложений .....	436
Компании .....	436
Тематические сайты.....	437
Русскоязычные сайты.....	437
<b>Приложение 4. Обзор компакт-диска .....</b>	<b>438</b>
Список используемых источников .....	439
Предметный указатель.....	440

# Предисловие

Человечество не стоит на месте и развивается, внедряя в жизнь все новые и новые технологии. Буквально три-четыре года назад мы радовались простенькому телефону с монохромным дисплеем, весом и размером далеким от своего названия — мобильный телефон. А сейчас удивить телефоном с цветным дисплеем и большим разрешением экрана сложно, да и это уже не роскошь, а обычное средство для связи с другими людьми. Технологии развиваются быстрыми темпами и кроме своего прямого назначения, в мобильные телефоны встраиваются камеры, радио-, видео- и МРЗ-проигрыватели, появилась возможность просмотра офисных документов. Словосочетание «мобильный офис», вошедшее в наш обиход, очень ярко и красочно рисует ситуацию, сложившуюся на мобильном рынке. Человек хочет одно небольшое по весу и размеру устройство, с хорошим разрешением экрана и с возможностями настольной компьютерной системы среднего уровня. Возможно ли это? Жизнь показывает, что да - ведущие компании мира, производящие телефоны, придумывают различные дизайны, пытаются какими-то невероятными способами совместить полноценную клавиатуру, большой экран и при этом оставить телефон действительно мобильным устройством. Кому-то удастся сделать это лучше, кому-то хуже, но движение идет в этом направлении и именно такой вариант развития, скорее всего, и определит будущее мобильных технологий.

Все телефоны делятся на две категории - это устройства, работающие на основе прошивки и устройства, работающие под управлением операционной системы. Телефоны с прошивкой несколько ограничены в своих дополнительных возможностях, но поддержка технологии Java 2 ME значительно улучшает обстановку, делая возможной загрузку в телефон программ сторонних производителей, тем самым, дополняя устройство функциями, которые изначально в этом телефоне не были предусмотрены. Телефоны с операционной системой намного мощнее и имеют много встроенных программ с возможностью инсталляции на устройство дополнительных программ написанных на языках программирования C++, Java 2 ME, OPL и даже Visual Basic.

На мировом рынке в сегменте операционных систем для мобильных устройств сейчас два потенциальных конкурента - Windows Mobile и Symbian OS. Операционная система Windows Mobile менее распространенная и более дорогая. Телефоны под управлением этой операционной системы стоят достаточно дорого и рассчитаны скорее на корпоративных клиентов, чем на рынок массового потребления. Но ситуация может резко измениться - рынок КПК (карманные персональные компьютеры) начинает медленно, но верно, сдавать свои позиции. Не даром даже такой монстр как компания Sony собрала свои вещички и ушла с рынка КПК. Компании, выпускающие КПК, со временем могут оказаться у разбитого корыта и массовый переход к телефонам с функциями КПК не за горами,

что уже и происходит. Тем более что за границей перестраивать производство умеют хорошо. Это только у нас до сих пор выпускается автомобиль квадратной формы - ВАЗ-2109, запущенный в производство очень давно.

Операционная система Symbian изначально создавалась для работы на телефонах, в отличие от Windows Mobile, которая оптимизирована под мобильные устройства. Компания с одноименным названием - Symbian Ltd. — является создателем этой операционной системы. Пакет акций Symbian Ltd. уже давно поделен между компаниями производящими основной поток мобильных телефонов. По сведениям на 1 января 2005 года процентное соотношение владением акций выглядит следующим образом:

- Nokia-47,9%;
- Ericsson- 15,6%;
- Sony Ericsson - 13,1%;
- Panasonic - 10,5%;
- Siemens - 8,4%;
- Samsung - 4,5%.

Мобильные устройства под управлением операционной системы Symbian делятся на два вида: коммуникаторы и смартфоны.

В самой Symbian OS телефоны делятся еще по платформам, представляющим определенный интерфейс пользователя, - это платформы UIQ, серия 60, серия 80 и серия 90. Все они имеют общий принцип работы, поэтому если вы знакомы, скажем, с серией 60, то вы знаете где-то 80% возможностей платформы UIQ. Рассмотреть все платформы в одной книге невозможно, поэтому за основу взята серия 60, а это около 70-80% мобильных устройств в мире, но мы обязательно будем уделять внимание и другим платформам.

## Структура книги

**Глава 1.** В этой главе содержатся основные сведения о работе с Symbian OS. Рассматриваются программы сторонних производителей, настройка системы, установка и удаление программ, пользование Интернетом и многое другое.

**Глава 2** знакомит читателя со средой программирования CodeWarrior 2.8 for Symbian OS компании Metrowerks, предназначенной для программирования мобильных приложений на языке программирования C++.

**Глава 3.** Компания Borland имеет свою среду программирования C++ BuilderX Mobile Studio, и в этой главе вы познакомитесь с основными принципами работы среды C++ BuilderX.

**Глава 4.** Для создания программ под Symbian OS необходимы инструментальные средства разработчика SDK. Глава содержит описание SDK серий 60,80, 90 и UIQ, от компаний Symbian, Sony Ericsson и Nokia, которые вы так же можете найти на компакт-диске.

**Глава 5.** Операционная система Symbian построена на основе модульной архитектуры, и эта глава содержит полную информацию о программной части системы и возможностях аппаратной архитектуры мобильных устройств.

**Глава 6** содержит основные понятия идиом программирования для мобильных приложений на C++ в Symbian OS.

**Глава 7.** В течение всей главы будет создаваться каркас GUI-приложения, с подробным объяснением всех составляющих, на основе GUI-приложения строятся все программы в Symbian OS.

**Глава 8** познакомит вас с интерфейсом пользователя платформ UIQ и серии 60 и с реализацией основных методов при работе с пользовательским интерфейсом. Так же рассматривается механизм локализации программ.

**Глава 9.** Большой вводный курс программирования графики для Symbian OS. Показана система работы с изображениями и методы рисования различных геометрических фигур на экране телефона.

**Глава 10.** Эта глава целиком посвящена технологии Java 2 ME, поддержка которой осуществляется Symbian OS на программном уровне.

**Приложение 1.** Справочник по наименованиям классов, интерфейсов и типов в Symbian OS версии 7.0s.

**Приложение 2.** Техническая информация об имеющихся моделях телефонов под управлением разных версий операционной системы Symbian.

**Приложение 3.** Обзор Интернет ресурсов посвященных программированию под операционную систему Symbian.

**Приложение 4.** Обзор компакт-диска.

**Приложение 5.** Список используемых источников.

## Что вы должны знать

Предполагается, что вы уже знакомы с минимальными основами языка программирования C++, умеете работать с визуальными средствами программирования и хотя бы в общих чертах понимаете принцип создания, компиляции и тестирования программ.

## Программное обеспечение

В течение всей книги для создания программ использовались среды программирования CodeWarrior for Symbian OS от компании Metrowerks и C++ BuilderX Mobile Studio компании Borland. Но программы так же рассчитаны на работу с Visual Studio и командной строкой. Необходимые для создания приложений SDK находятся на компакт-диске к книге.

## Исходные коды

Демонстрационные примеры, рассмотренные в книге, разбиты на проекты и находятся на компакт-диске в папке \Code. Каждый проект имеет понятное название, и догадаться о сути разбираемого примера будет легко. Кроме того, при рассмотрении примера, в книге всегда дается ссылка на папку проекта, находящуюся на компакт-диске. Исходные коды поставляются в нейтральной конфигурации, что дает возможность их импорта в CodeWarrior for Symbian OS любой



версии, C++ BuilderX версий 1.0 и 1.5, Visual C++.6 и Visual Studio.NET, а также работу с командной строкой.

## **Благодарности**

Особую благодарность хочу выразить издателю этой книги, Мовчану Дмитрию Алексеевичу, за всестороннюю помощь и поддержку в период работы над книгой и потрясающую коллекцию SDK, представленную в комплекте с книгой.

Очень признателен моей жене за рисунки к книге.

Отдельная благодарность Колесниченко Андрею Александровичу за дружеские отношения и за то, что он с нетерпением ожидал выхода этой книги.

# Глава 1. Знакомство с Symbian OS

Операционная система Symbian создана компанией Symbian Ltd. За годы существования Symbian OS было выпущено восемь версий этой операционной системы, и каждая новая версия вбирала в себя все самое лучшее из предыдущих выпусков. Первый выпуск Symbian OS (тогда она еще называлась EPOC 32) появилась в 1997 году. Эта версия операционной системы была основана на технических разработках компании Psion, которая на тот момент лидировала в производстве персональных органайзеров. Большая серия устройств под названием Psion Series в девяностых годах имела огромную популярность у частных и корпоративных клиентов. На базе подразделения Psion Software компаниями Psion, Nokia, Ericsson и Motorola была создана первая операционная система EPOC 32, в последствии переименованная в Symbian. С первого выпуска Symbian OS (для мобильных телефонов) является открытой системой, и ее использование производителями телефонов на своих устройствах осуществляется на основе лицензирования, что и послужило столь широкому ее распространению.

Первые пять версий Symbian OS были выпущены в течение последующих трех лет от даты выхода EPOC 32. Все они были построены на 16-разрядной архитектуре, что на тот момент было весьма неплохо, но с течением времени мобильные устройства становились мощнее, и потребность в 32-разрядной платформе назрела естественным образом. Мобильные устройства стали действительно мобильными и по размерам, и по своему техническому потенциалу. Такое развитие не осталось незамеченным компанией Symbian Ltd., и в 2000 году вышла шестая версия операционной системы Symbian с полноценной 32-разрядной архитектурой. Поддержка приложений, написанных под первые пять версий Symbian, в шестой и седьмой версиях отсутствует и это, прежде всего, объясняется сменой множества программных интерфейсов. Библиотека API в шестой и седьмой версиях значительно изменилась как в техническом аспекте, так и просто в названиях интерфейсов, классов и функций. Год спустя, в 2001 году, вышел новый релиз шестой версии под названием Symbian OS 6.1, где происходит разделение операционной системы на две части, а именно: на ядро и графическую подсистему. Это был очень важный шаг компании Symbian Ltd. Благодаря такому разделению системы на подсистемы, у производителей мобильных устройств появилась возможность сопровождать свои телефоны оригинальным графическим интерфейсом под стать своему фирменному стилю, а также адаптировать пользовательский интерфейс под конкретные устройства с различным разрешением экрана. Одна и та же версия Symbian OS может выглядеть по-разному на различных устройствах. Посмотрите на рис. 1.1, где представлены два пользовательских интерфейса смартфонов компании Nokia: N-Gage и N-Gage QD. Оба эти

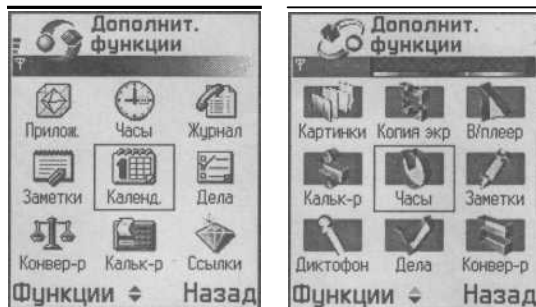


Рис. 1.1. Смартфоны N-Gage и N-Gage QD

устройства работают под управлением Symbian OS 6.1, но взгляните, насколько отличаются их графические интерфейсы.

Интерфейсы, изображенные на рис. 1.1, - это и есть свои фирменные стили, которые преобразовываются под конкретную марку устройства, что дает разработчикам массу новых возможностей. В число лицензиатов входят такие известные компании, как:

- Psion;
- Nokia;
- Motorola;
- Sony Ericsson;
- Siemens;
- Samsung;
- SUN;
- Panasonic;
- Sanyo;
- BenQ;
- Fujitsu;
- Sendo;
- Mitsubishi Electric;
- Borland;
- ARM;
- Intel;
- AppForge;
- G Metrowerks;
- Vodafone;
- Orange.

В 2005 году появилась последняя на сегодняшний день восьмая версия Symbian OS. Телефонов на платформе этой версии выпущено пока три модели - Nokia 6630, Nokia 6680 и Nokia 6681. Правда анонсировано еще несколько устройств, но их еще нет в продаже, поэтому в книге версия Symbian OS 8.0 не рассматривается. В итоге на сегодняшний день существуют версии Symbian OS 6.1 (7.0,7.0s, 8.0,8.0a).

По заявлениям Symbian Ltd. к концу 2005 года ожидается появление мобильных устройств на базе Symbian OS 9.0. Приложения, написанные под Symbian OS 6.1, совместимы для программ под Symbian OS 7.0 и выше, а вот обратной совместимости нет. К сожалению, такая особенность представляет дополнительные трудности программистам, тем более что телефонов, работающих на Symbian OS 6.1, сейчас на потребительском рынке порядка 50%.

Если вы знакомы с книгой «Программирование мобильных телефонов на Java 2 Micro Edition», то найдете некоторое сходство в написании приложений на Java под профили MIDP 1.0 и MIDP 2.0. Практически то же самое наблюдается в программах под Symbian OS 6.1 и Symbian OS 7.0. Это несколько усложняет нашу задачу, но, несмотря на это, Symbian OS - хорошо продуманная, многозадачная 32-разрядная операционная система, написанная на языке программирования C++ в лучших традициях объектно-ориентированного программирования. Пожалуй, отличительными особенностями этой операционной системы являются ее стабильность в работе и компактность. Так как на телефонах под управлением Symbian OS переустановить операционную систему невозможно, то стабильность в работе становится одним из определяющих факторов, поднявших эту операционную систему до таких высот.

## 1.1. Работа в Symbian OS

Прежде чем мы перейдем к созданию приложений, необходимо ознакомиться с основами работы в Symbian OS, то есть побывать в роли простого пользователя, чтобы осмыслить принцип работы программ, навигации в приложениях и рассмотреть нюансы работы с этой операционной системой.

Количество телефонов, работающих под управлением Symbian OS, большое (в *Приложении 2* вы найдете полный обзор телефонов на базе Symbian OS). Каждая модель телефона, как правило, имеет свой, присущий только этой модели, графический интерфейс. Каждый производитель придерживается своей системы расположения графических элементов, но общий принцип работы с приложениями для Symbian OS все же одинаков.

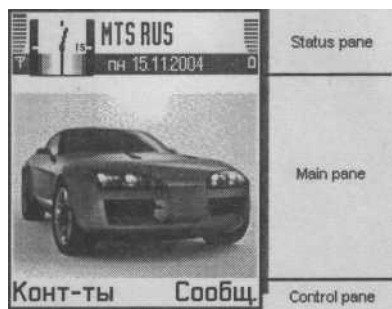


Рис. 1.2. Экран телефона Nokia N-Gage QD

Отображение информации сводится к экранному представлению данных, а переход с экрана на экран производится с помощью подэкранных клавиш телефона, джойстика или стилуса (телефоны с сенсорным экраном). На одном экране одновременно может отображаться только одно окно, исключения составляют лишь небольшие по размеру всплывающие диалоговые или информационные окна, например, уведомляющие пользователя об ошибке, о получении SMS, входящего звонка, лицензионном соглашении при установке приложения, списков и так да-

лее. Такие окна, как правило, появляются лишь на некоторое время, после чего исчезают или требуют подтверждения, отмены каких-то действий. Экран телефона делится на области (см. рис. 1.2, где изображен экран телефона Nokia серии 60).

На панели состояния (Status panel) находится логотип оператора сотовой связи, индикаторы заряда батареи и сети, число, время (см. рис.1.2). Панель состояния при переходе с экрана на экран, то есть при перемещении по меню, остается жестко закрепленной. Меняются только графические элементы, отображаемые на этой панели. Есть исключение, когда панель состояния может не использоваться в связи с работой программы в полноэкранном режиме, например, в играх. Обычно это красочные окна со своим продуманным интерфейсом и средствами навигации, напоминающими компьютерные игры. Посмотрите на рис. 1.3, где показаны меню игр Red Faction и Splinter Sell.

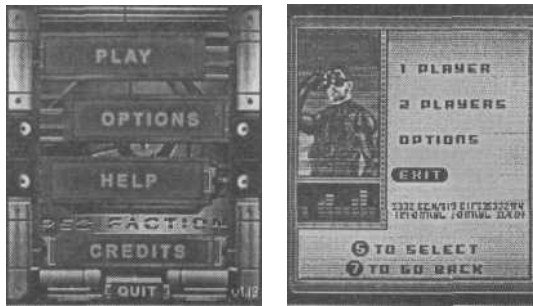


Рис. 1.3. Меню игр Red Faction и Splinter Sell

Панель контроля (Control panel) содержит назначенные команды для двух по-дэканных клавиш телефона. Набор команд может быть абсолютно разным и зависеть от решаемой задачи. В основном клавиша, находящая под левым нижним углом экрана телефона, содержит команды: **Продолжить**, **Принять**, **Функции**, **Выбор** и так далее, а клавиша, расположенная под правым нижним углом экрана, содержит команды отменяющего характера. Возможно и назначение обеим клавишам выполнения функций меню. Посмотрите на рис. 1.4, где показаны варианты взаимодействия с панелью контроля.

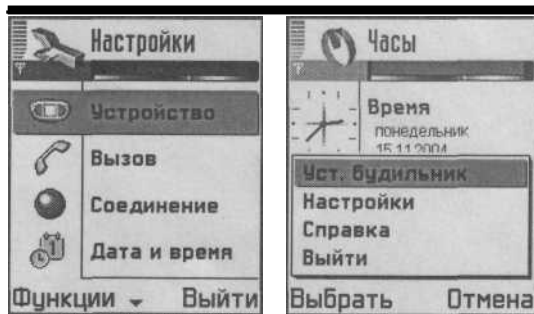


Рис. 1.4. Панель контроля

Меню, назначенное для одной из клавиш панели контроля, может содержать любое количество команд, но программист должен следить, чтобы навигация в программе была интуитивно понятной и не вызывала затруднений при переходе с экрана на экран. В телефонах, использующих пользовательский интерфейс UIQ, несколько иное расположение панелей, о чем вы узнаете из главы 7.

## 1.2. Навигация

*Рабочий стол* в зависимости от производителя может использоваться по-разному. Одни производители сразу отображают меню на рабочем столе в виде набора иконок, как показано на рис. 1.1 (но и при переходе на рабочий стол сохраняется панель контроля и панель состояния с графическими элементами, названием окна и так далее). Перемещение по командам меню происходит с помощью курсора, который перемещается по экрану джойстиком или управляющими клавишами: **Up** (Вверх), **Down** (Вниз), **Left** (Влево), **Right** (Вправо). Если одна из иконок или команда меню находится в фокусе курсора, то при нажатии клавиши выбора произойдет переход в программе в соответствии с выбранной командой. Посмотрите на рис. 1.5, где показаны два окна **Средства** и **Дополнительные функции** телефона N-Gage.



Рис. 1.5. Окна **Средства** и **Дополнительные функции** телефона N-Gage

При переходе фактически происходит простая смена экранов. В новом окне могут быть доступны еще ряд вложенных команд, а на панели контроля в этом случае располагаются команды возврата в родительское окно. К чему я все это объяснял: главное чтобы вы сейчас разобрались в схеме перехода с экрана на экран, иначе говоря, в *навигации*, потому что подобные принципы перемещения используются во всех программах под Symbian OS.

Но есть и еще один вид представления команд меню - это так называемые *вкладки* (см. рис. 1.6). Перемещаться по ним можно с помощью джойстика или клавиш **Left** и **Right**. Такой вид навигации достаточно часто используется в программах и принадлежит к одному из видов диалога.

Набор команд в отдельно взятом устройстве зависит от функциональных возможностей самого телефона. В телефоне всегда имеются встроенные програм-

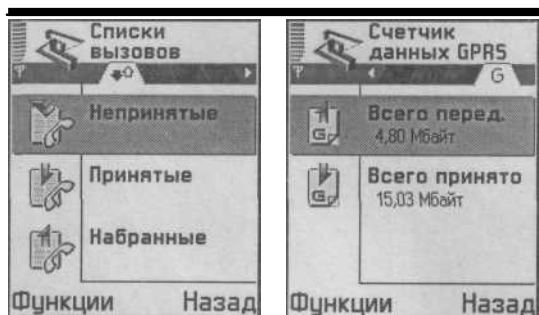


Рис. 1.6. Вкладка в Symbian OS

мы, поставляемые с Symbian OS. Это различный дополнительный софт, например, калькулятор, текстовый редактор, конвертер валют, диктофон, календарь, телефонная книга, композитор, просмотрщик изображений и многое другое. Надеюсь, вам стал понятен общий смысл экранной навигации в программах для Symbian OS.

### 1.3. Интернет

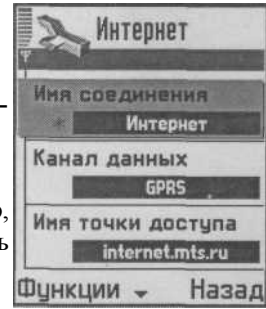
Операционная система Symbian дает возможность пользователю работать с сетью Интернет и просматривать WML- (язык разметки страниц для беспроводных устройств), XHTML- и HTML-страницы. Встроенные возможности просмотра стандартных HTML-страниц на некоторых устройствах по непонятным причинам не поддерживаются, но эту проблему весьма благополучно решают программы сторонних производителей, например мобильные Интернет-браузеры Opera, NetFront или Browser (о них вы узнаете дальше в этой главе из раздела *Обзор программ*).

Для доступа в Интернет можно использовать WAP (Wireless Application Protocol - протокол беспроводного соединения) или GPRS (General Packet Radio Services - служба пакетной передачи данных). С финансовой точки зрения выгодней работать через GPRS, оттого, что плата взимается за объем скаченных данных, а это приблизительно от 5 до 7 рублей за мегабайт (в зависимости от оператора сотовой связи). Некоторые операторы берут еще и посуточную абонентскую плату, но есть операторы, которые поступают более гуманно.

Для выхода в Интернет необходимо создать *точку доступа* с заданными параметрами. Обычно используются следующие параметры:

- имя соединения - это описательное имя для создаваемой точки доступа;
- канал данных - если вы планируете использовать GPRS, укажите именно это значение;
- имя точки доступа - этот параметр нужно узнать у своего поставщика услуги;
- имя пользователя — так же предоставляется оператором связи;

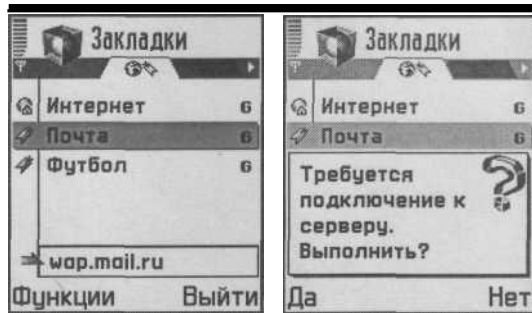
- пароль - предоставляет поставщик услуги;
- домашняя страница - это домашняя страница вашего оператора сотовой связи;
- IP-адрес шлюза - адрес шлюза WAP используемого при подключении.



На рис. 1.7 показано определение точки доступа.

После определения точки доступа можно беспрепятственно, конечно, при наличии денежных средств на счету, посещать интересные Интернет-страницы.

Интернет-сайты, созданные при помощи языка WML, можно просматривать через GPRS-WAP. Для этого зайдите в *Рис. 1.7. Определение точки доступа* меню телефона **Интернет** и создайте новую закладку, после чего нажмите **Выбрать**. Появится небольшое диалоговое окно с вопросом о подключении к серверу, как показано на рис. 1.8. Подтвердите свои намерения и ждите связи с сетью Интернет по каналу GPRS.



*Рис. 1.8. Соединение с Интернет*

Интернет-страницы, написанные с помощью языка WML, предназначены для отображения простой текстовой информации. Иногда это очень удобно, например, сидя в автобусе, можно быстро просмотреть свою электронную почту. Так что слухи о смерти WAP несколько преувеличены. С другой стороны, сайты созданные с помощью XHTML и HTML, выглядят куда привлекательней. Не стоит забывать и о том, что при доступе к Интернет-ресурсам в формате HTML можно не только просматривать страницы, но еще и скачивать файлы, например, вложение к письму электронной почты. Учитывая тот факт, что за один мегабайт принятой информации по каналу GPRS необходимо заплатить несколько рублей, это совсем неплохо, и вы действительно имеете в руках мобильный офис. Конечно, присутствуют и некоторые неудобства, например, маленький размер экрана, затрудняющий работу, однако технологии постоянно развиваются. Ведь телевизоры тоже когда-то были громоздкими и черно-оелыми.



## 1.4. Java-приложения

Операционная система Symbian позволяет запускать программы, написанные на языке C++ и Java 2 ME. Телефоны на платформе Symbian OS располагают как минимум 3,4 мегабайтами пользовательской памяти (для хранения программ и пользовательских данных, не путать с оперативной памятью). Дополнительно возможно подключение карт памяти различных форматов (в зависимости от производителя телефонов) размером 32, 64, 128 и 512 Мб. На подобный объем дополнительной памяти можно установить несколько десятков различных программ и главное, размер программы написанной на Java 2 ME или C++ абсолютно не ограничен в отличие от телефонов с поддержкой только Java 2 ME. В смартфонах под управлением Symbian OS при наличии *карты памяти* можно устанавливать программы, не заботясь об их размере. Поэтому при покупке карты памяти к телефону не скупитесь - приобретите карту наибольшего размера. Единственное, что необходимо учесть при выборе, это ограничения размера карты памяти, задекларированные производителем. Эту информацию можно найти в справочном руководстве телефона.

Программы, написанные на Java 2 ME или C++ можно приобрести, например, в Интернете, благо их стоимость невелика. Загружать программы можно прямо на телефон по каналу GPRS или на компьютер, а после передать их на телефон. Связь телефона с компьютером осуществляется через инфракрасный порт, Bluetooth или с помощью обычного кабеля, подключаемого к COM- или USB-порту компьютера. Для связи компьютера и телефона необходимо пользоваться программным обеспечением, поставляемым производителем телефона. Если у вас есть карта памяти и соединение с компьютером происходит через USB-кабель, то можно обойтись и без программного обеспечения. В этом случае при подключении телефона к компьютеру карта памяти телефона определится операционной системой компьютера как дополнительный съемный накопитель. Такой способ подключения напоминает работу с устройством USB Flash (флэш-карта): можно удалять или перемещать любые файлы и программы на карте памяти вашего телефона. Но все же лучше для связи телефона и компьютера использовать программное обеспечение, которое находится на сайте производителя телефонов. Для смартфонов на базе Symbian OS можно использовать неплохую программу под названием Oxygen Phone Manager 2.2.1 доступную на сайте <http://www.opm-2.com/symbian>. Программа имеет понятный интерфейс и работает с телефонами на платформе Symbian OS вне зависимости от производителя.

Программы, написанные на Java 2 ME, распространяются в заархивированном виде в двух файлах с расширениями \*.jad и \*.jar. Файл \*.jad - это дескриптор приложения, содержащий описательные характеристики для jar-файла. В свою очередь jar-файл - это и есть откомпилированная программа на Java 2 ME. Загрузив Java программу в смартфон, у вас есть два варианта *установки*. Один из способов - это использование файлового менеджера, вы узнаете о нем в этой главе из раздела 1.6 (*Обзор программ для Symbian OS*). Другим способом является

применение стандартных средств, предусмотренных в Symbian OS. В платформу Symbian встроена специальная программа под названием **Приложения**. Для того чтобы воспользоваться этой программой разместите Java приложение в каталог C:\(модель телефона)\1пз1а1Б, после этого программа **Приложения** увидит установленные Java-файлы. Зайдите в **Меню => Приложения** и перейдите на вкладку **Загружено**. На вкладке **Загружено** в основной области окна будут перечислены все Java-программы, доступные для установки на телефон. Воспользуйтесь командами **Функции => Установить** в меню вашего телефона. Появится диалоговое окно с вопросом о продолжении или отмене процесса установки. При положительном ответе на экране телефона появится еще одно диалоговое окно для выбора каталога установки. Посмотрите на рис. 1.9, где на примере игры Call of Duty 1.0 показан процесс установки Java игры.

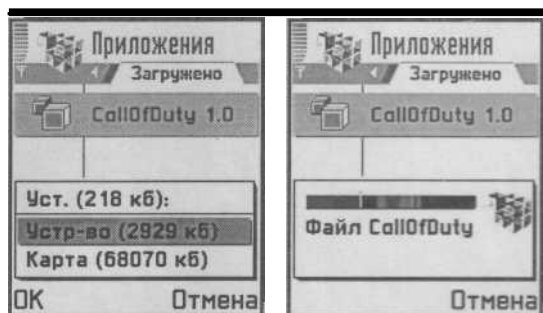


Рис. 1.9. Установка игры Call of Duty 1.0

При выборе каталога установки программы у вас есть два варианта это: память телефона и карта памяти (если она установлена на телефоне). На рис. 1.9 видно, как в диалоговом окне при выборе директории отображается сначала размер устанавливаемой программы в килобайтах, а затем свободное пространство на карте памяти и памяти устройства. Все приложения на Java и C++ лучше устанавливать на карту памяти для экономии системных ресурсов телефона. Как уже было отмечено, пользовательская память смартфонов равна как минимум 3,4 Мб. Из них от 800 Кб до 1,5 Мб занято самой системой, поэтому к оставшимся мегабайтам лучше относиться бережно, применяя карту памяти, которая может уместить в себе не один десяток программ.

После выбора каталога установки продолжите процесс инсталляции приложения, нажав подэкранную клавишу **Выбрать**. После окончания процесса установки зайдите в программу **Приложения**. Здесь вы увидите весь список установленных Java программ. Запуск программы осуществляется с помощью команд **Функции => Открыть**. Удаление установленных программ нужно производить через программу **Приложения**. Выберите программу для удаления и воспользуйтесь командами **Функции => Удалить**.

## 1.5. Программы на C++

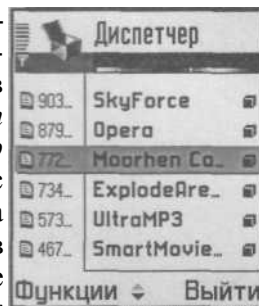
Самым главным и очевидным достоинством Symbian OS является возможность загрузки программ, написанных на языке программирования C++. Такие программы пишутся с помощью специализированных средств программирования. Операционная система Symbian OS написана на C++, и этот язык программирования считается для нее основным.

Огромная системная библиотека (API) компании Symbian Ltd., написанная на языке программирования C++, открывает потрясающие возможности для разработки мощных приложений.

Программы, написанные на C++, распространяются в файлах с расширением \*.sis. Пакет с расширением \*.sis создается в средах программирования (IDE), рассматриваемых в *главах 2 и 3*, с помощью специальной программы под названием SISAR или командной строки. Программа SISAR распространяется с инструментальными средствами разработчика (SDK) от компании Nokia, о которых вы узнаете из *главы 4*.

Процесс инсталляции C++-программ практически идентичен процессу установки Java-приложений за исключением нескольких моментов. При установке C++-программ также можно воспользоваться двумя

способами: с помощью файлового менеджера или встроенными средствами Symbian OS. Установку C++-программ с помощью файлового менеджера мы рассмотрим в этой главе в разделе *1.6 (Обзор программ для Symbian OS)*. Встроенная программа под названием **Диспетчер** производит установку и удаление программ с расширением \*.sis. При инсталляции программы на C++ установочный пакет должен находиться либо в системной памяти на диске C, либо в корневом каталоге диска E (карта памяти), иначе **Диспетчер** не найдет устано-



вочный пакет. На рис. 1.10 изображена программа **Диспетчер** со списком установленных программ.

В основной области окна находится выполненный список установленных программ. С левой стороны от названия любой из программ можно увидеть размер программ в килобайтах, а с правой стороны значок, показывающий установлена программа или нет. Значок определения установки программы может быть разным и зависит от производителя. Устанавливаются sis-программы достаточно просто. На рис. 1.11 изображен пошаговый процесс установки программы Real One Player для воспроизведения видеофайлов на телефоне.

Выделив с помощью курсора название устанавливаемой программы, в меню программы **Диспетчер** выберите команды **Функции => Установить**. Появится диалоговое окно с подтверждением действий по установке. После выбора команды ОК можно будет выбрать каталог для устанавливаемой программы. Программы, написанные на C++, могут быть достаточно внушительных размеров и дости-



Рис. 1.11. Установка программы Real One Player

гать 10-15 Мб. И, как правило, это шедевры игростроения! После завершения процесса установки в меню телефона появится красочно оформленная иконка установленной программы (см. рис. 1.11) Real One Player.

Удаление установленных программ необходимо производить так же через программу **Диспетчер**. Выберите с помощью курсора программу, которую вы хотите удалить, и воспользуйтесь командами **Функции => Удалить**. Никогда не удаляйте установленные Java 2 ME- и C++-программы файловыми менеджерами! Эти действия не корректны и создают множество проблем, в *главе 6* вы найдете одну небольшую историю как раз на эту тему.

Используя программы на C++ от сторонних производителей, на своем телефоне можно организовать полноценный мобильный офис и быть в курсе всех последних событий. В следующем разделе представлен небольшой обзор программ для Symbian OS.

## 1.6. Обзор программ для Symbian OS

На рынке программного обеспечения для Symbian OS сейчас имеются сотни всевозможных программ: системные, офисные, мультимедиа-, Интернет-программы, игры. Множество компаний и просто программистов-одиночек со всего земного шара активно участвуют в создании программного обеспечения и тем

самым развивают Symbian OS. Программы, предлагаемые пользователю разработчиками, могут распространяться как платно, так и бесплатно. Широко распространена система так называемых программ trial version (ограниченная по времени версия), которые работают лишь ограниченное время, после чего предлагают себя купить. Цена на подобные программы действительно разумная, поэтому объем продаж достаточно высок. Именно пробные версии программ для Symbian OS являются распространенным способом продвижения программного обеспечения на рынке, что обязательно нужно учитывать при создании своих программ.

Далее мы рассмотрим некоторые ключевые программы, без применения которых ни один пользователь не сможет обойтись, а заодно обозначим возможные приоритеты на будущее для вас как программиста.

### **1.6.1. Файловые менеджеры**

Наверно первое, что необходимо установить на свой смартфон - это файловый менеджер, без которого в принципе и «шага» нельзя сделать. Честно говоря, не понятно, почему до сих пор в стандартной поставке телефонов на базе той же серии 60, нет места файловому менеджеру. Здесь, правда, не берутся в расчет коммуникаторы - за деньги, которые они стоят, в комплекте не только файловый менеджер должен поставляться, но и палатка, надувная лодка и набор рыбацких снастей (кстати, было бы очень актуально в Финляндии).

Но вернемся к менеджерам. Количество таких программ исчисляется десятками. Наиболее известные это: FileMan, SeleQ и eFileManager. На рис. 1.12 эти программы изображены в рабочем состоянии, отображающие файловую систему смартфона Nokia N-Gage QD.

С помощью файлового менеджера, так же как и на компьютерных системах, можно выполнять операции по удалению, перемещению, копированию, сортировке, поиску и переименованию файлов. Обойтись без такого вида программ сложно. Что касается рассмотрения процесса *установки* Java- и C++-программ с помощью файлового менеджера, то тут все достаточно просто. Установочный файл может находиться в любом месте файловой системы. С помощью файлового менеджера вы находите необходимый файл и даете команду **Открыть**. Поскольку программы, написанные на Java 2 ME и C++, поставляются в виде заархивированного установочного пакета, то Symbian OS расценит действие по открытию установочного файла как команду к инсталляции программы. В зависимости от вида программы (Java 2 ME или C++) откроется соответствующий инсталлятор **Приложения** или **Диспетчер**. Процесс установки произойдет традиционным способом, который рассмотрен в *разделах 1.5 и 1.6* этой главы.

### **1.6.2. Веб-браузеры**

Не менее важным видом программ являются программы, обеспечивающие доступ в Интернет для просмотра веб-страниц. Таких программ очень много, но на слуху в основном такие «монстры», как Opera, NetFront и Browser. С помощью

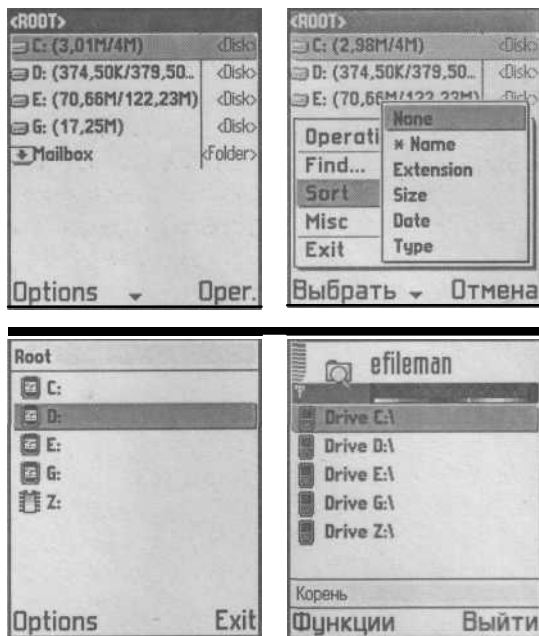


Рис. 1.12. Файловые менеджеры FileMan, SeleQ и eFileManager

этих программ можно просматривать веб-страницы в формате HTML. Экран телефона по размеру существенно отличается от монитора, но используемый алгоритм для форматирования больших по размеру веб-страниц делает возможным их просмотр на телефоне. На рис. 1.13 изображен Интернет-браузер Opera, NetFront и Browser.

Мобильные Интернет-браузеры позволяют не только просматривать HTML страницы, но и загрузить необходимый файл по каналу GPRS. Загрузить файл из Интернета можно и с применением специализированных программ (например, fGet). С помощью этой программы по прямой ссылке можно загрузить любой файл и что самое главное, программа fGet поддерживает дозагрузку файлов при обрыве связи!

### 1.6.3. Мультимедиа

Набор программ в этом секторе рынка немалый: это и программы для просмотра видео, MP3-плееры, программы для просмотра изображений, конвертировщики и так далее. В Symbian OS можно просматривать видео-файлы в формате \*.3gp и \*.nim, но качество их все же оставляет желать лучшего. Явным фаворитом здесь выглядит программа SmartMovie, воспроизводящая на телефоне видео в формате AVI очень приличного качества. Программа SmartMovie поставляется вместе с конвертировщиком, который устанавливается на компьютер и перекодирует исходный видео-файл в формат AVI для телефона.



Рис. 1.13. Интернет-браузер Opera, NetFront и Browser

В некоторых смартфонах воспроизведение MP3-файлов происходит аппаратно. Существует множество MP3-плееров для Symbian OS, поэтому выделить какую-то одну программу трудно. Все плееры обеспечивают качественное прослушивание MP3-файлов, самый известный из них - UltraMP3. На рис. 1.14 изображены программы SmartMovie и UltraMP3.



Рис. 1.14. Программы Smart Movie и UltraMP3

### **1.6.4. Игры**

Этот раздел можно сделать бесконечно длинным - игровая индустрия мобильных игр развивается быстрыми темпами, огромный спрос пользователей стимулирует выход все новых и новых игр. На смену двухмерных игр пришли полноценные трехмерные игры с мощной графикой. Не зря компания Nokia создала линейку смартфонов под названием N-Gage, позиционирующихся как игровой пульт. Кроме игр с расширением \*.sis в N-Gage доступен специальный класс игр, поставляемых на картах памяти. Можно с уверенностью сказать, что мобильные игры, написанные как на Java 2 ME, так и на C++, будут завоевывать у пользователей все большую и большую популярность.

Помимо перечисленных видов программ существует множество всевозможных программ, например, для работы с архивами, чтения книг, просмотра файлов Word, Excel и Power Point, которые мы просто не в силах рассмотреть. Для вас, как для программиста, открывается большое поле деятельности и, учитывая, что сроки создания программ для Symbian OS могут исчисляться всего неделями, интерес к этой сфере деятельности только увеличивается.

В следующих трех главах мы поговорим об инструментальных средствах разработки, служащих для написания программ на языке программирования C++ в Symbian OS.



## Глава 2. Среда программирования IDE Metrowerks CodeWarrior for Symbian OS

На сегодняшний день в мире существует множество инструментальных средств программирования, необходимых для создания программ под различные платформы. По своим возможностям ведущие среды программирования примерно равны. Все они имеют в своем арсенале визуальную графическую оболочку, компилятор, линковщик, отладчик, систему справки и поддержки, механизм обновлений и многое другое. Выбор определенной среды программирования зависит в основном от направленности разрабатываемого проекта и личных пристрастий программиста. Дискутировать на тему что лучше: продукты от Microsoft, Borland или Sun, смысла нет. По большому счету программист, работающий с продуктами компании Borland, открыв в первый раз, например, Visual Studio.NET не растеряется и разберется в тонкостях работы этой среды довольно быстро, но дело как раз совсем не во внешнем виде. Каждая среда программирования имеет свою направленность, то есть при ее создании разработчики четко представляют круг задач, которые с помощью этой среды программирования можно решать. Специализированные инструментарии, направленные на решение конкретной задачи, -это и есть приоритет, по которому необходимо выбирать средство программирования. На сегодняшний день на рынке представлены две специализированные среды для разработки программ под Symbian OS: C++ BuilderX Mobile Studio от компании Borland и CodeWarrior for Symbian от компании Metrowerks. С ними мы и познакомимся. Учитывая тот факт, что книг по этим средствам программирования нет, информация, полученная в этой и следующей главах, будет довольно ценной.

Что касается среды программирования Visual C++, создание программ в этой среде возможно, но связано с большими трудностями. Среда Visual C++ не поддерживает процессоры типа ARM и потребуются выполнить множество настроек, чтобы с ней работать. Нужно не забывать и о том, что Microsoft имеет свою мобильную платформу Windows Mobile 2003 (компания Microsoft не давно анонсировала новую платформу Windows Mobile 5), а Symbian OS - это прямой конкурент. И если создание Symbian приложений в Visual C++.6 и Visual C++.NET еще возможно, то в будущем такая возможность вряд ли будет иметь место. Если же вам посчастливится найти работу в компании, специализирующейся на разработке программ для Symbian OS, то первым требованием будет знание средств программирования CodeWarrior for Symbian, C++ BuilderX Mobile Studio и командной строки. Сведения по работе с командной строкой можно найти в справочной информации SDK. О продукции компании Borland мы поговорим в эле-

дующей главе, а сейчас перейдем к рассмотрению среды CodeWarrior от компании Metrowerks.

Metrowerks -очень крупная и известная компания, выпускающая большое количество программных продуктов для платформ Windows, Linux, Unix, Palm, Mac и Symbian. Достаточно зайти на сайт компании [www.metrowerks.com](http://www.metrowerks.com) и ознакомиться с ассортиментом представленных программ. Если вы никогда не сталкивались с продуктами Metrowerks, то будете приятно удивлены. Компания предоставляет инструментарию для создания программ под Windows, Palm, Symbian, PlayStation 2, Game Boy, Game Boy Advanced, Java 2 ME.

Что касается платформы Symbian, то имеется несколько версий средств программирования, это:

- IDE Metrowerks CodeWarrior Personal 2.8 for Symbian OS,
- IDE Metrowerks CodeWarrior Professional 3.0 for Symbian OS,
- IDE Metrowerks CodeWarrior OEM 3.0 for Symbian OS.

Версии CodeWarrior Personal и Professional могут поставляться в коробках, а версия CodeWarrior OEM - это версия Professional, только в облегченном виде без коробки, рассчитанная в основном на крупные компании с большим количеством рабочих мест. Версия CodeWarrior Professional имеет в своем составе несколько SDK, она размером в 103 Мб, и ее стоимость гораздо выше, чем версия CodeWarrior Personal. Поэтому в этой главе мы будем рассматривать работу CodeWarrior Personal 2.8 for Symbian OS, тем более что эта trial-версия работает 90 дней, тогда как Professional и OEM всего 15 дней. Но все версии одинаковы, и разобравшись с работой одной из них, вы разберетесь и с остальными.

Среду программирования CodeWarrior Personal 2.8 for Symbian OS вы сможете загрузить с сайта компаний Metrowerks и Nokia. На сайтах этих компаний находятся trial-версии продуктов с ограниченным сроком работы (90 дней). Программа «весит» 64 Мб и конечно при 12 Кбит/с процесс скачивания программы может затянуться. К сожалению, согласно лицензии распространение trial-версии этой среды на CD с книгой невозможно.

Среду программирования CodeWarrior можно так же загрузить с сайта компании Nokia — просто компания Nokia купила права на CodeWarrior for Symbian OS у компании Metrowerks. При загрузке CodeWarrior с сайта Nokia ([http:// forum.nokia.com](http://forum.nokia.com)), необходимо зарегистрироваться на сайте этой компании. Правильно заполните все поля и отошлите форму на сервер, нажав кнопку подтверждения. После этого на указанный адрес электронной почты придет письмо со ссылкой, перейдя по которой, вы подтвердите свои намерения о регистрации. И затем получите еще одно письмо, уведомляющее о регистрации на сайте Nokia. Далее как зарегистрированный пользователь вы вправе пользоваться ресурсами сайта и, в том числе, скачивать программные средства. Аналогичная ситуация существует и в компании Symbian Ltd., зарегистрировавшись на сайте [http:// www.symbian.com/developer](http://www.symbian.com/developer), изображенной на рис. 2.1, вам станут доступны все ресурсы системы.

А теперь давайте приступим к рассмотрению IDE Metrowerks CodeWarrior Personal 2.8 for Symbian OS, начиная с инсталляции на ваш компьютер. В даль-

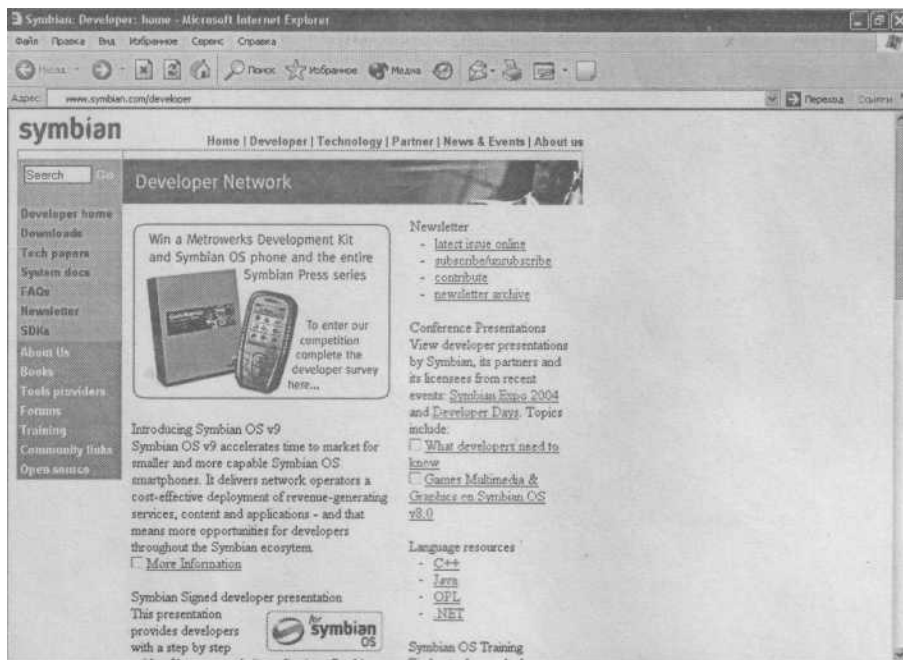


Рис. 2.1. Сайт компании Symbian Ltd

нейшем при упоминании аббревиатуры CodeWarrior будет иметься в виду версия IDE Metrowerks CodeWarrior Personal 2.8 for Symbian OS.

## 2.1. Установка CodeWarrior for Symbian Personal v2.8.3

1. Двойной клик на установочном файле CW\_Symbian\_Personal\_2.8.3 запустит мастер установки программы Metrowerks CodeWarrior for Symbian OS. В первом появившемся диалоговом окне **Welcome to the InstallShield Wizard** вы

увидите традиционное приветствие (см. рис. 2.2). Нажмите кнопку **Next** для продолжения процесса установки программы.

2. Затем появится небольшое по размеру информационное диалоговое окно (см. рис. 2.3), где оговаривается время работы устанавливаемого продукта на

ваш компьютер. Сроки работы программы, к сожалению, ограничены - это 90 дней со дня инсталляции, после чего вам придется либо приобрести Metrowerks CodeWarrior, либо удалить его с машины. Нажмите кнопку **OK** для продолжения установки.

3. Следующее диалоговое окно под названием **License Agreement** ознако-

мит вас с условиями лицензионного соглашения. При согласии с выдвигаемыми требованиями установите флажок в поле **I accept the terms of the license agreement** и нажмите кнопку **Next**.

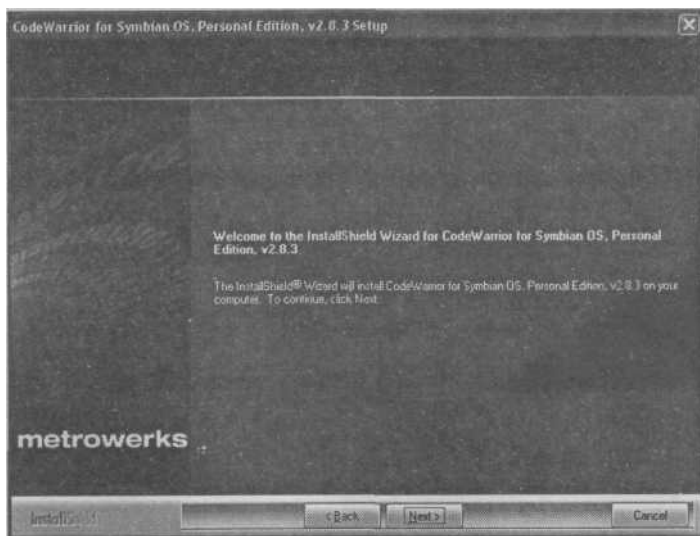


Рис. 2.2. Диалоговое окно  
Welcome to the InstallShield Wizard

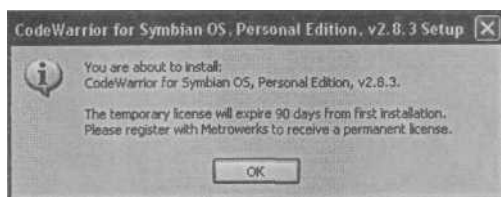


Рис. 2.3. Информационное окно

4. Появившееся диалоговое окно **Product Information** представит новые возможности и обновления текущей версии Metrowerks CodeWarrior.

Нажмите

кнопку **Next** для продолжения инсталляции программы.

5. Далее откроется диалоговое окно **Setup Type**, изображенное на рис. 2.4,

где при необходимости можно изменить каталог для установки программы, а так же выбрать тип установки. Для выбора есть два варианта: **Complete** (Полный) и

**Custom** (Выборочный). Всегда полезно знать, что именно программа установки

инсталлирует на компьютер, поэтому выберем **Custom** и нажмем кнопку **Next**.

6. Для выбора в диалоговом окне **Select Components**, представленном на

рис. 2.5, предлагаются три пункта: **CodeWarrior Tools**, **CodeWarrior Manuals** и

**Trill Seekers**. Для полной установки необходимо выбрать все пункты, а на жест

ком диске вам понадобится примерно 170 Мб свободного пространства. Для про

должения установки нажмите кнопку **Next**.

7. В следующем диалоговом окне **Select Program Folders** необходимо про

извести выбор папки в меню **Пуск**, где будет располагаться группа иконок для

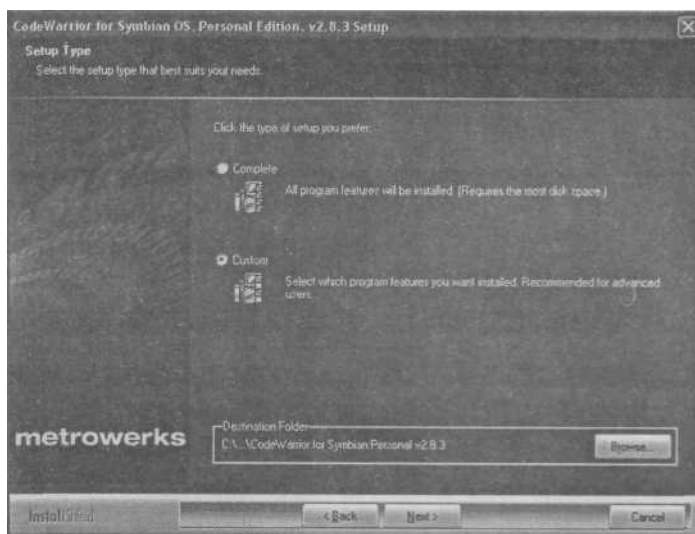


Рис. 2.4. Диалоговое окно Setup Type

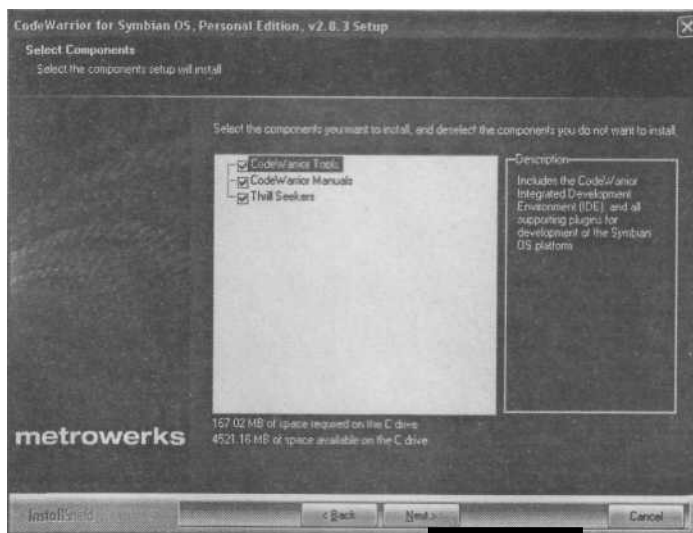


Рис. 2.5. Диалоговое окно Select Components

быстрого старта программы. Выберите нужную папку или оставьте значения по умолчанию и нажмите кнопку Next.

8. Затем появится диалоговое окно Select File Association Options, представленное на рис. 2.6. В этом окне перечислены предполагаемые файлы с различными расширениями, которые в последствии будут ассоциироваться со ере-

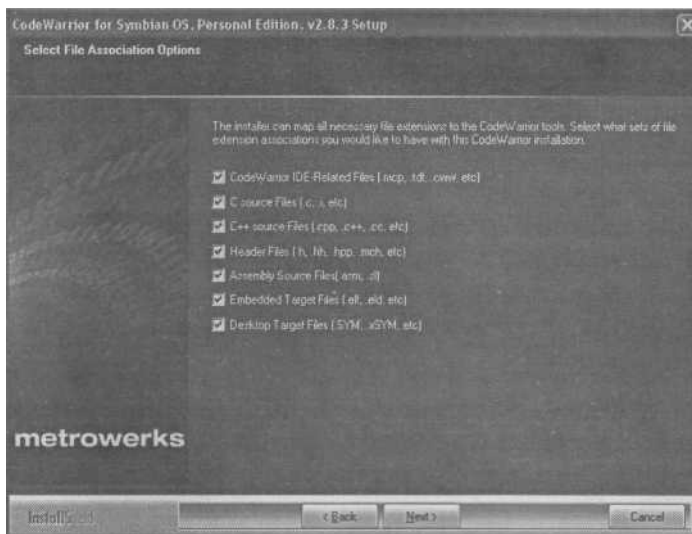


Рис. 2.6. Диалоговое окно Select File Association Options

дой программирования Metrowerks CodeWarrior. Выберите напротив необходимых вам полей флажки и нажмите кнопку Next.

9. Следующее окно содержит общую информацию об устанавливаемых компонентах и каталогах выбранных в процессе настройки инсталляции программы. Нажмите кнопку Next и процесс установки вступит в окончательную стадию. По завершению инсталляции появится последнее диалоговое окно с предложением о перезагрузке компьютерной системы.

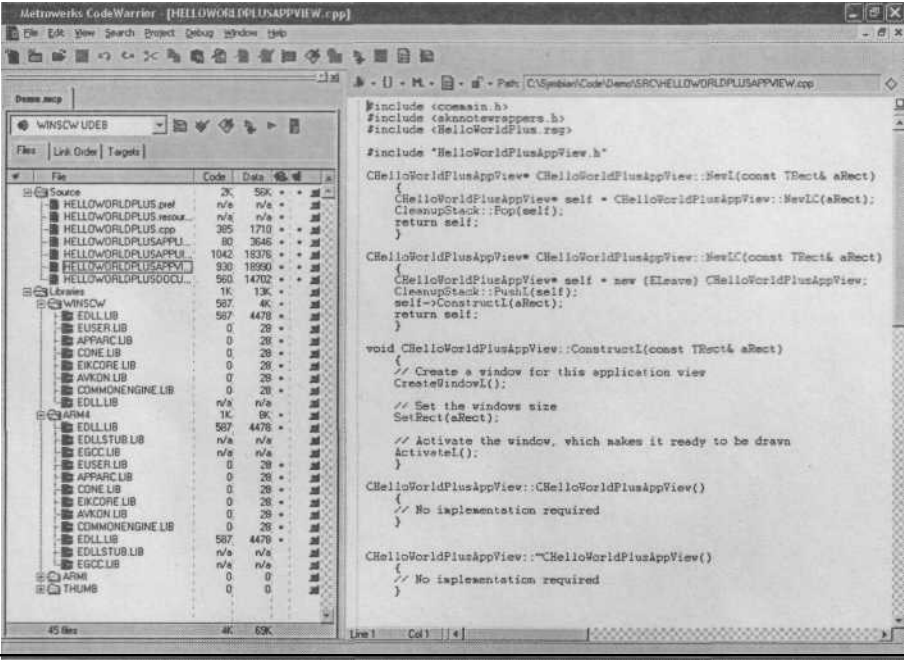
## 2.2. Знакомство с Metrowerks CodeWarrior for Symbian Personal v2.8.3

После установки Metrowerks CodeWarrior в меню Пуск будет сформирована групповое меню **CodeWarrior for Symbian Personal v2.8.3**, состоящее из пяти команд.

- **CodeWarrior Help** открывает контекстную справку Metrowerks CodeWarrior в формате CHM;
- **CodeWarrior IDE** запускает среду программирования Metrowerks CodeWarrior for Symbian Personal v2.8.3;
- **CodeWarrior Updater** дает возможность обновить текущую версию среды программирования, но требуется связь с Интернет;
- **Quick Start Guide** - небольшое справочное пособие по работе с Metrowerks CodeWarrior в формате PDF;
- **Release Note** - описание выпуска Metrowerks CodeWarrior версии 2.8.

Для *запуска программы* выберите команду из меню Пуск => **Все программы** => **Metrowerks CodeWarrior** => **CodeWarrior for Symbian Personal v2.8.3** => **Co-**

**deWarrior IDE.** Откроемся рабочее окно среды программирования Metrowerks CodeWarrior. Изначально при запуске среды отображается пустое рабочее пространство с командами меню и панелью инструментов, и только после раскрытия рабочего проекта, Metrowerks CodeWarrior открывает текстовый редактор и сопутствующие панели. На рис. 2.7 изображено рабочее пространство Metrowerks CodeWarrior с открытым проектом HelloWorldPlus.



**Рис. 2.7.** Рабочее окно Metrowerks CodeWarrior с проектом HelloWorldPlus

При открытии проекта, рабочее пространство в среде Metrowerks CodeWarrior разделяется на две области: окно текстового редактора и окно **Workspace** (Рабочее пространство). В верхней части Metrowerks CodeWarrior располагается линейка меню и панель инструментов с кнопками для быстрой работы с системой.

Линейка меню в среде Metrowerks CodeWarrior содержит набор команд для открытия, закрытия, редактирования файлов, компиляции проектов, установки опций, отладки, запуска эмулятора и многого другого. Это стандартные команды, применяемые практически во всех средствах программирования приложений. Некоторые команды имеют вторичные меню, а некоторые команды при их выборе открывают диалоговые окна. Если около названия команды в меню присутствует небольшой по размеру черный треугольник, то пункт меню содержит дополнительное, вторичное меню. Если же пункт меню содержит многоточие, то

при выборе этой команды открывается диалоговое окно. Рассмотрим подробно существующие команды меню.

### 2.2.1. Меню File

В меню **File**, изображенном на рис. 2.8, содержатся следующие команды:

- **New** (Ctrl+Shift+N) - открыть диалоговое окно для создания нового проекта или файла;
- **Open** (Ctrl+O) - открыть проект или файл;
- **Find and Open** (Ctrl+D) - найти и открыть искомый файл;
- **Close** (Ctrl+W) - закрыть текущие открытые файлы;
- **Save** (Ctrl+S) - сохранить открытый файл;
- **Save All** (Ctrl+Shift+S) - сохранить все файлы;
- **Save As** - сохранить файлы в заданной директории;
- **Save A Copy As** - сохранить файлы и копировать в заданную директорию;
- **Revert** - заменить текущий файл его предварительно сохраненной версией;
- **Open Workspace** - открыть окно **Workspace**;
- **Close Workspace** - закрыть окно **Workspace**;
- **Save Workspace As** - сохранить окно **Workspace**;
- **Import Project From .mmp File** - импорт проекта через файл MMP;
- **Import Project** - импорт проекта в формате XML;
- **Export Project** - экспорт проекта;
- **Page Setup** - предварительный просмотр;
- **Print** (Ctrl+P) - печать файла;
- **Open Recent** - перечисление пяти последних открытых файлов;
- **Exit** - выход из Metrowerks CodeWarrior.

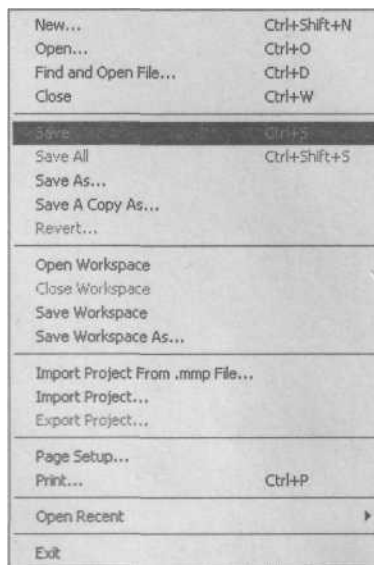


Рис. 2.8. Меню File

### 2.2.2. Меню Edit

В меню **Edit**, показанном на рис. 2.9, содержатся следующие команды:

- **Undo** (Ctrl+Z) - шаг назад;
- **Redo** (Ctrl+Shift+Z) - шаг вперед;
- **Cut** (Ctrl+X) - вырезать;
- **Copy** (Ctrl+C) - копировать;
- **Paste** (Ctrl+V) - вставить;



- **Delete** (Delete) - удалить;
- **Select All** (Ctrl+A) - выделить все;
- **Q Balance** (Ctrl+B) - выделить текст в фигурных скобках от курсора и до окончания скобок;
- **Shift Left** (Ctrl+[) - сдвинуть текст влево на одну позицию табуляции;
- **Shift Right** (Ctrl+]) - сдвинуть текст вправо на одну позицию табуляции;
- **Get Previous Completion** (Alt+Shift+/) - действует в обход команды **Code Completion**, давая возможность вставить текущий символ;
- **Get Next Completion** (Alt+/) - действует в обход команды **Code Completion**, давая возможность вставить следующий символ;
- **Code Completion** - настройка автоматического завершения исходного кода;
- **Preference** - свойства Metrowerks CodeWarrior;
- **WINSW UDEB Settings** (Alt+F7) - эта команда может меняться и зависит от выбранной платформы для процесса компиляции (подробности в *разделах 2.6 и 2.7*);
- **Version Control Settings** осуществляет контроль над версией VCS;
- **Commands and Key Bindings** открывает выборочные команды Metrowerks CodeWarrior;
- **Symbian Environments** открывает диалоговое окно для настройки подключенных в CodeWarrior эмуляторов телефонов.

### 2.2.3. Меню View

В *меню View*, изображенном на рис. 2.10, содержатся команды:

- **Toolbar**- настройка панели **Toolbar** (Инструментальная панель), имеет вложенное меню с набором команд:

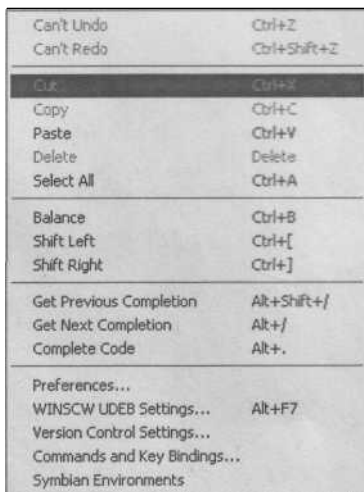


Рис. 2.9. Меню Edit

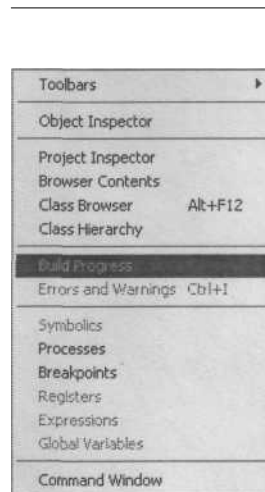


Рис. 2.10. Меню View

- **Hide Window Toolbar** - скрывает **Window Toolbar**;
- **Reset Window Toolbar** - сброс **Window Toolbar**;
- **Clear Window Toolbar** — отчищает **Main Toolbar**;
- **Hide Main Toolbar** - скрывает **Main Toolbar**;
- **Reset Main Toolbar** - сброс **Main Toolbar**;
- **Clear Main Toolbar** - отчищает **Main Toolbar**.
- **Object Inspector** - конфигурирует объекты исходного файла;
- **Project Inspector** - конфигурирует текущий проект;
- **Browser Contents** - обзор системной библиотеки;
- **Class Browser (Alt+F12)** - обзор классов;
- **Class Hierarchy** - обзор иерархии системной библиотеки;
- **Build Progress** - открывает окно **Build Progress**;
- **Errors and Warnings (Ctrl+I)** - открывает окно ошибок и предупреждений;
- **Symbolics** - открывает окно **Symbolics** для исследования текущих файлов проекта;
- **Processes** - открывает окно **Processes**;
- **Breakpoints** - открывает список текущих настроек точек останова;
- **Registers** - предоставляет возможность просмотра регистров;
- **Expressions** - открывает окно **Expressions**;
- **Global Variables** - показывает глобальные переменные проекта;
- **Command Windows** - открывает окно **Command Windows**.

## 2.2.4. Меню Search

В меню Search, изображенном на рис. 2.11, содержатся следующие команды:

- **Find (Ctrl+F)** - найти;
- **Replace (Ctrl+H)** - заменить;
- **Find in Files (Ctrl+Shift+M)** - найти в файле;
- **Find Next (F3)** - найти следующие;
- **Find in Next File (Ctrl+T)** - найти в следующем файле;
- **Enter Find String (Ctrl+E)** - ввести искомую строку;
- **Find Selection (Ctrl+F3)** - найти выбранное;
- **Replace Selection (Ctrl+=)** - заменить выделенное;
- **Replace and Find Next (Ctrl+L)** - заменить и найти следующие;
- **Replace All** - заменить все;
- **Find Definition (Ctrl+')** - найти определение;
- **Go Back (Ctrl+Shift+B)** - вернуться;
- **Go Forward (Ctrl+Shift+F)** - идти вперед;
- **Go to Line (Ctrl+G)** - перейти на строку;
- **Compare Files** - открывает окно для сравнения файлов, папок;

Find...	Ctrl+F
Replace...	Ctrl+H
Find In Files...	Ctrl+Shift+M
Find Next	F3
Find In Next File	Ctrl+T
Enter Find String	Ctrl+E
Find Selection	Ctrl+F3
Replace Selection	Ctrl+=
Replace and Find Next	Ctrl+L
Replace All	
Find Definition	Ctrl+'
Go Back	Ctrl+Shift+B
Go Forward	Ctrl+Shift+F
Go to Line...	Ctrl+G
Compare Files...	
Apply Difference	
Unapply Difference	

Рис. 2.11. Меню Search

- Apply Difference - взять определение;
- **Unapply Difference** - вернуть определение.

## 2.2.5. Меню Project

В меню **Project**, изображенном на рис. 2.12, содержатся команды:

- **Add** - добавить в проект;
- **Add Files** - добавить в проект файл;
- **Create Group** - сформировать группу;
- **Create Target** - выбрать платформу;
- **Create Design** - создает новый дизайн проекта;
- **Check Syntax (Ctrl+;)** - проверяет синтаксис;
- **Precompile** - пред компиляция;
- **Compile (Ctrl+F7)** - компиляция;
- **Disassemble (Ctrl+Shift+F7)** - дизассемблирование;
- **Make (F7)** - сделать сборку проекта;
- **Stop Build (Ctrl+Break)** - остановить компиляцию;
- **Remove Object Code (Ctrl+-)** — удалить сформированный объектный код из проекта;
- **Re-search for Files** - повторный поиск;
- **Reset Project Entry Patch** - сброс настроек для подключений в проект;
- **Synchronize Modification Date** - синхронизация модифицированных данных;

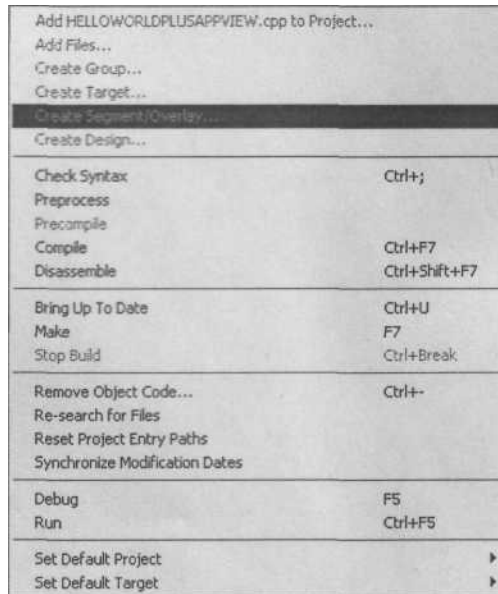


Рис. 2.12. Меню Project

- **Debug (F5)** - отладка проекта;
- **Run (Ctrl+F5)** - запустить тестирование программы на эмуляторе;
- **Set Default Project** - перечисляются доступные файлы;
- **Set Default Target** - выбор платформы, под которую будет совершаться компиляция проекта, имеет вложенное меню с изменяющимся набором команд:
  - **WINSW UDEB** - отладочная версия для эмулятора;
  - **WINSW UREL** - конечная версия для эмулятора;
  - **ARM1 UDEB** - отладочная версия для платформы ARM1;
  - **ARM1 UREL** - конечная версия для платформы ARM1;
  - **THUMB UDEB** - отладочная версия для платформы THUMB;
  - **THUMB UREL** - конечная версия для платформы THUMB;
  - **Build All** - выбрать все платформы.

## 2.2.6. Меню Debug

В меню **Debug**, изображенном на рис. 2.13, имеются команды:

- **Break** - остановить;
- **Kill (Shift+F5)** - «убить» все процессы;
- **Restart (Ctrl+Shift+F5)** - сначала;
- **Step Over (F10)** - выполнить следующий оператор и на этом остановиться;
- **Step Into (F11)** - выполнить следующий оператор, но если это вход в функцию, то войти в нее и на этом остановиться;
- **Step Out (Shift+F11)** - выполнить функцию и остановиться на операторе вызвавшей эту функцию;
- **Run to Cursor** - перейти к курсору;
- **Change Program Counter** - установка стрелки;
- **Set Breakpoint (F9)** - установить точку останова;
- **Set Eventpoint** - открывает вторичное меню где можно установить следующие события:
- **Set Log Point** - установка логов системы;
- **Set Script Point** - установка сценария;
- **Set Skip Point** - установка пропуска точки;
- **Set Sound Point** - установка звукового сигнала;

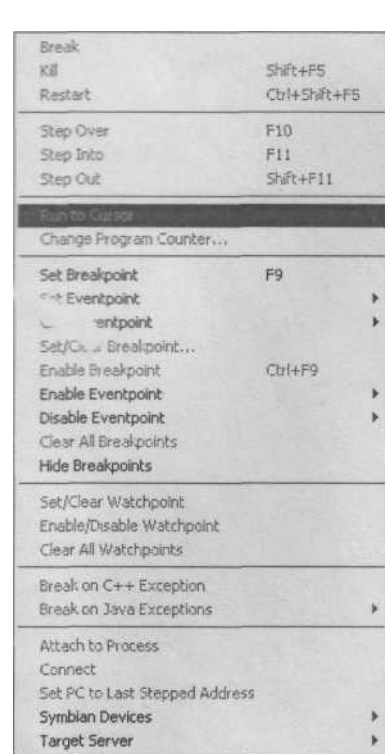


Рис. 2.13. Меню Debug

- **Set Trace Collection off** - выключает трассировку;
- **Set Trace Collection on** - включает трассировку.
- **Clear Eventpoint** - чистит установленные события;
- **Clear Log Point** - чистит установленные **Log Point**;
- **Clear Script Point** - чистит установленные сценарии;
- **Clear Skip Point** - удаляет пропуски точек;
- **Clear Sound Point** - удаляет метки звуковых сигналов;
- **Clear Trace Collection off** - чистит выключенную трассировку;
- **Clear Trace Collection on** - чистит включенную трассировку;
- **Set/Clear Breakpoint** - установка или очистка точки останова;
- **Enable Breakpoint (Ctrl+F9)** - подключение точки останова;
- **Enable Eventpoint** - подключение следующих событий;
- **Enable Log Point** - делает доступной установку логов системы;
- **Enable Script Point** - делает доступной установку скриптов;
- **Enable Skip Point** - делает доступной установку точек пропуска;
- **Enable Sound Point** - делает доступной установку звуковых меток;
- **Enable Trace Collection off** - делает доступной установку выключения трассировки;
- **Enable Trace Collection on** — делает доступной установку включения трассировки.

**Q Disable Eventpoint** — отключение следующих событий:

- **Disable Log Point** - отключает установку логов системы;
- **Disable Script Point** - отключает установку скриптов;
- **Disable Skip Point** - отключает установку точек пропуска;
- **Disable Sound Point** - отключает установку звуковых меток;
- **Disable Trace Collection off** - отключает установку выключения трассировки;
- **Disable Trace Collection on** — отключает установку включения трассировки.
- **Clear All Breakpoint** - чистит установленные точки останова;
- **Hide Breakpoints** - убирает точки останова;
- **Set/Clear Watchpoint** - устанавливает или чистит **Watchpoint**;
- **Enable/Disable Watchpoint** - разрешить или запретить использование **Watchpoint**;
- **Clear All Watchpoint** - очистить все установленные **Watchpoint**;
- **Break on C++ Exception** - остановить исключения для C++;
- **Break on Java Exceptions** - остановить исключения для Java;
- **Attach to Process** - подключиться к процессу;
- **Connect** - связь;
- **Symbian Device** - устройства Symbian OS;
- **Target Server** - определить сервер для связи.

## 2.2.7. Меню Window

В меню **Window**, изображенном на рис. 2.14, содержатся команды: □ **Close (Ctrl+W)** - закрывает текущее окно;

- **Close All Editors Document** (Ctrl+Shift+W) - закрывает все документы от  
крытые в текстовом редакторе;
- Q **Cascade** — выстраивает окна каскадом;
- **Tile Horizontally** - располагает окна по горизонтали;
- **Tile Vertically** - располагает окна по вертикали.

## 2.2.8. Меню Help

В меню **Help**, показанном на рис. 2.15, содержатся команды:

- **CodeWarrior Help** — справочная система Metrowerks CodeWarrior;
- **Index** - открывает справочную систему Metrowerks CodeWarrior на  
стра  
нице Индекс;
- **Search** — открывает справочную систему Metrowerks CodeWarrior на  
стра  
нице Поиск;
- **Symbian Release Note** - открывает информацию о релизе Metrowerks  
CodeWarrior for Symbian OS Personal 2.8;
- **Licensee Authorization** - лицензионное соглашение;
- **Metrowerks Website** - переход на веб страницу компании Metrowerks;
- **About Metrowerks CodeWarrior** - открывает диалоговое окно с  
информа  
цией о Metrowerks CodeWarrior for Symbian OS Personal 2.8.

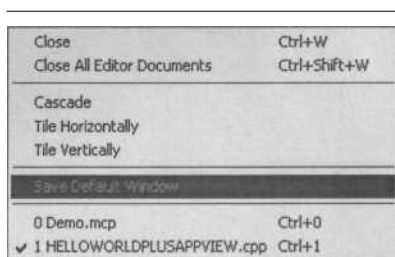


Рис. 2.14. Меню Window

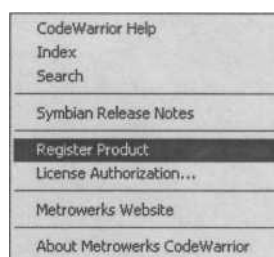


Рис. 2.15. Меню Help

## 2.2.9. Панель инструментов

Панель инструментов Metrowerks CodeWarrior выполнена в виде набора пиктограмм или кнопок, каждая из которых связана с определенным пунктом меню для осуществления быстрой работы с программой. Все кнопки активны, но, в зависимости от прделываемой работы в среде программирования, некоторые кнопки могут быть отключены и недоступны. На рис. 2.16 представлена панель



Рис. 2.16. Панель инструментов Metrowerks CodeWarrior

инструментов Metrowerks CodeWarrior. При наведении курсора на пиктограмму панели инструментов появляется контекстная подсказка с названием назначенной для этой кнопки команды меню.

- New Text File** - создает новый текстовый файл;
- New** — создает новый файл;
- Open** - открывает необходимый файл;
- Save** — сохранить;
- Undo** - шаг назад;
- Redo** - шаг вперед;
- Cut** - вырезать;
- Copy** - копировать;
- Paste** - вставить;
- Find** — найти;
- Find Next** - найти следующее;
- Replace Selection** - переместить выделенную область;
- Compile** - компилировать;
- Make** - сделать сборку проекта;
- Stop Build** — остановить сборку;
- Debug** - отладка;
- Errors and Warnings** - показать ошибки и предупреждения;
- Preferences** - свойства Metrowerks CodeWarrior;
- WINSW UDEB Settings** - установка свойств платформы.

## 2.2.10. Окно *Workspace*

После запуска среды Metrowerks CodeWarrior, с левой стороны будет располагаться окно **Workspace** (Рабочее пространство), где отображается древовидная структура рабочего проекта. На рис. 2.17 изображено окно **Workspace** с открытым проектом Demamcr.

Окно **Workspace** содержит одну большую текстовую область, разделенную на три вкладки: **Files** (Файлы), **Link Orders** (Порядок линковки) и **Target** (Адресат), а также панель инструментов с кнопками быстрого доступа и списком версий релиза.

Во вкладке **Files** окна **Workspace** при открытии или создании проекта формируются две папки **Source** (Исходные коды) и **Libraries** (Библиотеки). В папке **Source** находятся файлы с исходными кодами, включая все имеющиеся файлы ресурсов. Раскрывается папка нажатием левой кнопкой мыши на иконке, выполненной в виде квадрата с плюсом. После открытия папки **Source**, в окне **Workspace** будет показана древовидная структура проекта. В папке **Libraries** перечислены подключенные и используемые в проекте библиотечные файлы. Напротив названия каждого файла, с правой стороны вкладки **Files** окна **Workspace**, находится таблица с перечислением дислокации файла, его размером, выходной версией релиза, а также небольшая по размеру черная квадратная кнопка около каждого названия файла. Кнопки располагаются в конце таблицы, при нажатии на одной

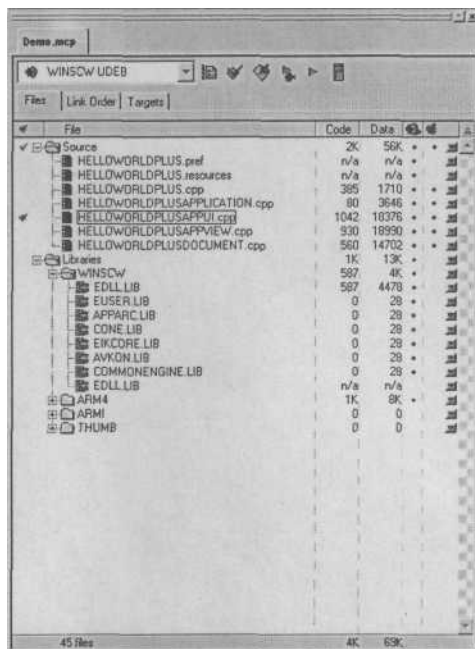


Рис. 2.17. Окно Workspace

из них, появляется контекстное меню, показанное на рис. 2.18, с перечислением подключаемых файлов из системной библиотеки.

Если один и более файлов проекта не откомпилированы или производились какие-то изменения в файлах, то с левой стороны от названий файлов на вкладке Files окна Workspace будет находиться красная галочка. Это своего рода индикатор, сигнализирующий о необходимости компиляции определенных файлов проекта. Посмотрите на рис. 2.17, где с левой стороны от названия файла HELLOWORLDPLUSAPPUI.cpp, а также напротив названия папки Source, присутствует такая галочка.

Вкладка Link Orders окна Workspace содержит перечисление файлов, необходимых для линковки (сборки) проекта. Список файлов упорядочен в соответствии правил сборки проекта. С правой стороны от названий файлов находится идентичная таблица, как и на вкладке Files, с перечисленными свойствами файлов.

Третья, и последняя, вкладка Target окна Workspace содержит перечисление адресатов необходимых для компоновки проекта.

Панель инструментов окна Workspace включает шесть кнопок быстрого доступа (см. рис. 2.17). При наведении курсора мыши появляется подсказка с названием выполняемой команды:

- WINSW UDEB Settings - установка свойств платформы;
- Synchronize Modification Date - синхронизация модифицированных данных;



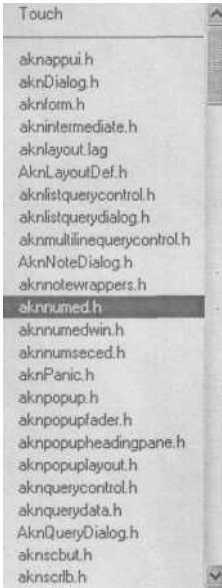


Рис. 2.18. Контекстное меню таблицы на вкладке Files окна Workspace

- **Make** - сделать сборку проекта;
- **Debug** - отладка;
- **Run** - запустить эмулятор;
- **Project Inspector** - инспектор проекта.

Для любой из трех вкладок окна **Workspace** доступна работа с контекстным меню. Нажав правой кнопкой мыши на названии файла, появится меню с перечислением команд. Общее количество команд одинаково, но доступность той или иной команды варьируется от выбранного файла. Рассмотрим все существующие команды контекстного меню окна **Workspace**:

- **Open in Windows Explorer**- открыть в Windows Explorer;
- **Check Syntax** — проверить синтаксис;
- **Preprocess** - препроцессор;
- **Compile** - компилировать;
- **Compile If Dirty** - компилировать, если не компилирован;
- **Disassemble** - дизассемблирование;
- **Add Files** - добавить файл;
- **Create Group** - создать группу;
- **Remove** - удалить.

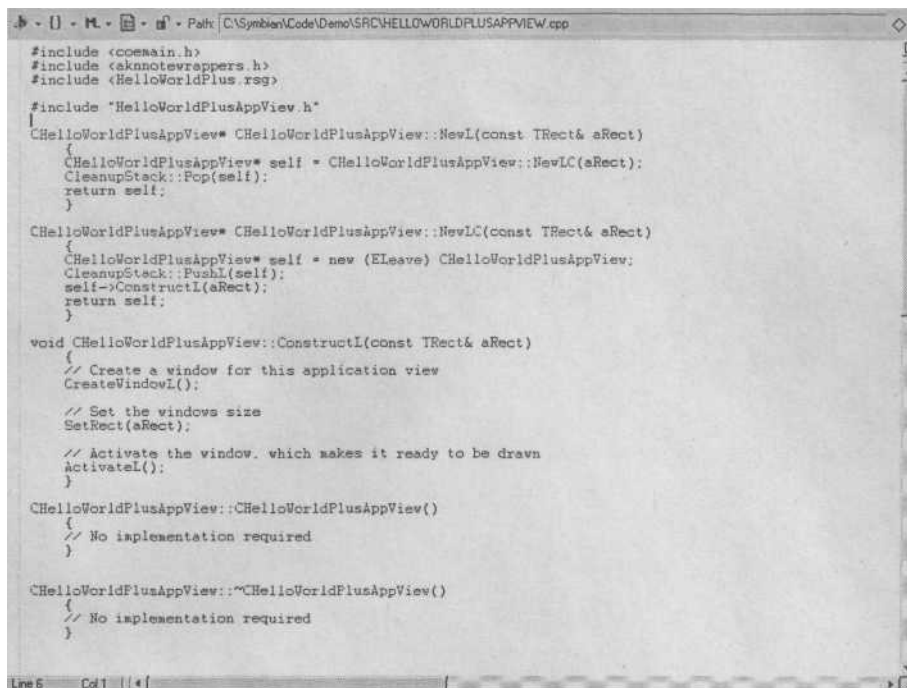
## 2.2.11. Текстовый редактор

Текстовый редактор, изображенный на рис. 2.19, занимает основную часть рабочего окна Metrowerks CodeWarrior и предназначен для работы с исходным кодом программы.

Текстовый редактор имеет встроенную систему распознавания синтаксисов языков C++, Java и Ассемблер. Продуманная подсветка синтаксиса облегчает работу с программным кодом. По своему усмотрению, через команды меню **Edit => Preferences => Editor**, можно настроить цвет ключевых слов. При написании исходного кода в текстовом редакторе, в момент обращения к объекту, то есть, написав название объекта и поставив оператор точка, появится контекстное меню с доступным для этого объекта набором функций.

В верхней части текстового редактора располагается инструментальная панель с элементами управления, которые состоят из кнопок быстрого доступа и нередактируемого текстового поля. В текстовом поле прописан путь к открытому на данный момент файлу. Кнопки быстрого доступа(см. рис. 2.19)- это своего рода подсказки, значительно упрощающие работу с исходным кодом:

- **Header Files** - заголовочные файлы;
- **Functions** - функции;
- **Markers** - маркеры;
- **Document Settings** - установки документа;



```
#include <coemain.h>
#include <sknotewrappers.h>
#include <HelloWorldPlus.rsg>
#include "HelloWorldPlusAppView.h"
}
CHelloWorldPlusAppView* CHelloWorldPlusAppView::NewL(const TRect& aRect)
{
    CHelloWorldPlusAppView* self = CHelloWorldPlusAppView::NewLC(aRect);
    CleanupStack::Pop(self);
    return self;
}

CHelloWorldPlusAppView* CHelloWorldPlusAppView::NewLC(const TRect& aRect)
{
    CHelloWorldPlusAppView* self = new (ELeave) CHelloWorldPlusAppView;
    CleanupStack::PushL(self);
    self->ConstructL(aRect);
    return self;
}

void CHelloWorldPlusAppView::ConstructL(const TRect& aRect)
{
    // Create a window for this application view
    CreateWindowL();

    // Set the windows size
    SetRect(aRect);

    // Activate the window, which makes it ready to be drawn
    ActivateL();
}

CHelloWorldPlusAppView::CHelloWorldPlusAppView()
{
    // No implementation required
}

CHelloWorldPlusAppView::~CHelloWorldPlusAppView()
{
    // No implementation required
}

Line 5 Col 1 / 4
```

Рис. 2.19. Текстовый редактор Metrowerks CodeWarrior

□ **Version Control** - контроль версии.

Редактирование исходного кода осуществляется обычным путем, как и во всех средствах программирования - через меню **Edit** или горячие клавиши, и через панель инструментов и контекстное меню, появляющееся при нажатии правой кнопки мыши в области текстового редактора. Контекстное меню дублирует множество команд из линейки меню Metrowerks CodeWarrior и включает в себя следующие команды:

- **Redo** - шаг назад;
- **Undo** - шаг вперед;
- **Paste** - вставить;
- **Find and Open File** - найти и открыть файл;
- **Compile** - компилировать;
- **Preprocess** - препроцессор;
- **Disassemble** - дизассемблирование;
- **Set Breakpoint** - установить точку останова;
- **Set Eventpoint** - открывает вторичное меню, где можно установить события;
- **Set Log Point** - установка логов системы;
- **Set Script Point** - установка сценария;
- **Set Skip Point** - установка пропуска точки;
- **Set Sound Point** - установка звукового сигнала;

- **Set Trace Collection off** - выключает трассировку;
- **Set Trace Collection on** - включает трассировку;
- **Set Software Breakpoint** - установить программные точки останова.

Нижняя часть текстового редактора содержит **Status Bar** (Панель состояния)

и полосу прокрутки для просмотра исходного кода, который не помещается в основную область редактора. Соответственно есть и горизонтальная полоса прокрутки. Панель состояния Metrowerks CodeWarrior воспроизводит в цифровых значениях активную строку и столбец.

## 2.3. Настройка Metrowerks CodeWarrior

Среда Metrowerks CodeWarrior имеет множественные настройки как для своего внешнего вида, так и для свойств всего проекта. Чтобы настроить Metrowerks CodeWarrior, нажмите на панели инструментов кнопку **References** или выберите в меню команды **Edit => Preferences**. Откроется диалоговое окно **IDE Preferences**, показанное на рис. 2.20.

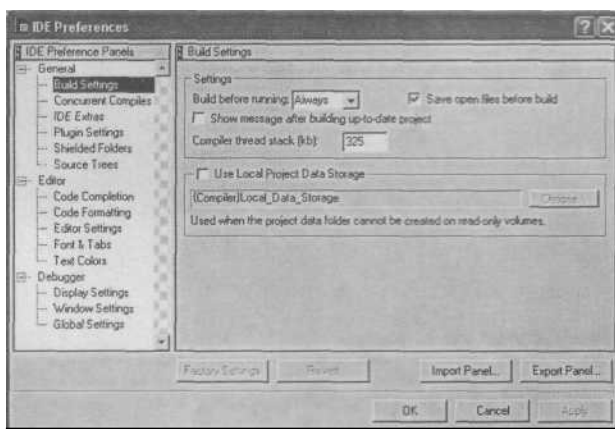


Рис. 2.20. Диалоговое окно IDE Preferences

Окно **IDE Preferences** разделено на две части. С левой стороны располагается область **IDE Preferences Panels**, где представлен список опций, подлежащих настройке. Список выполнен в виде древовидной иерархии и делится на три ключевые группы: **General** (Основной), **Editor** (Редактор) и **Debugger** (Отладчик). Все три группы включают в себя ряд пунктов, переходя по которым вы сможете настраивать определенные опции для работы с Metrowerks CodeWarrior. Каждая панель диалогового окна **IDE Preferences** имеет в нижней части окна семь кнопок:

- **Factory Setting** - сброс измененных опций до настроек по умолчанию;
- **Revert** - возврат установленных опций до сохраненной ранее панели на строек;
- **Import Panel** - сохранение панели в формате XML;

- **Export Panel** - загрузка панели в формате XML;
- **OK** - подтверждение выбранных действий;
- **Cancel** - отмена;
- **Apply** - применить установленные свойства ко всем панелям.

Выделяя курсором мыши необходимый пункт в области **IDE Preferences Panels**, щелкните на нем левой кнопкой, и вы переместитесь на панель с нужными для настройки опциями. Каждый пункт представляет собой новую панель с различными элементами управления для настройки свойств системы и все пункты, как уже говорилось, разделены на три основные группы.

### 2.3.1. Группа *General*

Группа **General** предназначена для конфигурации основных свойств Metro-werks CodeWarrior и содержит пять пунктов, каждый из которых открывает новую панель в диалоговом окне **IDE Preferences**.

#### *Панель Build Settings*

Выбрав пункт **Build Settings** (Настройки компоновки) в группе **General**, откроется панель **Build Settings**, изображенная на рис. 2.20, где происходит установка опций для *настроек компоновки проекта*. Панель **Build Settings** разделена на две статические области с наборами элементов управления. Верхняя область **Settings** (Настройки) содержит следующие опции:

- список **Build before running** устанавливает возможность или невозможность компоновки проекта до запуска его на эмуляторе. Этот список содержит три варианта установок: **Always** (Всегда), **Ask** (По запросу) и **Never** (Никогда);
- флаг **Save open files before build** позволяет автоматически сохранять сохранившиеся активные файлы перед началом компоновки проекта;
- флаг **Show message after building up-to-date project** задает возможность уведомления пользователя о произведенной компоновке проекта;
- текстовое поле **Compiler thread stack** распределяет стек в потоке компиляции проекта (измеряется в килобайтах);

Нижняя область **Use Local Project Data Storage** (Локальное сохранение проектных данных), на панели **Build Settings**, сформирована таким образом, что при выборе флага **Use Local Project Data Storage** становится активным функция выбора сохранения данных в заданной директории. Для этого необходимо нажать кнопку **Choose** (Выбор) и в появившемся окне указать папку для сохранения проектных данных. В случае если флаг **Use Local Project Data Storage** выбран не был, опция сохранения становится не доступной.

#### *Панель Concurrent Compiles*

Панель **Concurrent Compiles** (Параллельная компиляция) следит за выполнением одновременных процессов *компиляции*. На панели располагается флаг **Use Concurrent Compiles**, выбор которого инициализирует доступность переключателей **Recommended** (Рекомендовано) и **User specified** (Установки поль-

зователя). Переключатель **Recommended** включает тройственный режим параллельной компиляции, который является установкой по умолчанию, и режим **User specified** для установок собственных значений. При выборе переключателя **User specified** становится доступным инкрементный регулятор (up-down control) с числовыми значениями для выбора значений параллельной компиляции.

## Панель IDE Extras

Панель **IDE Extras** (Дополнения к IDE), изображенная на рис. 2.21, содержит *дополнительные настройки* среды программирования Metrowerks Code-Warrior.

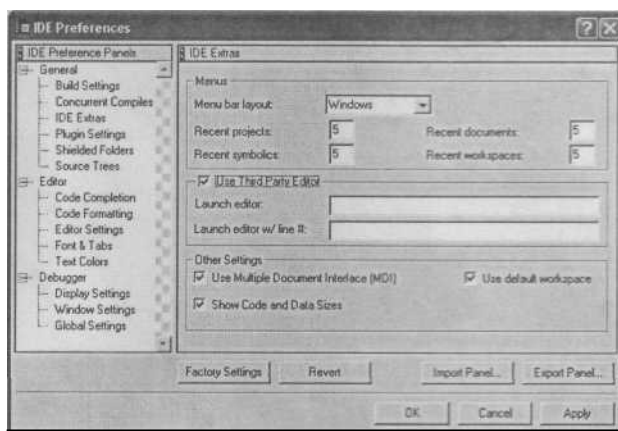


Рис. 2.21. Панель IDE Extras

Панель **IDE Extras** поделена на три статические области: **Menu** (Меню), **Use Third Party Editor** (Дополнительный редактор) и **Other Settings** (Другие установки). Область **Menu** включает в себя следующие пять элементов управления:

- Список **Menu bar layout** избирает вид меню для Windows или Macintosh;
- Текстовое поле **Recent projects** устанавливает значение, на основании которого определяется число доступных последних проектов;
- Текстовое поле **Recent symbolic** устанавливает значение, на основании которого определяется число доступных последних символов;
- Текстовое поле **Recent documents** устанавливает значение, на основании которого определяется число доступных последних документов;
- Текстовое поле **Recent workspace** устанавливает значение, на основании которого определяется число доступных последних **Workspace**.

Следующая область, **Party Editor**, дает возможность выбора стороннего текстового редактора для работы с исходным кодом программы. При активизации флага **Use Third Party Editor**, становятся доступны два текстовых поля **Launch**

**Editor** (Запустить редактор) и **Launch Editor w/Line #** (Запустить редактор командной строки). Деактивизировав флаг **Use Third Party Editor**, вы выключите возможность подключения стороннего текстового редактора.

Последняя область, **Other Settings**, на панели **IDE Extras**, имеет три флага:

- Use Multiple Document Interface** - использовать интерфейс, включающий многодокументный показ данных;
- Show Code and Date Size** - показать размер кода и данных;
- Use default workspace** - использовать по умолчанию окна **Workspace**.

### Панель **Plugin Settings**

Панель **Plugin Setting** (Установки для дополнений), показанная на рис. 2.22, необходима для поиска параметров *неисправности в дополнительно подключаемых модулях* среды Metrowerks CodeWarrior.

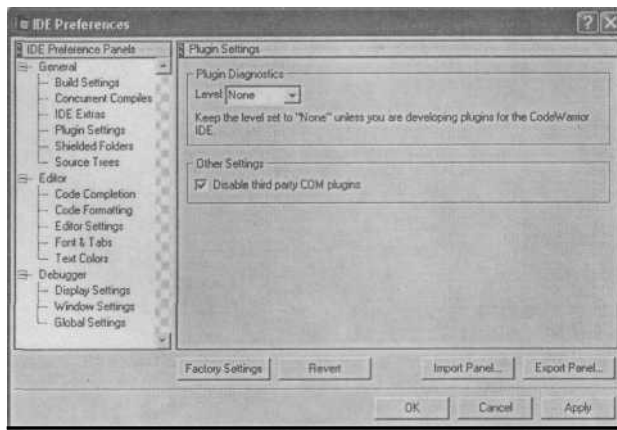


Рис. 2.22. Панель Plugin Setting

Панель **Plugin Setting** разделена на две статические области: **Plugin Diagnostic** (Диагностика дополнительных модулей) и **Other Settings** (Другие настройки). В области **Plugin Diagnostic** располагается список **Level** (Уровень) с тремя значениями для определения диагностических действий: **None** (Не определять), **Errors Only** (Только ошибки) и **All Info** (Вся информация). Область **Other Settings** имеет только один флаг **Disable third party COM Plugins** (Отключить дополнительные модули COM), выбор которого отключает загрузку в Metrowerks CodeWarrior дополнительно подключаемых модулей, созданных на основании COM модели.

### Панель **Shielded Folder**

Панель **Shielded Folder** (Защищенные папки), изображенная на рис. 2.23, предоставляет возможность *игнорировать определенные папки* во время выполнения операций по поиску или сравнению проектных данных.

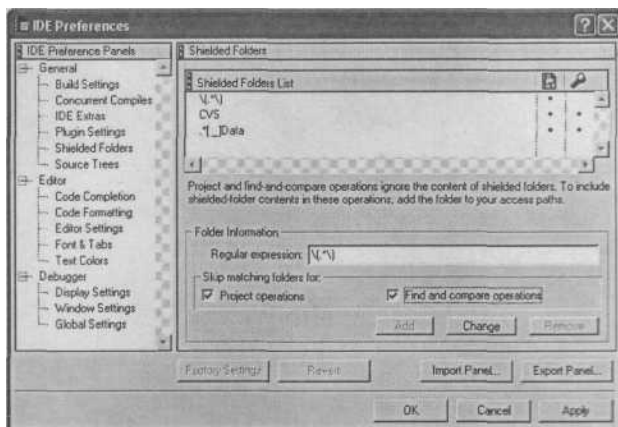


Рис. 2.23. Панель Shielded Folder

На панели **Shielded Folder** вы найдете не редактируемую текстовую область **Shielded Folder List**, где представлен список папок, которые будут игнорироваться Metrowerks CodeWarrior. Для редактирования имеющихся данных в области **Shielded Folder List**, необходимо выделить курсором мыши заданную строку и в текстовом поле **Regular Expression** (Правильное название) прописать название папки. Кроме того, на панели **Shielded Folder** имеется два флага: **Project operations** (Проектные операции) и **Find and compare operations** (Операции поиска и сравнения), выбрав которые, вы пропустите действия по проектным операциям и операциям поиска и сравнения для перечисленных папок в области **Shielded Folder List**. Три дополнительные кнопки на панели **Shielded Folder**, вдобавок к имеющимся постоянным кнопкам, перечисленным в начале *раздела Настройка Metrowerks CodeWarrior*, дают возможность добавлять (Кнопка **Add**), выбирать (Кнопка **Change**) и удалять (**Remove**) соответствующие папки.

### **Панель Source Trees**

Панель **Source Trees** (Исходная древовидная структура) диалогового окна **IDE Preferences**, показанная на рис. 2.24, необходима для изменения или удаления пути к подключенным в среду Metrowerks CodeWarrior инструментальным средствам разработчика (SDK).

Основная и самая большая по размеру область панели **Source Trees** выполнена в виде не редактируемого текстового поля с перечислением подключенных SDK в Metrowerks CodeWarrior. Выделив название необходимого SDK левой кнопкой мыши, в текстовом поле **Name** статической области **Source Trees Information** (Информация для исходной структуры проекта), появится название выбранного SDK, которое можно отредактировать. А уже в области **Type**, с помощью кнопки **Choose**, можно указывать путь к каталогу SDK. Дополнительные кнопки **Add**, **Change** и **Remove** дают возможность добавлять, выбирать или удалять SDK.

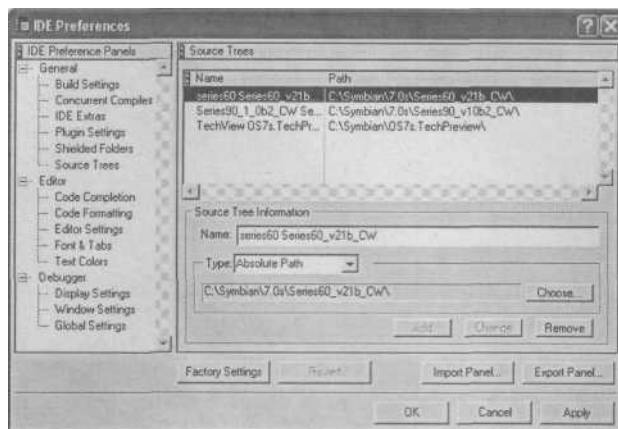


Рис. 2.24. Панель Source Trees

## 2.3.2. Группа Editor

Группа **Editor** (Редактор) предназначена для конфигурации основных параметров текстового редактора среды Metrowerks CodeWarrior. Это форматирование текста, цвет шрифта, отступы, стили текста. В группу **Editor** диалогового окна **IDE Preferences** входят пять пунктов, выбор которых открывает панели для установки соответствующих опций.

### Панель Code Completion

Панель **Code completion** (Комплектация кода), изображенная на рис. 2.25, предназначена для установок связанных с написанием исходного кода в текстовом редакторе.

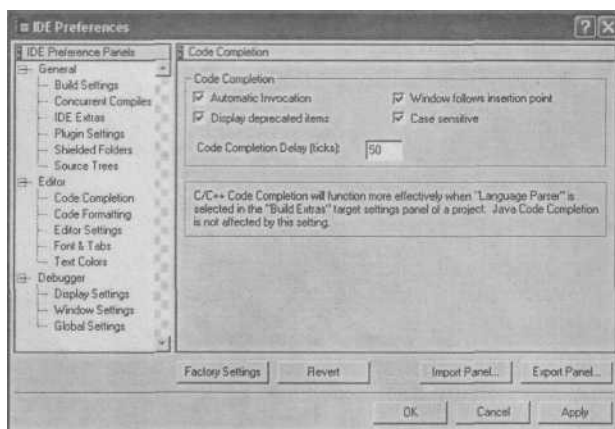


Рис. 2.25. Панель Code completion



На панели имеются пять элементов управления, с помощью которых задаются соответствующие настройки. Рассмотрим элементы управления панели **Code completion**:

- флаг **Automatic Invocation** - автоматическое открытие **Code completion** для завершения кода;
- флаг **Display deprecated items** - устаревшие элементы текста закрашиваются в серый цвет;
- флаг **Case sensitive** производит установку чувствительности набора исходного кода к выбранному регистру на клавиатуре;
- текстовое поле **Code Completion Delay (ticks)** задает время, по которому будет открыта **Code completion**.

### Панель Code Formatting

На панели **Code Formatting** (Форматирование кода), которая представлена на рис. 2.26, находится множество флагов для настройки *стиля написания исходного кода*. Вы можете выбрать стиль написания, применяемый в C++ или Java, а также установить отступы и выравнивание в исходном коде.

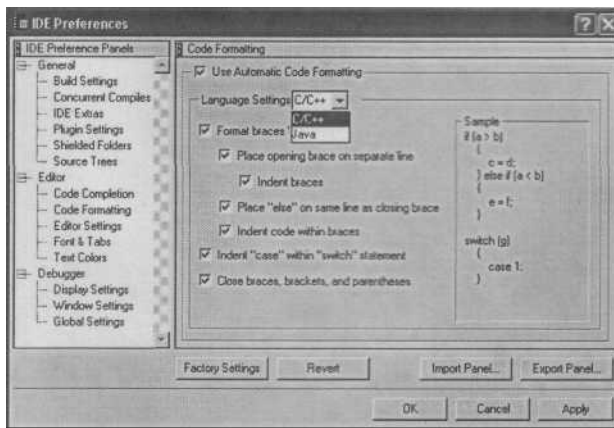


Рис. 2.26. Панель Code Formatting

Давайте рассмотрим элементы управления панели **Code Formatting**:

- флаг **Use Automatic Code Formatting** — этот флаг устанавливает автоматическое форматирование кода в текстовом редакторе согласно установленным настройкам на панели **Code Formatting**;
- список **Language Settings** дает возможность выбрать стиль написания для C++ или Java;
- флаг **Format braces** автоматически устанавливает форматирование закрывающей фигурной скобки;
- флаг **Place opening brace on separate line** задает написание открывающей фигурной скобки с новой строки;

- флаг **Indent braces** - это выравнивание фигурных скобок клавишей табуляции;
- флаг **Place "else" on same line as closing brace** размещает ключевое слово `else`, на одной строке с закрывающийся фигурной скобкой;
- флаг **Indent code within braces** устанавливает отступ от фигурной скобки;
- флаг **Indent "case" within "switch" statement** - выравнивание оператора `case` в логической цепочке `switch`;
- флаг **Close braces, brackets, and parentheses** - автоматическое добавление к открывающей скобке закрывающую скобку.

## Панель Editor Setting

Панель **Editor Setting** (Настройка редактора) включает в себя большое количество опций для *настройки свойств текстового редактора*. На этой панели происходит настройка контекстного меню, шрифтов текста, местоположение окон и ряд других специфических свойств. Панель **Editor Setting** разделена на три статические области: **Remember** (Напоминания), **Contextual Menus** (Контекстные меню) и **Other Settings** (Другие установки), где расположен соответствующий набор элементов управления. Панель **Editor Setting** изображена на рис. 2.27.

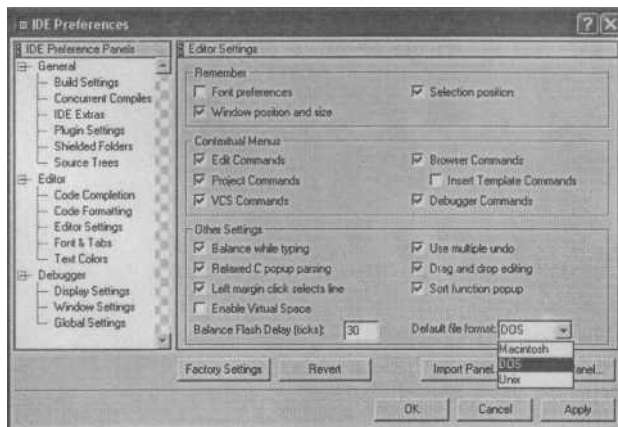


Рис. 2.27. Панель Editor Setting

В статической области **Remember** находятся три флажка:

- **Font preferences** - сохранение установленных параметров шрифта для каждого нового открытого текстового файла;
- **O Selection position** - выбор позиции курсора для ввода текста;
- **L) Window position and size** - выбор позиции окна.

Статическая область **Contextual Menus**, служит для настройки доступных опций в появляющихся контекстных меню при работе с текстовым редактором Metrowerks CodeWarrior. Область **Contextual Menus** содержит следующие элементы управления:

- флаг **Edit Commands** - выбор этого флага добавляет в контекстное меню команду редакции (Edit);
- флаг **Project Commands** добавляет в контекстное меню команду **Проект** (Project);
- флаг **VCS Commands** добавляет в контекстное меню команду контроля версии системы (Version Control System);
- флаг **Browser Commands** добавляет в контекстное меню команду **Обзор** (Browser);
- флаг **Insert Template Commands** позволяет добавить в контекстное меню шаблон вставки;
- флаг **Debugger Commands** добавляет в контекстное меню команду Отладка (Debug).

И последняя статическая область, **Other Settings**, на панели **Editor Setting**, необходима для настройки дополнительных опций в текстовом редакторе, с помощью следующих элементов управления:

- флаг **Balance while typing** - выбор синтаксиса текста при работе с оператором while;
- флаг **Relaxed C popup parsing** - подключает C подобный синтаксический анализатор;
- флаг **Left margin click selects line** - выделение всей строки текста по щелчку левой кнопки мыши напротив выбранной строки с левой стороны поля текстового редактора;
- флаг **Enable Virtual Space** - автоматическое добавление пробела в текст после операции вставки;
- флаг **Use multiple undo** - использование отмены произведенной операции;
- флаг **Drag and drop editing** - подключение в текстовый редактор функции drag-and-drop;
- флаг **Sort function popup** производит сортировку функций согласно алфавиту;
- текстовое поле **Balance Flash Delay** - установка задержки в 1/60 секунды для подсветки редактируемых символов;
- список **Default file format** - формат сохранения файлов.

### **Панель Font and Tabs**

Панель **Font & Tabs** (Шрифт и табуляция) служит для *установки свойств шрифта и позиций табуляции* в текстовом редакторе. На рис. 2.28 изображена панель **Font & Tabs**.

Панель **Font & Tabs** поделена на две статические области. В верхней области **Font settings** (Установки шрифта) присутствуют три списка: **Font (Шрифт)**, **Size** (Размер) и **Script** (Скрипт). Выбрав необходимые значения в этих списках, вы зададите соответствующие свойства в текстовом редакторе для отображения шрифта.

В области **Tab settings** (Установки табуляции) панели **Font & Tabs**, при

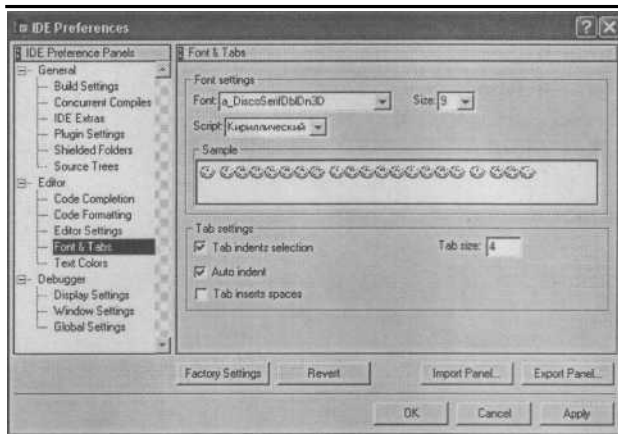
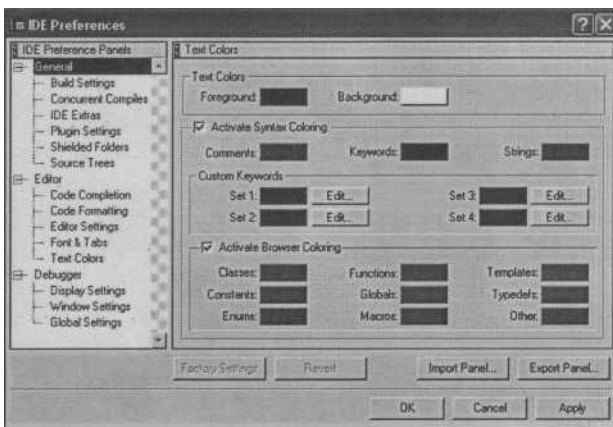


Рис. 2.28. Панель Font & Tabs

- Tab indents selection - выравнивание строки текста после нажатия кнопки Tab на клавиатуре;
- Auto Indent - автоматическое применение отступов от предыдущей строки по нажатию кнопки Enter на клавиатуре;
- Tab Inserts Spaces - пустое пространство вставленное в текст по нажатию кнопки Tab;
- Tab Size определяет размер пустого пространства.

### Панель Text Color

Исходя из названия панели Text Color (Цвет текста) следует, что на этой панели располагаются элементы управления для настройки цвета текста согласно его синтаксической принадлежности. Здесь вы сможете установить подсветку синтаксиса исходного кода для классов, функций, констант, макросов, типов, переменных, комментариев. На рис. 2.29 показана панель Text Color.



Работать с панелью Text Color достаточно просто. Напротив названия каждой опции находится кнопка, окрашенная в определенный цвет. Этот цвет на данный момент является приоритетным для опции. Для изменения цвета нажмите на кнопку, откроется редактор Color (Цвет)

Рис. 2.29. Панель Text Color

с цветовой схемой для выбора цвета. Указав необходимые настройки цвета в редакторе **Color**, нажмите кнопку ОК и таким образом вы переназначите цвет текста для одной из опций, что позволит вам контролировать подсветку синтаксиса в текстовом редакторе среды Metrowerks CodeWarrior.

### 2.3.3. Группа **Debugger**

Третья и последняя группа **Debugger** (Отладчик) в диалоговом окне **IDE Preferences**, настраивает отладчик Metrowerks CodeWarrior для работы над проектом. В группе **Debugger** содержатся три пункта, при выборе которых открываются панели для настройки различных опций. Рассмотрим панели группы **Debugger**.

#### **Панель Display Settings**

На панели **Display Settings** (Установки представления), изображенной на рис. 2.30, задаются параметры для установки опций отображения цвета, сортировки функций и десятичных значений, необходимых в отладке проекта.

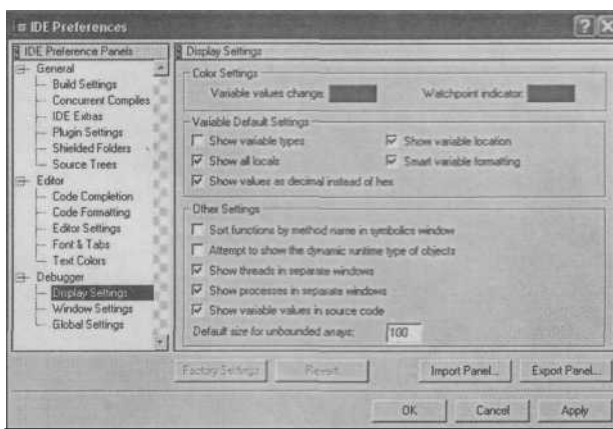


Рис. 2.30. Панель Display Settings

Панель **Display Settings** состоит из трех областей, между которыми логически распределены элементы управления. В области **Color Settings** (Установки цвета), находятся две опции: **Variable values change** (Изменяемое значение переменной) и **Watchpoint indicator** (Индикатор точки разрыва). Как и в случае с панелью **Text Color**, редакция двух этих опций осуществляется на основе назначения цвета. Нажав на кнопку возле названия одной из опций, вы откроете редактор **Color** для установки необходимого цвета. Но обычно для этих опций используется красный цвет.

Следующая область на панели **Display Settings** носит название **Variable Default Settings** (Установки значений по умолчанию). В этой области находится пять флагов для *установки представления переменных*:

- Show variable types** - показать тип переменной;
- Show all locals** - показать все локальные переменные;
- Show values as decimal instead of hex** — показывать только в десятичном значении;
- Show variable location** - показать расположение переменной;
- Smart variable formatting** - форматирование.

Последняя область **Other Settings** (Другие установки) содержит ряд опций для настройки различных параметров представления отладчика:

- Флаг **Sort functions by method name in symbolics window** - сортировать функции по имени;
- Флаг **Attempt to show the dynamic runtime type of objects** - показать dynamic runtime для объектов;
- Флаг **Show threads in separate windows** - показать потоки;
- Флаг **Show process in separate windows** - показать процессы;
- Флаг **Show variable values in source code** - показать значение переменной в исходном коде;
- Текстовое поле **Default size for unbounded arrays** - заданный размер для массивов.

### **Панель Window Settings**

Панель **Window Settings** (Настройки окна) дает возможность сформировать *удобное появление окон* или панелей отладчика в процессе отладки программы. На панели **Window Settings** находятся четыре переключателя и один флаг. Рассмотрим эти элементы управления:

- Do nothing** - ничего не делать;
- Hide non-debugging Windows** - скрывает окна, несвязанные с отладкой;
- Minimize non-debugging windows** - свернуть все окна, несвязанные с отладкой;
- Close non-debugging Windows** - закрыть все окна, несвязанные с отладкой;
- Флаг Do nothing to project windows** - не делать ничего с окнами проекта.

### **Панель Global Settings**

На панели **Global Settings** (Глобальные параметры настройки), изображенной на рис. 2.31, находятся опции для настройки общих параметров. Например, кэширования файлов, автоматического запуска приложений и библиотек, выхода из отладчика.

Панель **Global Settings** окна **IDE Preferences** разделена на две статические области. Область **Cache Edited Files Between Debug Sessions** (Редактирование файлов между сеансами отладки) содержит одноименный флаг, выбор которого делает активным текстовое поле **Maintain files in cache** (Поддержка файлов в КЭШе). В этом поле необходимо указать количество дней поддержки файлов в КЭШе, доступных для отладки/Также в этой области имеется кнопка **Purge cache** (Отчистка Кеша) для отчистки КЭШа отладчика.

Область **Other Settings** (Другие установки) включает в себя шесть флагов для настройки дополнительных параметров отладки:

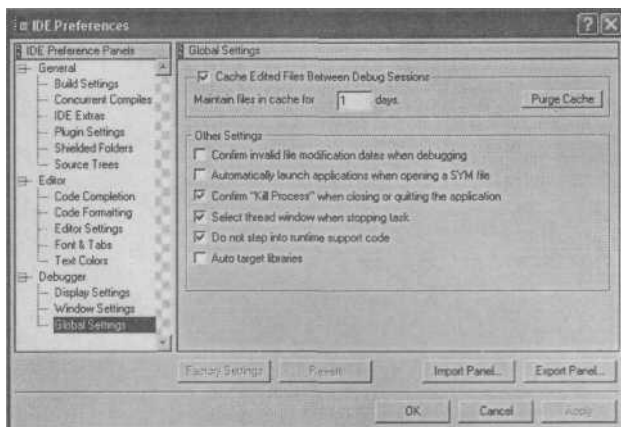


Рис. 2.31. Панель Global Settings

- **Confirm invalid file modification dates when debugging** - подтверждает не допустимость модификации данных при отладке;
- **Automatically launch applications when SYM file opened** - автоматически запускает приложение при открытом SYM файле;
- LJ Confirm "Kill Process" when closing or quitting** — подтверждение для уничтожения процесса при закрытии проекта;
- **Select stack crawl window when task is stopped** - выбрать стек при остановке программы;
- **Don't step into runtime support code** - не выполнять пошаговую отладку в runtime коде;
- **Auto Target Libraries** - автоматический выбор библиотек.

## 2.4. Создание проекта

В стандартной поставке Metrowerks CodeWarrior for Symbian OS Personal 2.8 нет инструментальных средств разработчика (SDK), необходимых для создания программ под операционную систему Symbian. Изучать существующие SDK от различных производителей мы будем в следующей главе, но при создании проекта в среде Metrowerks CodeWarrior, пакеты SDK необходимы, поэтому при создании проекта предположим, что на компьютере уже установлены нужные нам средства. А уже в *главе 4* рассмотрим подробно как процесс инсталляции SDK, так и функциональные возможности этого программного обеспечения. В среде Metrowerks CodeWarrior for Symbian OS Personal 2.8 можно создать пустой и сформированный на основе шаблона проект, задействовав необходимый пакет SDK. Создание пустого проекта не должно вызывать у вас затруднений, поэтому сосредоточим внимание на формировании шаблонного проекта.

Откройте Metrowerks CodeWarrior, выбрав команду в меню **ПУСК** => **Metrowerks CodeWarrior** => **CodeWarrior for Symbian Personal v2.8.3** => **CodeWarrior**

**IDE.** После открытия Metrowerks CodeWarrior воспользуйтесь командой из линей- ' ки меню **File => New (Ctrl+Shift+N)**, откроется диалоговое окно New изображенное на рис. 2.32.

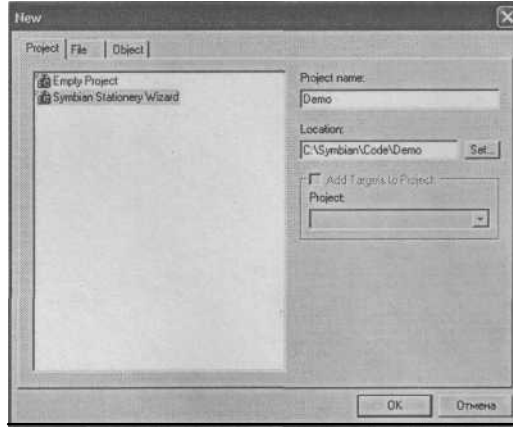


Рис. 2.32. Диалоговое окно New

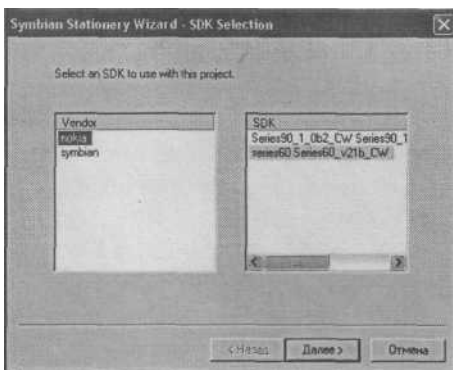
Диалоговое окно New разделено на две части. С левой стороны находятся два пункта: **Empty Project** (Пустой проект) - для создания пустого проекта и **Symbian Stationery Wizard** (Стационарный шаблон для Symbian) - для организации проекта на основании шаблона. Выделите необходимый пункт левой кнопкой мыши. Мы создаем шаблонный проект, поэтому воспользуемся пунктом **Symbian Stationery Wizard**. С правой стороны диалогового окна New, расположены три текстовых поля: **Project Name** (Имя проекта), **Location** (Директория) и **Project** (Проект). В поле **Project Name** задайте имя проекту, в поле **Location** укажите директорию, где будут находиться файлы создаваемого проекта. В этом случае важно помнить, что директория расположения проекта должна обязательно находиться на диске, где установлены Metrowerks CodeWarrior и SDK, обычно это диск C и поэтому все сформированные проекты должны находиться на этом диске, иначе вас ожидает масса различного рода ошибок во время компиляции. Текстовое поле **Project** неактивно в момент создания шаблонного проекта и работает тогда, когда создается пустой проект или в проект добавляется файл. Задав имя проекту, например, Demo, укажите директорию расположения рабочего каталога проекта и нажмите кнопку OK в диалоговом окне New.

Откроется новое диалоговое окно **Symbian Stationery Wizard — SDK Selection**, изображенное на рис. 2.33. Окно состоит из двух текстовых областей: **Vendor** (Производитель) и **SDK**. В области **Vendor** будет находиться список компаний, чьи SDK установлены на компьютере. При выборе производителя, щелкнув левой кнопкой мыши на соответствующем названии, в области SDK отразится полный перечень установленных SDK данного производителя. Создавая проект Demo, воспользуемся SDK компании Nokia и выберем пакет серии 60. Для



этого нажмем в диалоговом окне **Symbian Stationary Wizard - SDK Selection** кнопку Далее.

В новом появившемся окне **Symbian Stationary Wizard - Stationary Selection** будет приведен перечень доступных шаблонов. В зависимости от производителя, количество шаблонов может быть различным, но как минимум имеется пустой шаблон и шаблон типа Hello World. Выберите Hello World и нажмите кнопку ОК (для создания структуры проекта).



После чего в окне **Workspace** появятся две папки **Source** (Исходные коды) и **Libraries** (Библиотеки). Раскрыть обе

папки можно, щелкнув левой кнопкой мыши на пиктограмме квадрата с плюсом возле названия папок. В папке **Source** находятся все файлы проекта, а в папке **Libraries** перечислены подключенные в проект библиотеки. В окне **Workspace**, напротив названия исходных файлов в папке **Source**, проставлены галочки красного цвета - это говорит о том, что ни один файл проекта еще не был откомпилирован. По мере работы над проектом, в момент изменения исходного кода в любом файле проекта, автоматически будет появляться красная галочка у названия файла в окне **Workspace**, давая тем самым сигнал о необходимости компиляции этого файла проекта.

В свою очередь в рабочем каталоге проекта будет сформирован ряд папок, где и располагаются файлы всего проекта. В этом демонстрационном примере, каталог для проекта находится на диске C:\Symbian\Code\Demo. Проследовав по данному адресу, вы обнаружите следующие папки проекта:

- Demo\_Data - выходные данные проекта;
- group - проектные данные и файлы ресурсов;
- inc - содержат заголовочные файлы;
- sis - для файлов PKG и SIS;
- src - исходные коды проекта.

Также в папке Demo находятся дополнительные файлы, созданные средой программирования Metrowerks CodeWarrior, которые необходимы для компиляции проекта. На этом этап создания проекта заканчивается и можно приступать к редакции или написанию исходного кода программы. Для открытия файла проекта нажмите два раза левой кнопкой мыши на названии файла в окне Workspace, и в текстовом редакторе откроется содержимое выбранного файла.

## 2.5. Импорт проекта

Кроме создания проекта в Metrowerks CodeWarrior есть возможность *импорта проектов*, созданных другими средствами программирования и имеющих расшире-

Рис. 2.33. Диалоговое окно Symbian Stationary Wizard - SDK Selection

ние \*.mmp (Make project). Все примеры к книге, находящиеся на компакт-диске, необходимо импортировать в Metrowerks Code Warrior именно таким образом.

Для импорта проекта в Metrowerks CodeWarrior выберите команду в меню **File => Import Project From .mmp File**. Появится диалоговое окно **Symbian Importer - SDK Selection**, изображенное на рис. 2.34. В этом окне, как и в случае создания пустого проекта, необходимо выбрать производителя и SDK, после чего нажать кнопку ОК.

После этого появится новое диалоговое окно **Symbian Importer - MMP File Selection**, изображенное на рис. 2.35. Это окно имеет два текстовых поля: **MMP File Selection** (Выбор файла \*.mmp) и **Platform Selection** (Выбор платформы). В поле **MMP File Selection** необходимо указать путь к файлу проекта MMP, который будет импортирован в Metrowerks CodeWarrior, проще всего это сделать при помощи кнопки **Browse**, расположенной возле поля **MMP File Selection**. Проектный файл MMP почти всегда находится в папке \group и имеет название самого проекта, например, HelloWorld.mmp. Во втором поле **Platform Selection** нужно указать платформу, для которой создается приложение. Платформы обозначаются заглавными прописными буквами и перечислены в диалоговом окне **Symbian Importer — MMP File Selection**. Платформа предопределяет вид создаваемого дистрибутива, рассчитанного для работы на эмуляторе или телефоне. Существуют всего пять вариантов выбора платформ:

- **WINS32** - для эмулятора телефона в Windows необходимо обязательно включать эту платформу дабы иметь возможность тестировать программы на эмуляторе телефона;
- **ARM4** - 32-битный набор бинарных команд, предназначенный для работы на ARM процессорах. Если приложение откомпилировано для этой платформы, то работа программы доступна только для ARM4 и ARM1. Сформированная программа для ARM4 работает быстрее, чем для THUMB, но создается больший по размеру дистрибутив, то есть использует больше ROM памяти;

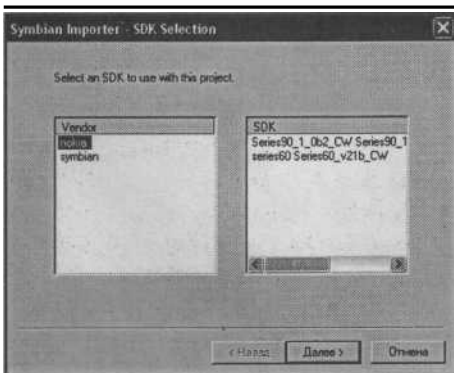


Рис. 2.34. Диалоговое окно Symbian Importer - SDK Selection

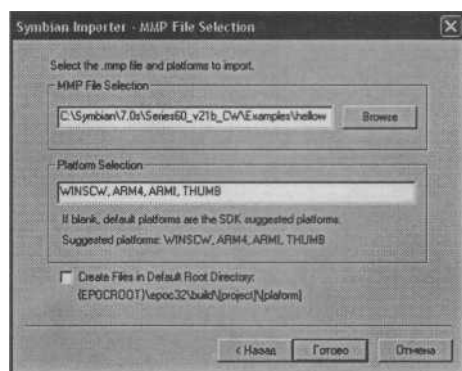


Рис. 2.35. Диалоговое окно Symbian Importer - MMP File Selection

- **ARM1** - 32-битный набор бинарных команд, предназначенный для работы на ARM процессорах. Если приложение откомпилировано для этой платформы, то работа программы доступна для ARM4, ARM1 и THUMB.

Сформированная программа для ARM4 работает быстрее, чем для THUMB, но создается больший по размеру дистрибутив, то есть использует больше ROM памяти. Эта платформа задана по умолчанию;

- **THUMB** - 16-битный набор бинарных команд, предназначенный для работы на ARM процессорах. Если приложение откомпилировано для этой

платформы, то работа программы доступна только для THUMB и ARM1.

Сформированная программа для THUMB работает медленнее, чем для ARM1, но использует меньше ROM памяти;

- **Build All** - включает все четыре вышеперечисленные платформы.

В поле **Platform Selection** пропишите имя платформы, под которую планируется создание программы или перечислите через запятую нужное количество платформ. Платформа WINSCW необходима для эмуляции программы на телефоне, платформа THUMB используется в основном в телефонах Sony Ericsson и платформа ARM1 используется во всех остальных телефонах, так что можно смело прописать все три платформы. Также в диалоговом окне **Symbian Importer -MMP File Selection** имеется флаг **Create Files in Default ROOT Directory**, для задания директории проекта по умолчанию. После задания директории нажмите кнопку ОК, и Metrowerks Code Warrior импортирует выбранный проект.

## 2.6. Компиляция проекта

*Компиляция* текущего проекта на самом деле самая простая и интересная из операций, которая может принести вам массу положительных эмоций или большое количество отрицательной энергии, выплеснутой на бедную клавиатуру или заплыванный монитор. Сама по себе компиляция проекта происходит на машинном уровне, и повлиять на процесс ее работы нельзя, а вот допустить ошибки при написании исходного кода программы можно всегда. При ошибках в программе, во время компиляции, над текстовым редактором открывается дополнительное окно **Errors** с перечислением всех ошибок проекта. Щелкнув на названии ошибки в этом окне, в текстовом редакторе появится красная стрелка, указывающая на строку с ошибкой. На рис. 2.36 показано окно **Errors**.

Перед компиляцией проекта нужно установить платформу, для которой создается конечный продукт. Это может быть WINSCW, ARM4, ARM1, THUMB, то есть если вы планируете в дальнейшем тестировать программу на компьютере при помощи эмулятора, то выбрать надо WINSCW. Если вы создаете готовый продукт с последующей инсталляцией его на телефон, выберите платформы ARM4, ARM1, THUMB в зависимости от производителя телефона. Для всех четырех платформ доступны два варианта: отладочная версия UDEB (например, WINSW UDEB) и окончательная версия программы UREL (например, WINSW UREL). Установка платформы производится командой меню **Project => Set Default Target** или с помощью списка перечисления платформ на панели инструментов окна **Workspace**.

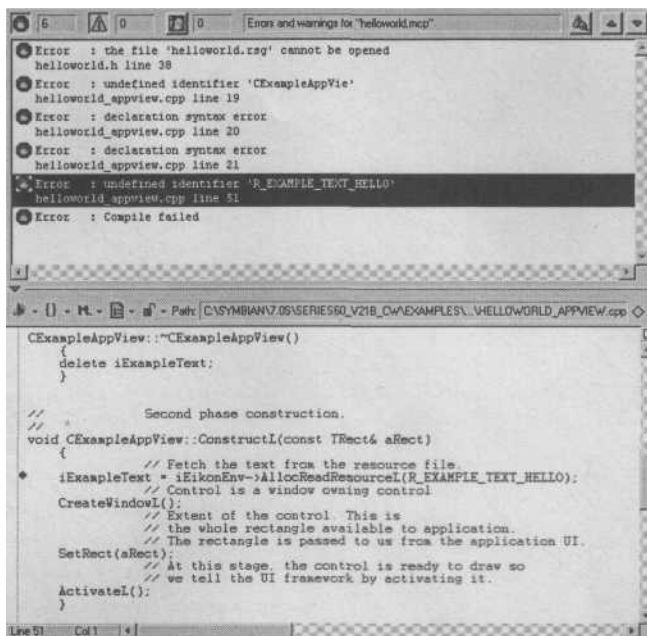


Рис. 2.36. Окно Errors и текстовый редактор

Чтобы откомпилировать рабочий проект для платформы WINSCW, выберите в меню команду **Project => Compile** (Ctrl+F7), также можно воспользоваться и контекстным меню. Для этого в окне **Workspace** нажмите правой кнопкой мыши на названии папки **Source** или отдельно взятого файла исходного кода и в появившемся контекстном меню выберите команду **Compile**. После этого запустится процесс компиляции файлов проекта и в окне **Workspace** напротив названия файлов, по мере компиляции проекта красные галочки будут сменяться на крутящиеся колеса и затем исчезать вовсе.

При изменении исходного кода в одном файле необязательно перекомпилировать весь проект - достаточно откомпилировать только измененный файл. Далее необходимо собрать проект, используя команду **Project => Make** (F7) или нажать на кнопку **Make** на панели инструментов среды Metrowerks CodeWarrior. Кнопка **Make** также дополнительно расположена в окне **Workspace**. После этих действий произойдет процесс сборки проекта, но цикл компиляции проекта можно пропустить и сразу выполнить команду **Make**. В этом случае, если проект не был откомпилирован, произойдет его одновременная компиляция и сборка.

Для того чтобы протестировать программу на эмуляторе телефона нажмите на кнопку **Run**, находящуюся на панели инструментов окна **Workspace** или Metrowerks CodeWarrior. Эта опция также доступна из меню по команде **Project => Run** (Ctrl+F5). Выполнив эту команду, появится эмулятор телефона, на экране которого будут представлены иконки доступных приложений. Тестируемая программа всегда находится последней в списке.

## 2.7. Создание установочного пакета

После того как вы многократно проверили свою программу на эмуляторе и откомпилировали проект для платформ ARMI UREL или THUMB UREL, а также уверены на сто и более процентов в стабильной и надежной работе программы, тогда можно приступить к этапу по созданию установочного пакета программы. *Установочный пакет* — это заархивированные в специфический архив компоненты всей программы с расширением SIS (Symbian Installation System). В таком виде распространяются программы для Symbian OS. В Metrowerks Code-Warrior упаковка программы происходит одновременно с процессом компиляции под платформы ARM4, ARMI и THUMB, но при этом необходимо произвести ряд обязательных настроек в среде Metrowerks CodeWarrior.

В окне **Workspace** в списке с перечислением платформ выберите **ARMI UREL** - если вы делаете программу для телефонов, работающих с этой платформой, а почти все устройства работают с ARMI, то это настройки по умолчанию. Так же выбор платформы доступен через команду меню **Project => Set Default Target**. В том же окне **Workspace** щелкните правой кнопкой мыши на названии папки с исходными кодами **Source**, и в появившемся контекстном меню выберите команду **Add Files** (Добавить файл). Откроется диалоговое окно **Select files to add** (Выбрать добавляемый файл), перейдите в папку с проектом и выберите файл с расширением PKG. После этого нажмите кнопку ОК. Файл PKG - это описательный проектный файл, необходимый для создания SIS архива, он находится в папке проекта \sis или в папке \group, где и лучше всего его располагать. Следующее диалоговое окно **Add Files** (Добавить файл), изображенное на рис. 2.37, содержит перечень платформ, с которыми будет ассоциироваться добавленный в проект файл.

В диалоговом окне **Add Files** снимите флажки напротив всех платформ, оставив только флажок для платформы ARMI UREL (как это показано на рис. 2.37), и нажмите кнопку ОК. В проект будет добавлен файл PKG, на основании которого создается установочный пакет SIS. Файл

PKG имеет ряд атрибутов, которые будут рассмотрены в *главе 7*. Поэтому рекомендую вам на данном этапе воздержаться от создания SIS-файла, отложив формирование установочного пакета до *главы 7*.

Далее выберите в меню команду **Edit** =\* **ARMI UREL Setting** (Alt+F7) или нажмите на инструментальной панели Metrowerks CodeWarrior крайнюю справа кнопку **ARMI UREL Setting**. Откроется диалоговое окно с одноименным названием, изображенное на рис. 2.38.

Диалоговое окно **ARMI UREL Setting** разделено на две области. С левой стороны в *Рис. 2.37*. Диалоговое окно области **Target Setting Panels** располагает-

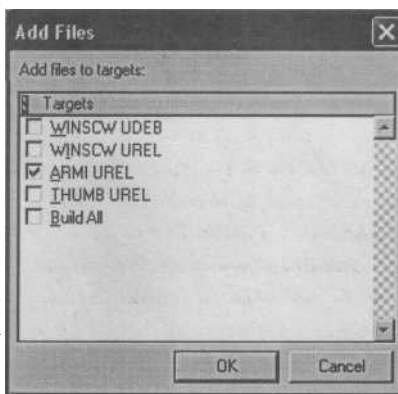


Рис. 2.37. Диалоговое окно Add Files

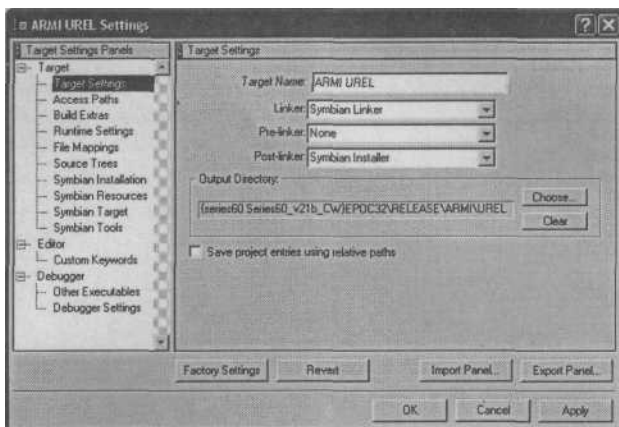


Рис. 2.38. Диалоговое окно **ARM UREL Setting**

ся перечисление панелей, выполненное в виде древовидной структуры. Щелчок левой кнопки мыши на названии панели в области Target Setting Panels, откроет выбранную панель во второй области диалогового окна.

Изначально при открытии диалогового окна ARM UREL Setting доступна панель Target Setting, представленная на рис. 2.38. На этой панели в списке Post-Linker выберите опцию Symbian Installer и нажмите кнопку Apply (Применить). Затем перейдите на панель File Mappings, щелкнув левой кнопкой мыши на названии этой панели в области Target Setting Panels. На рис. 2.39 изображена панель File Mappings. В списке Edit Language необходимо выбрать None. В основной области панели File Mappings, выполненной в виде списка, найдите строку Text \*.pkg, выделите ее курсором мыши и нажмите кнопку Flags. Появится кон-

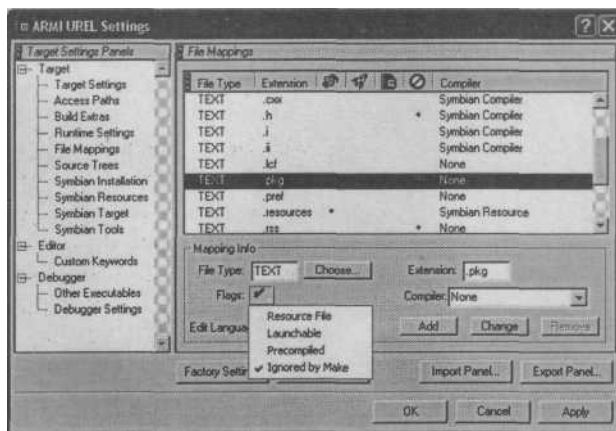


Рис. 2.39. Панель **File Mappings**

текстное меню, где необходимо поставить флаг напротив названия **Ignored by make** (Игнорировать при компиляции и компоновке) и нажать кнопку **Apply**. Теперь добавленный проектный файл PKG будет игнорироваться при компиляции. Любые другие заданные опции на панели **File Mappings** приведут к ошибке, и SIS архив создан не будет!

Последний этап по настройке опций - это переход на панель **Symbian Installation** в диалоговом окне **ARMIUREL Setting**. На этой панели в поле **Output File Name** задайте имя создаваемому установочному пакету и нажмите кнопки **Apply** и **OK**. После этого можно переходить к компиляции проекта для платформы ARMI UREL. Если вы решите создать установочный пакет для программы Demo, созданной в этой главе в качестве демонстрационного примера, у вас могут возникнуть ошибки при компиляции, из-за неправильной компоновки файла PKG. Поэтому сначала изучите главу 7. В следующей главе будет рассматриваться среда программирования C++ BuilderX Mobile Studio.

## Глава 3. IDE C++ BuilderX Mobile Studio

Компания Borland - одна из ведущих корпораций рынка программного обеспечения, имеющая в своем арсенале огромное количество разнообразных продуктов. В компании хорошо развита система распространения программного обеспечения через Интернет в виде trial-версий, которые работают ограниченное количество времени и предоставляют пользователю возможность ознакомиться с инструментальными средствами. На сайте компании <http://www.borland.com> можно найти перечень доступных программ, а так же среду программирования C++ BuilderX Mobile Studio, предназначенную для создания программ под Symbian OS.

После обязательной регистрации на сайте компании Borland, у вас появится возможность загрузить инсталляционный пакет C++ BuilderX Mobile Studio (600 Мб) непосредственно с сайта. На адрес вашей электронной почты придет письмо с файлом активизации продукта, без которого запустить C++ BuilderX Mobile Studio не удастся, поэтому сохраните полученное письмо как указано в инструкции.

Большой объем установочного пакета Mobile Studio отчасти связан с тем, что в него входят сразу две среды программирования:

- JBuilderX Mobile Edition;
- C++ BuilderX 1.5 Mobile Edition.

Среда программирования JBuilderX Mobile Edition предназначена для создания Java-приложений под Symbian OS и телефонов, работающих на основе прошивки. Это мощная среда с большим спектром функциональных возможностей, приятным интерфейсом и множеством дополнительных утилит.

Среда программирования C++ BuilderX 1.5 Mobile Edition служит для создания программ на C++ под Symbian OS. Компания Borland имеет и более раннюю версию C++ BuilderX 1.0, которая фактически ничем не отличается от версии 1.5. В версии C++ BuilderX 1.5 содержатся дополнительно несколько инструментальных пакетов разработчика (SDK), которые описываются в *главе 4* и находятся на компакт-диске, прилагаемом к книге. Если у вас есть ранняя версия C++ BuilderX 1.0, то работайте с ней, обновление вам ничего не даст. Необходимые SDK, записанные на компакт-диске, легко интегрируются в первую версию C++ BuilderX. Обе среды программирования для языков Java и C++ распространяются и по отдельности, но уже в коробочных версиях. Цена получается не маленькая, но они того стоят. В этой главе будут рассмотрены обе версии C++ BuilderX как одно целое.

Теперь перейдем к установке C++ BuilderX Mobile Studio. Убедитесь, что у вас на компьютере установлена библиотека Java 2 Runtime, необходимая для работы C++ BuilderX Mobile Studio.



## 3.1. Установка IDE C++ BuilderX Mobile Studio

1. После скачивания с сайта компании trial-версии C++ BuilderX Mobile Studio, у вас на компьютере появится новый ZIP-архив под названием Mobile\_Stu-

**dio trial 2.5.** Распакуйте его в любое удобное для вас место. Затем зайдите в этот

каталог и найдите файл под названием Launcher\_windows, двойной клик левой кнопки мыши на названии файла запустит программу установки C++ BuilderX Mobile Studio. В первом появившемся диалоговом окне **Borland Mobile Studio Install**, изображенном на рис. 3.1, будет предложено на выбор три варианта инсталляции. Вариант **Complete Trial Install** произведет полную установку программных продуктов компании Borland (как для Java программирования, так и для C++). Два последующих варианта установки Mobile Studio дают возможность выбрать между средами программирования C++ BuilderX Mobile Edition Trial и J BuilderX Mobile Edition Trial. Мы будем придерживаться варианта **Complete Trial Install**, поэтому выберите эту опцию в диалоговом окне **Borland Mobile Studio Install**, и нажмите на кнопку **Complete Trial Install**.

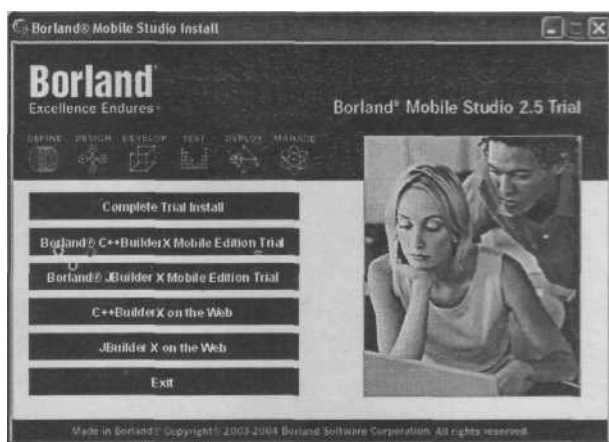


Рис. 3.1. Диалоговое окно Borland Mobile Studio Install

2. В следующем диалоговом окне **Complete Trial Install**, изображенном на

рис. 3.2, будут перечислены выбранные элементы для установки. Не снимайте

флажки напротив перечисленных элементов, все они необходимы для работы с Borland Mobile Studio, а просто нажмите кнопку **Install** для продолжения инсталляции.

3. Далее появится диалоговое окно с приветствием: **C++ BuilderX 1.5 Mobile Edition Trial**.

После нажатия кнопки Next откроется новое диалоговое окно с

лицензионным соглашением. Ознакомившись с лицензией, на основе которой распространяется данный вид продукта, нажмите кнопку Next. В появившемся окне

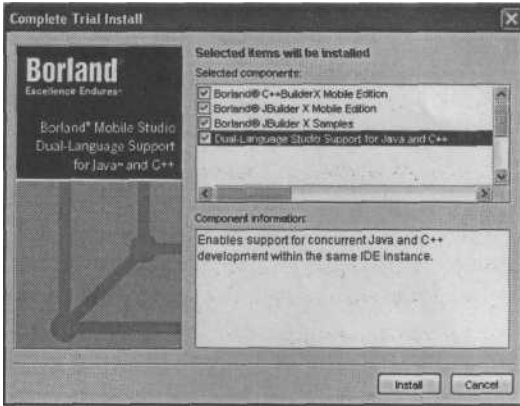


Рис. 3.2. Диалоговое окно Complete Trial Install

**Choose Install Set**, изображенном на рис. 3.3, предлагаются для выбора компоненты к установке, выберите все элементы и нажмите кнопку **Next**.

4. Следующее диалоговое окно **Choose Install Folder** служит для выбора директории инсталляции C++ BuilderX 1.5 Mobile Edition -это корневой каталог и именно туда необходимо устанавливать среду программирования. Все программные продукты, связанные, так или иначе, с Symbian OS, устанавливайте в корневой каталог диска C, то есть все программы SDK, сре-

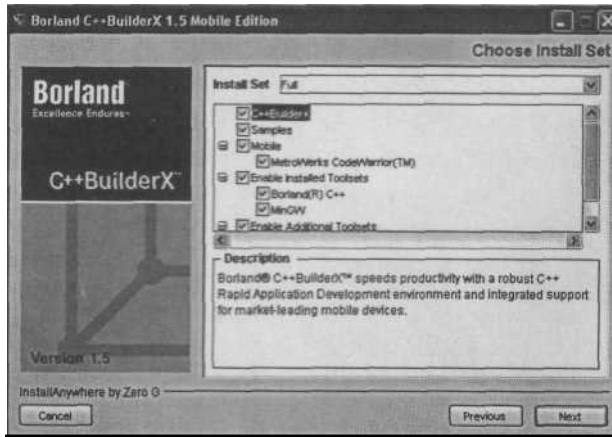


Рис. 3.3. Диалоговое окно Choose Install Set

ды программирования и различные дополнительные утилиты устанавливаются на один диск в корневой каталог. Нажав кнопку **Next**, вы запустите процесс установки. По окончании инсталляции C++ BuilderX 1.5 Mobile Edition процесс установки автоматически перейдет к инсталляции JBuilderX Mobile Edition Trial.

5. При установке JBuilderX Mobile Edition Trial в нескольких диалоговых окнах будет показана различная информация для определения настроек по установке. В итоге появится диалоговое окно **Pre-Installation Summary**, изображенное на рис. 3.4, с суммирующей информацией для начала запуска установки JBuilderX Mobile Edition Trial. В последствии при инсталляции JBuilderX Mobile Edition Trial будут появляться еще несколько диалоговых окон, уведомляющих

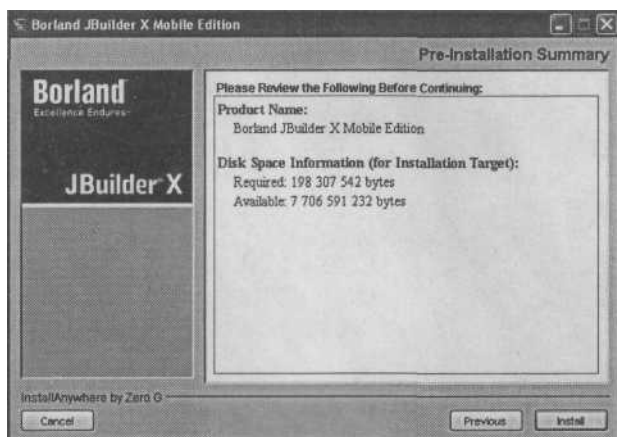


Рис. 3.4. Диалоговое окно Pre-Installation Summary

о нормальном прохождении хода установки, все они достаточно просты. По окончании инсталляции Mobile Studio у вас на рабочем столе появятся два ярлыка для C++ BuilderX и JBuilderX.

При первом запуске C++ BuilderX или JBuilderX необходимо ввести серийный код и указать путь к файлу активации, после чего можно спокойно наслаждаться работой с системой. Поскольку нас интересует C++ BuilderX, то далее в главе подробно изучается именно это продукт.

## 3.2. Изучаем C++ BuilderX

Среда программирования C++ BuilderX — это очень хороший инструмент для создания полноценных мобильных приложений под Symbian OS. Приятный интерфейс, хорошо разработанная система справки, прекрасный текстовый редактор, средства отладки и компиляции - все эти достоинства C++ BuilderX дают разработчику отличный рабочий инструмент!

На рис. 3.5 изображена среда программирования C++ BuilderX с открытым проектом Demo. В этой главе проект Demo создан с помощью C++ BuilderX, и если внимательно изучить программный код проекта Demo, сделанного во второй главе при помощи Metrowerks Code Warrior, то обнаружится, что он совершенно идентичен. Это обыкновенные шаблоны, которые создаются автоматически и реализуют минимальное GUI-приложение для Symbian OS, о котором вы подробно узнаете из главы 7.

Чтобы *открыть* C++ BuilderX воспользуйтесь иконкой на рабочем столе, созданной во время инсталляции, или выполните команду в меню ПУСК => **Все программы => Borland C++BuilderX => C++ BuilderX**.

Рабочее пространство среды программирования C++ BuilderX гармонично разделено на панели, открыть или закрыть которые можно в меню команды **View**. Основная и самая большая часть рабочего пространства C++ BuilderX отведена

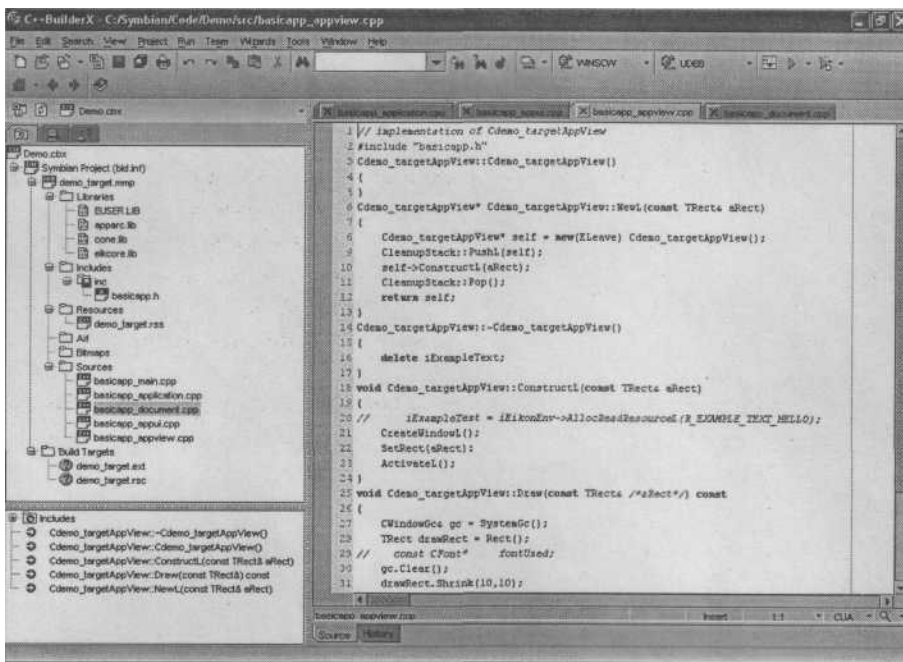


Рис. 3.5. Среда программирования C++ BuilderX

под текстовый редактор и панель **Project** (Проект). Текстовый редактор среды C++ BuilderX имеет встроенную интеллектуальную систему подсветки синтаксиса и возможность индивидуальной настройки под семантику различных языков программирования. Панель **Project** в C++ BuilderX выполняет функции окна **Workspace** как в Metrowerks CodeWarrior и Visual Studio, и имеет абсолютно аналогичные возможности в представлении структуры данных проекта.

Линейка меню C++ BuilderX содержит команды: **File** (Файл), **Edit** (Редактировать), **Search** (Поиск), **View** (Вид), **Project** (Проект), **Run** (Запустить), **Team** (Группа), **Wizards** (Мастера), **Tool** (Инструменты), **Window** (Окно) и **Help** (Помощь). Это вполне стандартный набор команд, используемый в различных инструментальных средствах. Рассмотрим подробно линейку меню C++ BuilderX.

### 3.2.1. Меню File

Меню **File**, изображенное на рис. 3.6, имеет в своем арсенале двадцать две команды, которые необходимы для создания проекта, открытия и сохранения файлов, печати и выхода из системы:

- **New** (Ctrl-N) - открывает диалоговое окно для создания нового проекта;
- **New File** - создает новый файл;
- **Open Project** - открывает проект;
- **Open File** (Ctrl-O) - открывает файл;

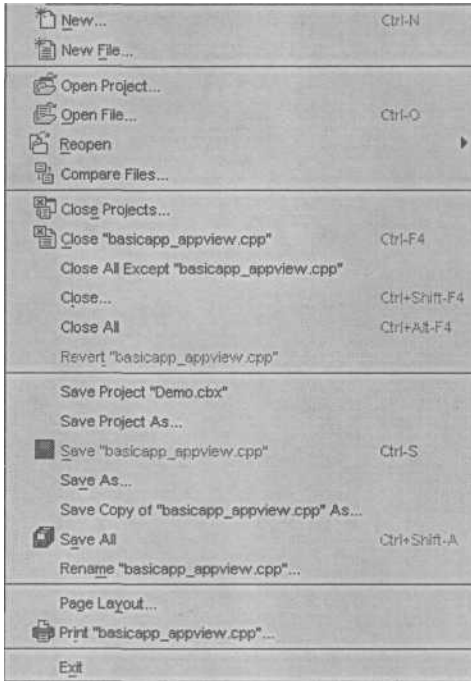


Рис. 3.6. Команды меню File

- **Print** — печать;
- **Exit** - выход.

### 3.2.2. Меню Edit

Меню **Edit**, изображенное на рис. 3.7, содержит небольшой набор стандартных команд для редакции исходного кода в текстовом редакторе. Аналогичные команды доступны и в текстовом редакторе через контекстное меню, которое появляется после нажатия правой кнопки мыши в свободной области редактора.

- **Undo** (Ctrl-Z) - шаг назад;
- **Redo** (Ctrl+Shift-Z) - шаг вперед;
- **Cut** (Ctrl-X) - вырезать;
- **Copy** (Ctrl-C) - копировать в буфер обмена;
- **Paste** (Ctrl-V) - вставить из буфера обмена;
- **Delete** — удалить;
- **Format All** - форматировать исходный код согласно установленным опциям для текстового редактора;
- **Select All** (Ctrl-A) - выделить все;
- **Sync Edit** (Ctrl+Shift-J) - синхронизировать редакцию.

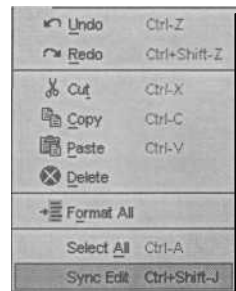


Рис. 3.7. Меню Edit

- **Reopen** - повторное открытие закрытых, но уже использовавшихся файлов или проектов;
- **Compare Files** - сравнение файлов;
- **Close Project** - закрывает проект;
- **Close File** (Ctrl-F4) - закрывает активный файл;
- **Close All Except** — закрыть все файлы кроме...;
- **Close** (Ctrl+Shift-F4) - закрыть;
- **Close All** (Ctrl+Alt-F4) - закрыть все;
- **Q Revert** - вернуть;
- **Save Project** - сохранить проект;
- **Save Project As** - сохранить проект как;
- **Save** (Ctrl-S) - сохранить;
- **Save As** - сохранить как;
- **O Save Copy** - сохранить копию;
- **Save All** (Ctrl+Shift-A) - сохранить все;
- **Rename** - переименовать;
- **Page Layout** - предварительный просмотр;

### 3.2.3. Меню Search

*Меню Search* предназначено для осуществления различных видов поиска в проекте, например, для поиска текста в файле или файл исходного кода. В меню **Search** имеются следующие команды:

- Find** (Ctrl-F) - найти;
- Find in Patch** (Ctrl-P) - найти путь;
- Replace** (Ctrl-R) - повторить поиск;
- Replace in Patch** - повторить поиск пути;
- O Search Again** (F3) - возобновить поиск снова;
- Incremental Search** (Ctrl-E) - возрастающий поиск в файле;
- Go to Line** (Ctrl-G) - перейти к строке;
- Go to Address** (Ctrl+Shift-G) - перейти к адресу;
- Find Classes** (Ctrl-Minus) - найти класс по названию.

### 3.2.4. Меню View

В *меню View* находятся команды, с помощью которых можно настроить внешний вид среды программирования C++ BuilderX, существуют такие команды:

**G Toolbars** - эта команда имеет вложенное меню с перечислением инструментальных панелей для настройки внешнего вида C++ BuilderX:

- File** - инструментальная панель **Файл**;
- Editing** — инструментальная панель **Редактировать**;
- Search** - инструментальная панель **Поиск**;
- Build** — инструментальная панель компиляции, компоновки и отладки приложений;
- Platform** - инструментальная панель для выбора платформы;
- Configuration** — инструментальная панель **Конфигурация**;
- Run/Debug** - инструментальная панель запуска и отладки программ;
- Navigation** - инструментальная панель **Навигации**;
- Help** — инструментальная панель **Помощь**;
- Show All** - показать все панели;
- Hide All** - убрать все панели.
- Project** (Ctrl+Alt-P) - проектная панель;
- Content** (Ctrl+Alt-C) - содержание проекта;
- Structure** (Ctrl+Alt-S) - структура проекта;
- Messages** (Ctrl+Alt-M) - панель сообщений
- Status Bar** - строка состояния;
- Switch Viewer to History** — выбор из предыдущих видов;
- Context Menu** (Ctrl-F10) - открыть контекстное меню;
- Remove All Message Tabs** - удалить все сообщения;
- Hide All** (Ctrl+Alt-Z) - сброс всех настроек;
- History** - история;
- Back** (Ctrl+Alt-Left) - возврат;

- **Forward** (Ctrl+Alt-Right) - вперед;
- **Breakpoints** (Ctrl+Alt-B) - точка останова;
- **CPU View** (Ctrl+Alt-U) - показать низкоуровневую информацию для отладки;
- **Debug Event Log** - показать события при отладке.

### 3.2.5. Меню *Project*

*Меню Project* содержит команды для компиляции и сборки проекта, а также набор команд для работы над проектными данными в целом:

- **Make Project** (Ctrl-F9) - компилирует и компоует проект;
- **Rebuild Project** - перекомпиляция и компоновка проекта;
- **Q Link Project** - линковка проекта;
- **Make** (Ctrl+Shift-F9) - компиляция и компоновка;
- **Q Rebuild** - перекомпиляция и компоновка;
- **Make Project Group** - компилирует и компоует групповой проект;
- **Rebuild Project Group** - перекомпиляция и компоновка группового проекта;
- **Link Project Group** — линковка группового проекта;
- **Add Files** - добавить файл;
- **New Folders** - новая папка;
- **New Directory View** - новая директория;
- **Remove from Project** - удалить из проекта;
- **Refresh** — свойства;
- **Rename** - переименовать;
- **Project Properties** - свойства проекта;
- **Default Project Properties** - установить свойства проекта по умолчанию;
- **Project Group Properties** - свойства группового проекта.

### 3.2.5. Меню *Run*

В *меню Run* включены команды для запуска эмулятора, отладки, конфигурации проекта и установки точек останова, рассмотрим команды меню **Run**:

- **Run Project** (F9) - запустить проект;
- **Debug Project** (Shift-F9) - отладка проекта;
- **Configurations** — конфигурация;
- **Run to Cursor** (F4) - перейти к курсору;
- **Pause Program** - пауза;
- **Resume Program** - резюме;
- **Reset Program** (Ctrl-F2) - сбросить программу;
- **Inspect** (Alt-F5) — инспектор;
- **Add Breakpoint** - добавить точку останова, есть три варианта:
- **Add Line Breakpoint** - добавить точку останова в строчку;
- **Add Data Breakpoint** - добавить точку останова в данные;
- **G Add Address Breakpoint** - добавить точку останова по адресу.

### 3.2.6. Меню *Team*

Меню *Team* содержит различные команды, связанные с проектными данными:

- Select Project VCS** - выбрать проект VCS;
- Configure Version Control** - конфигурация версии;
- Update** - обновления;
- LJCommit** - совершить;
- File Status** - статус файла;
- Add** - добавить;
- CJ Remove** - удалить;
- Update Project** - обновить проект;
- Commit Browser** - передать браузеру;
- Place Project into CVS** - проект CVS;
- Create Local Repository** - создать локальный архив;
- Sync Project Settings** - синхронизация проектных настроек;
- CVS Administration** - администрирование.

### 3.2.7. Меню *Wizards*

В этом меню всего три команды, включающие процесс настройки опций с помощью мастера:

- New Build Configuration** — новая конфигурация для отладки;
- Target Settings** - цель для установки;
- Build Options** - опции для отладки.

### 3.2.8. Меню *Tools*

Меню инструментальных средств содержит команды:

- IDE Options** — опции среды программирования C++ **BuilderX**;
- Editor Options** - опции текстового редактора;
- Configure File Associations** — конфигурация файлов ассоциации;
- Build Tools** - инструменты для отладки;
- Exported Settings Group** - настройка экспортной группы;
- Symbian SDK Configuration** - конфигурация SDK;
- Rest GDB** - сбросить GDB;
- Reload Toolset** - переставить инструментальные средства;
- Configure Tools** - конфигурации инструментов.

### 3.2.9. Меню *Window*

Команды меню *Window* связаны с настройкой окон в C++ BuilderX:

- New Browser** - новое окно браузера;
- Minimize Browsers** - минимизировать окна;
- Restore Browsers** - восстановить окна;
- Cascade Browsers** - установить окна каскадом;



- **Tile Browsers Vertically** - установить окна по вертикали;
- **Tile Browsers Horizontally** - установить окна по горизонтали;
- **Select Browser** - выбрать браузер;
- **Select Message** - выбор сообщения.

### 3.2.10. Меню Help

Меню помощи включает в себя команды вызова контекстной справки в C++ BuilderX:

- **Help Topics** - помощь;
- **C++ BuilderX Home Page** - домашняя страница компании Borland;
- **Release Notes** - информация о релизе C++ BuilderX;
- **About C++ BuilderX** - о программе C++ BuilderX.

### 3.2.11. Панель инструментов

*Панель инструментов* среды программирования C++ BuilderX, изображенная на рис. 3.8, содержит кнопки быстрого доступа. Все кнопки панели инструментов дублируют команды меню и упрощают процесс работы над проектом. Рассмотрим кнопки быстрого доступа среды программирования C++ BuilderX:

- Q **New** - новый файл;
- **Open File** - открыть файл;
- **Reopen** - открыть вновь;
- **Close** - закрыть;
- **Save** - сохранить;
- O **Save All** - сохранить все;
- **Print** - печать;
- **Undo** - шаг назад;
- O **Redo** - шаг вперед;
- **Copy** - копировать;
- **Paste** - вставить;
- **Cut** - вырезать;
- **Find** - найти;
- **Search Again** - найти вновь;
- **Replace** - переместить;
- **Find Classes** - найти классы;
- **Make Project** - компилировать и компоновать проект;
- **Выбор платформы** - выбор платформы при компиляции проекта;
- **Выбор версии** - выбор версии при компиляции проекта;
- D **Messages** - сообщение;



Рис. 3.8. Панель инструментов C++ BuilderX

- **Run Project** - запустить проект;
- **Debug Project** - отладка проекта;
- **Go To** — перейти к;
- **Back** - возврат;
- **Forward** - вперед;
- **Help Topics** - помощь.

### 3.2.12. Панель Project

Панель **Project** (Проект) среды программирования C++ BuilderX, изображенная на рис. 3.9, необходима для представления структуры проекта. На панели **Project** расположены три кнопки быстрого доступа: **Close Project** (Закреть активный проект), **Refresh** (Свойства проекта) и **Select and Open Project** (Выбор и открытие проекта), которые дублируют команды меню **Project**.

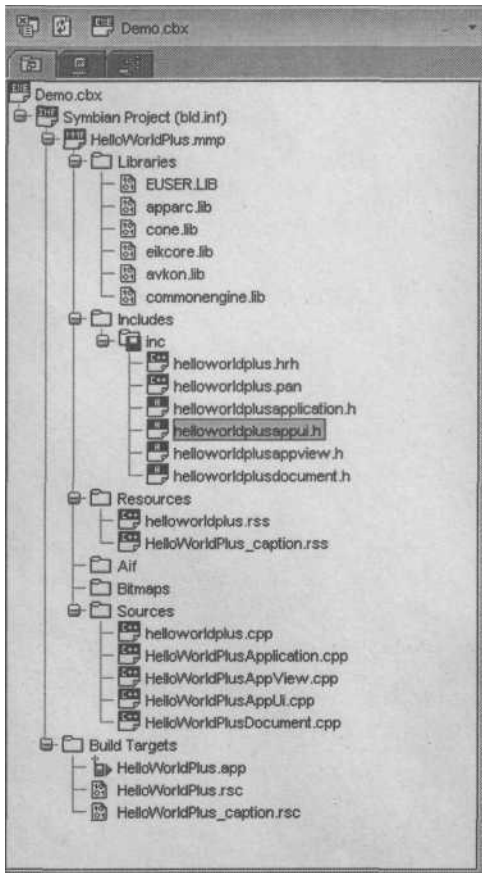


Рис. 3.9. Панель Project

Панель **Project** состоит из трех вкладок: **Project Content** (Структура проекта), **File Browser** (Обзор файлов) и **Class Browser** (Обзор классов). Вкладка **Project Content** отображает полную структуру проекта, включая файлы ресурсов, MMP- и AIF-файлы (что очень удобно). На вкладке **File Browser** представлена файловая система вашего компьютера, вкладка **Class Browser** служит для обзора подключенных в проект классов. По умолчанию обзор классов отключен, чтобы включить эту опцию нажмите правой кнопкой мыши на пункте **Class browsing disabled for active project** во вкладке **Class Browser**. Появится контекстное меню, в нем выберите команду **Properties** (Свойства), откроется диалоговое окно **Project Properties**, изображенное на рис. 3.10. Поставьте флаг напротив названия **Enable C++ Class Browsing** (как показано на рис. 3.10) и нажмите кнопку **OK**. На вкладке **Class Browser** появятся перечисления системных классов подключенного SDK. Двойной щелчок левой кнопкой мыши на названии класса приведет к открытию выбранного за-

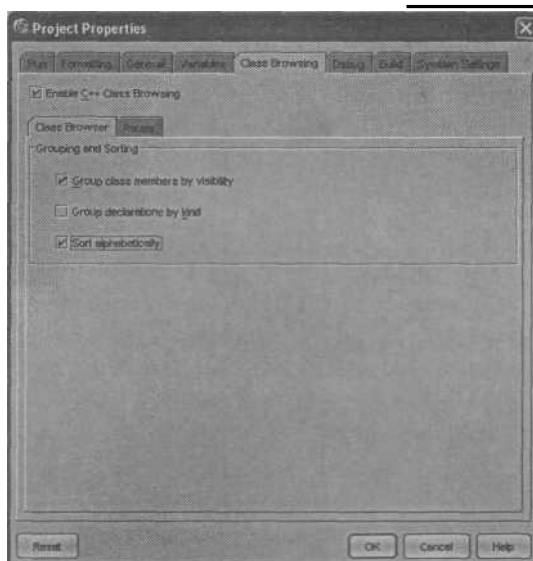


Рис. 3.10. Диалоговое окно Project Properties

торе. Будьте осторожны и не занимайтесь редактированием системных классов написанных не одним десятком программистов.

### 3.3. Подключение SDK

В комплект поставки C++ BuilderX 1.5 Mobile Edition входят SDK серии 60 и 90, UIQ2.1. Чтобы подключить дополнительные инструментальные средства разработчика в C++ BuilderX 1.5 Mobile Edition или в C++ BuilderX 1.0, выполните команду меню **Tools => Symbian SDK Configuration**. Откроется диалоговое окно **SDK Configuration** (рис. 3.11). В области **Current SDK Configuration** (Текущие SDK-конфигурации) будет перечислен список подключенных в C++ BuilderX инструментальных средств разработчика.

В диалоговом окне **SDK Configuration** вы найдете пять небольших по размеру кнопок. Кнопка **Add** (Добавить) добавляет SDK в C++ BuilderX, кнопка **Remove** (Удалить) удаляет выбранный SDK, кнопка **Edit** (Редактировать) дает возможность редакции подключенных SDK. Кнопка **Close** (Заккрыть) закрывает диалоговое окно **SDK Configuration** и кнопка **Help** (Помощь) вызывает контекстную справку C++ BuilderX.

Для добавления SDK нажмите кнопку **Add** в диалоговом окне **SDK Configuration**. Откроется новое окно **Add SDK Configuration**, изображенное на рис. 3.12.

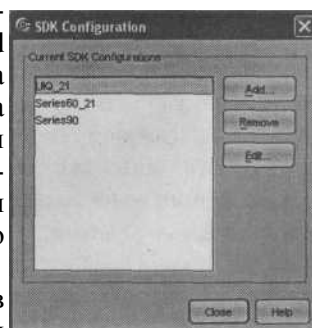


Рис. 3.11. Диалоговое окно SDK Configuration

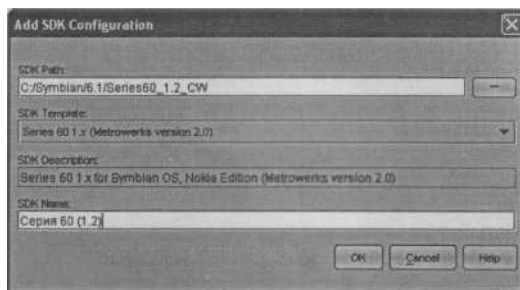


Рис. 3.12. Диалоговое окно  
Add SDK Configuration

В текстовом поле **SDK Path** (Путь к SDK) диалогового окна **Add SDK Configuration** укажите путь к папке, в которой установлено подключаемое SDK. В списке **SDK Template** (Шаблоны SDK) выберите версию SDK, после чего она автоматически отобразится в поле **SDK Description**. В поле **SDK Name** (Имя SDK) задайте любое удобное для вас имя подключаемому пакету инструментальных средств и нажмите кнопку **OK**. SDK добавятся в C++ BuilderX.

### 3.4. Создание проекта

Для создания нового проекта в среде программирования C++ BuilderX выполните команду меню **File => New**. Откроется диалоговое окно **Object Gallery**, изображенное на рис. 3.13. В этом окне на вкладке **Mobile C++** показаны возможные варианты создания новых проектов, выполненные в виде пиктограмм.

- **Import Symbian C++ Project** - произвести импорт готового проекта в C++ BuilderX;



Рис. 3.13. Диалоговое окно Object Gallery  
вкладка Mobile C++

- **New Symbian GUI Application** - создать новый проект с шаблонным GUI-приложением;
- **New Symbian Empty Project** - сформировать новый пустой проект;
- **New Symbian DLL** - создать динамически подключаемую библиотеку **DLL**;
- **New Symbian C++ File** - образовать новый файл исходного кода с расширением \*.cpp;
- **New Symbian .H File** - создать новый заголовочный файл с расширением \*.h;
- **New Symbian Resource File** - сформировать файл ресурса;
- **New Symbian AIF Wizard** - запускает мастер создания или добавления AIF-ресурсов в программу.

В качестве демонстрационного примера выберем опцию **New Symbian GUI Application**, создав тем самым шаблонный проект с predetermined классами для GUI- (Графический Интерфейс Пользователя) приложения. Избрав нужный шаблон, нажмите кнопку ОК. Появится диалоговое окно **New Symbian GUI App Wizard - Step 1 of 2**, изображенное на рис. 3.14.



Рис. 3.14. Диалоговое окно New Symbian GUI App Wizard - Step 1 of 2

В диалоговом окне **New Symbian GUI App Wizard - Step 1 of 2** содержатся два текстовых поля и три списка. В поле **Project Name** (Имя проекта) задается имя проекта, выберем в качестве примера название Demo. В поле **Project Directory** (Проектная директория) необходимо указать путь для сохранения проекта. Это должен быть тот же самый диск, где у вас установлен пакет SDK и лучше, если это будет корневой каталог диска C. В списке SDK выберите версию пакета инструментального средства разработчика, например, серию 60. В списке **Platform** (Платформа) нужно указать платформу, это может быть WINS C W, ARM1 или THUMB. В списке **Build** выберите версию для компиляции и компоновки

приложения: UDEV (Отладочная версия) или UREL (Окончательная версия). Нажмите кнопку **Next** для продолжения настроек. Появится новое диалоговое окно **New Symbian GUI App Wizard - Step 2 of 2**, которое можно увидеть на рис. 3.15.

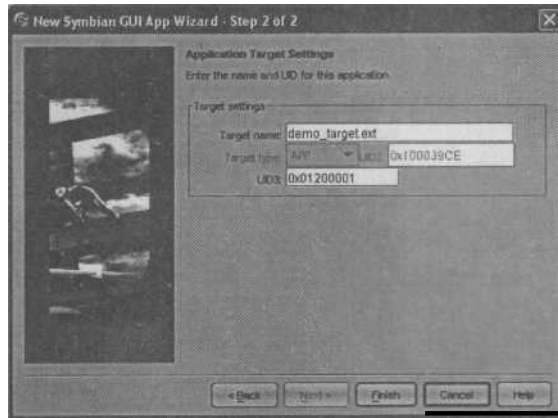


Рис. 3.15. Диалоговое окно New Symbian GUI App Wizard - Step 2 of 2

В диалоговом окне **New Symbian GUI App Wizard - Step 2 of 2** в поле **Target Name** нужно задать целевое имя проекту в формате имя\_цель .ext. В поле UID3 указывается уникальный идентификатор приложения, подробно об идентификаторах вы узнаете из главы 7. Затем нажмите кнопку **Finish**. На панели **Project** во вкладке **Project Content** появится сформированная структура проекта Demo.

## 3.5. Импорт проекта

Чтобы *импортировать проект*, созданный в любой среде программирования, выберите команду **File => New** и в появившемся диалоговом окне **Object Gallery** остановите свой выбор на опции **Import Symbian C++ Project**, а после нажмите кнопку ОК. Откроется диалоговое окно **Import Symbian C++ Project Wizard - Step 1 of 2**, изображенное на рис. 3.16.

В качестве примера для импорта послужит проект Demo, созданный ранее в среде Code Warrior. В окне **Import Symbian C++ Project Wizard - Step 1 of 2** в списке SDK необходимо указать версию пакета инструментального средства разработчика. В текстовом поле **Bld.inf** нужно найти проектный файл bld.inf располагающийся, как правило, в папке \group импортируемого проекта. В списке **Platform** нужно указать платформу, это может быть WINSCW, ARM1 или THUMB. В списке **Build** выберите версию UDEV или UREL. После выбора всех опций в области **Contents of bld.inf file** будут представлены файлы для импорта

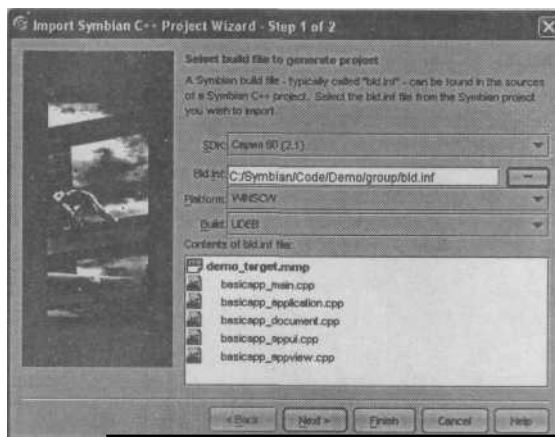


Рис. 3.16. Диалоговое окно Import Symbian C++ Project Wizard - Step 1 of 2

проекта в среду программирования C++ BuilderX, нажмите кнопку **Next**. Появится новое диалоговое окно **Import Symbian C++ Project Wizard - Step 2 of 2**, содержащее одно текстовое поле **Project file name**, в котором нужно задать имя проекту и нажать кнопку **Finish**, выполнив тем самым импорт проекта в C++ BuilderX. Создание пустого проекта происходит по той же схеме, что и создание двух рассмотренных типовых проектов, особых затруднений вызвать у вас это не должно.

## 3.6. Компиляция проекта

Для *компиляции* активного проекта в среде программирования C++ BuilderX выберите команду меню **Project => Make Project** или воспользуйтесь горячими клавишами **Ctrl-F9**. На время компиляции программного кода появится небольшое по размеру информационное окно **Build Progress** с сообщениями о процессе компиляции проекта. По окончании компиляции на панели **Build**, в нижней части рабочего пространства C++ BuilderX можно ознакомиться с полным отчетом процесса компиляции. Если в момент компиляции будут найдены ошибки в проекте, то все они будут отображены на панели **Build**, как показано на рис. 3.17. Двойной щелчок левой кнопкой мыши на названии ошибки на панели **Build**, откроет текстовый редактор и пометит ошибку в исходном коде красной полосой.

Для того чтобы *протестировать* откомпилированную программу на компьютере при помощи эмулятора телефона, поставляемого с SDK, выполните команду меню **Run => Run Project** или используйте горячую клавишу **F9**. На некоторое время появится информационное окно **Build Progress**, сообщающее о процессе сборки проекта, по окончании которого запустится эмулятор телефона. В зависимости от выбранного SDK для создаваемого проекта, запустится соответствующий эмулятор телефона.

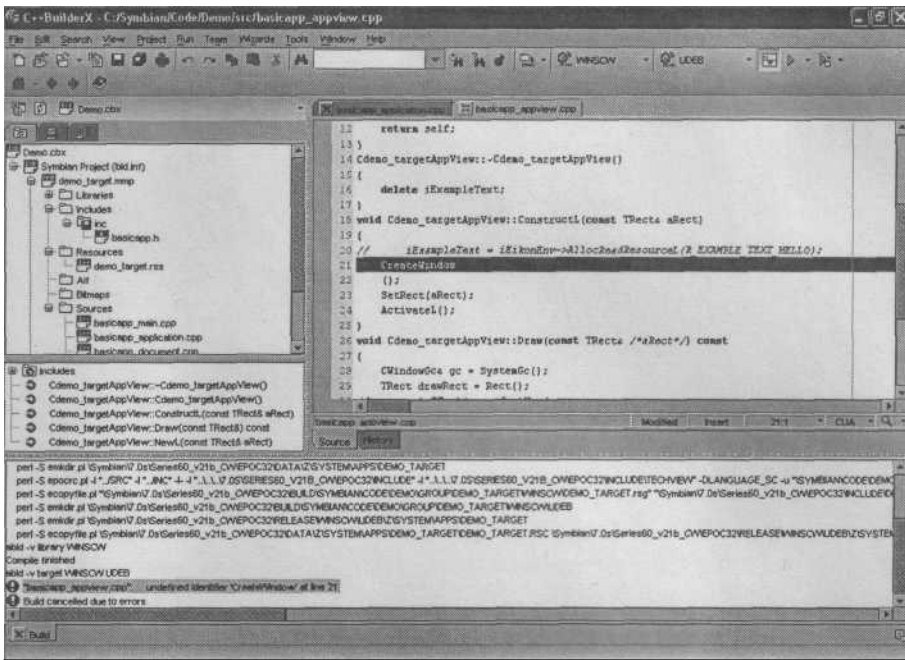


Рис. 3.17. Найденные ошибки в процессе компиляции проекта

### 3.7. Создание установочного пакета

После того как вы убедитесь в стабильной работе приложения и будете готовы перенести программу на телефон, создайте *установочный пакет* SIS. Для этого на панели инструментов кнопкой **WINSCW** (Выбор платформы) измените платформу на ARM1 и кнопкой **UDEB** (Выбор версии) установите версию UREL. Аналогичные действия можно выполнить с помощью окна свойств проекта. Для этого используйте команду меню **Project => Project Properties** и в появившемся окне перейдите на вкладку **Symbian Settings**, где с помощью списков **Platform** и **Build** установите необходимые значения. Выполнить цикл по компиляции проекта можно с помощью команд меню **Project => Make Project**. В рабочем каталоге проекта появится файл с расширением \*.sis, а на панели **Project** среды программирования C++ BuilderX в структуре проекта появится дополнительная надпись **Package File**. Но на этом этапе создание установочного пакета и переноса его на телефон не рекомендуется, изучите сначала главу 7, и только после этого приступайте к созданию пакета SIS в среде программирования C++ BuilderX.



## Глава 4. Инструментальные средства разработчика

Для создания мобильных приложений под Symbian OS необходимы инструментальные средства разработчика (SDK) предоставляемые производителями телефонов. Благодаря издательству ДМК и лично Мовчану Дмитрию Алексеевичу читатели этой книги имеют уникальную возможность воспользоваться компакт-диск к книге, где собраны SDK от крупнейших производителей телефонов. К сожалению SDK имеют большой размер (например, SDK UIQ2.1 «весит» в архиве 206 Мб) и в связи с этим не все инструментальные средства поместились на компакт-диск.

Инструментальные средства разработчика включают в себя системную библиотеку, различные утилиты, документацию, примеры и ряд дополнительных пакетов, например, для работы с камерой или звуком. Компаний, производящих телефоны на базе Symbian OS, много и поэтому существует масса различных SDK. Базовая версия SDK компании Symbian Ltd. содержит пользовательский интерфейс UIQ, созданный дочерней компанией UIQ Technology AB. Пользовательский интерфейс UIQ рассчитан на дисплей телефона большого размера с разрешением как минимум 208 x 320 пикселей.

В свою очередь компания Nokia создала свой пользовательский интерфейс под названием серия 60, где разрешение экрана равно 176 x 208 пикселей. Но при этом Nokia еще имеет серии 80 и 90, рассчитанные на разрешение экрана от 200 x 640 пикселей и более. На этих двух платформах производятся дорогие коммуникаторы, которые пока не имеют массового спроса у потребителей. Таким образом, на данный момент в мире существуют две основные платформы -UIQ и серия 60, представляющие разные интерфейсы пользователя.

На UIQ базируются телефоны компаний Sony Ericsson, Motorola, BenQ, Ari-ma, а на серии 60 телефоны компаний Nokia, Siemens, Sendo X, Panasonic, Samsung, Lenovo. Для корректной работы всех SDK необходима библиотека ActivePerl, которую можно свободно загрузить в Интернете по адресу [www.activeperl.com](http://www.activeperl.com). Так же необходимы: Java Runtime 1.3.1, без которой у вас не будет работать программа SISAR, упаковывающая программу в архив \*.sis для последующей инсталляции на телефон, и AIF Builder, добавляющий в программу иконку. Даже если у вас уже установлена Java Runtime поздней версии, вам все равно понадобится именно Java Runtime 1.3.1 дистрибутив, который очень хитрым образом инсталлирует себя в каталог Program Files в папку JavaSoft и именно к этой папке обращаются программы SISAR и AIF Builder. Специально искать Java Runtime 1.3.1 и ActivePerl необходимости нет, поскольку некоторые SDK имеют в своей поставке

эти компоненты. Поэтому дальше в этой главе в хронологическом порядке будут рассматриваться имеющиеся SDK и в этом порядке рекомендуется их устанавливать на компьютер, старайтесь проследовать всем рекомендациям. После книги «Программирование мобильных телефонов на Java 2 Micro Edition» было получено множество писем с вопросом: почему у меня возникают ошибки при компиляции. А при анализе ситуации выяснилось, что кто-то из читателей не установил Java 2 SDK, кто-то библиотеку Java Runtime, а кто-то посчитал, что рекомендации по установке программных средств в корневой каталог лишены смысла, хотя в книге подробно шаг за шагом был описан процесс инсталляции SDK. Во избежание проблем при компиляции, сборке и упаковке программ строго следуйте перечисленным рекомендациям на протяжении всей книги.

## 4.1. Программные средства компании Sony Ericsson

Компания Sony Ericsson принимает активное участие в развитии Symbian OS и имеет несколько моделей телефонов под управлением этой операционной системы. Телефоны Sony Ericsson имеют пользовательский интерфейс UIQ, где разрешение экрана телефона как минимум 208 x 320 пикселей и сравнимо с дисплеем КПК. Телефоны Sony Ericsson обладают сенсорным дисплеем, двухсторонней

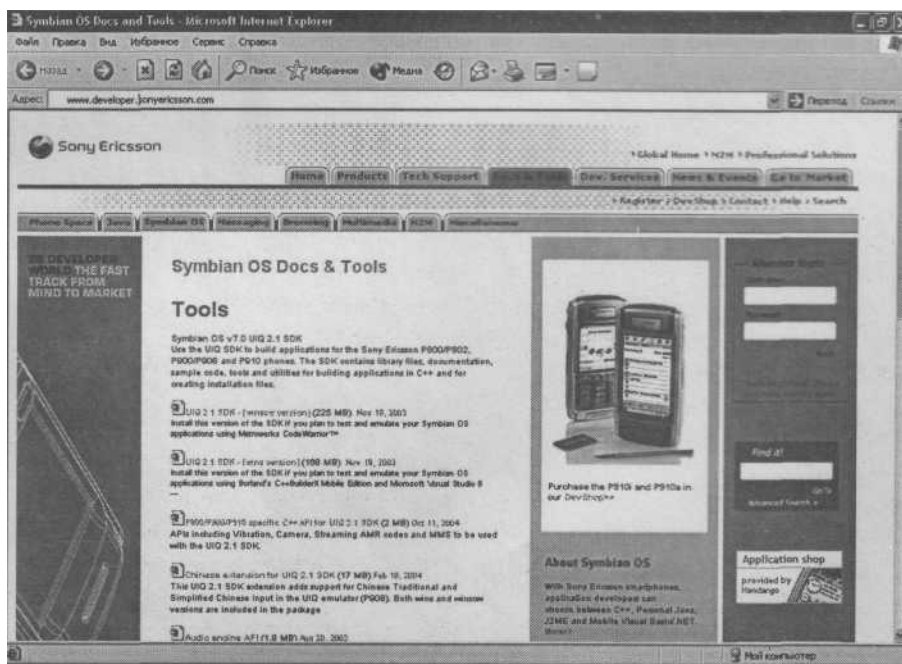


Рис. 4.1. Сайт компании Sony Ericsson

клавиатурой с буквами и цифрами, что, признаться весьма оригинально и удобно в работе. Сайт компании Sony Ericsson содержит большое количество информации, документации, SDK для Symbian OS и платформы Java 2 ME.

Инструментальное средство разработчика компании Sony Ericsson строится на платформе UIQ и является универсальным SDK для всех моделей телефонов, работающих с UIQ. Инструментальное средство разработчика UIQ существует в двух версиях: UIQ 2.0 (для телефонов Sony Ericsson P800) и UIQ 2.1 (для телефона Sony Ericsson P900 и более поздних версий). Платформа UIQ2.0 базируется на версии Symbian OS 6.1 и CLDC 1.0/MIDP 1.0 (для Java 2 ME программ), тогда как UIQ 2.1 строится на версии Symbian OS 7.0 и поддерживает CLDC 1.0/ MIDP 2.0. Несколько запутанная ситуация, но и это еще не все. Инструментальное средство разработчика (SDK) компании Sony Ericsson, основывающееся на платформе UIQ, распространяется двумя различными дистрибутивами, для среды программирования Metrowerks CodeWarrior (UIQ WINSCW) и для C++ BuilderX Mobile Studio (UIQ WINS). Два этих дистрибутива особенно ничем не отличаются, но предназначены для работы с определенной средой программирования, что несколько усложняет работу с программным обеспечением. К слову сказать, точно такая же ситуация складывается и с инструментариями от Nokia Но как вы знаете из главы 3, компания Borland поступила гуманно, и среда программирования C++ BuilderX поддерживает оба вида SDK, поэтому достаточно иметь версию, рассчитанную на работу с Metrowerks CodeWarrior. Эта версия UIQ2.1 как раз и находится на компакт-диске к книге в папке \Sony Ericsson. Там же находится пакет обновлений для UIQ 2.1 и дополнительная библиотека для работы со звуком. На сайте компании по адресу в Интернет: <http://developer.sonyericsson.com> всегда можно найти последнюю информацию об SDK.

### 4.1.1. Установка SDK

Процесс *установки SDK* от компании Sony Ericsson достаточно прост: множество диалоговых окон с вопросами различной сложности приведут к установке SDK на ваш компьютер. Но в процессе инсталляции есть ряд моментов, играющих ключевую роль при установке SDK. Поэтому при инсталляции UIQ сосредоточимся на этих моментах, пропуская диалоговые окна с приветствиями, лицензионным соглашением и другой информацией. Рассмотрим процесс установки UIQ.

1. Как только вы дойдете до диалогового окна **Choose Destination Location**,

изображенного на рис. 4.2, вам будет предложено выбрать директорию для уста

новки SDK. По умолчанию это C:\Symbian\UIQ\_21 (в эту папку и нужно уста навливать SDK). В последствии при формировании своих проектов, создайте папку с любым названием в директории C:\Symbian\UIQ\_21\Epos32\Ваша папка и вы гарантированно избежите массы ошибок при компиляции и упаковке программы.

2. Следующий важный момент при установке SDK - это выбор компонентов.

На рис. 4.3 изображено диалоговое окно **Select Components**. В этом окне необхо-

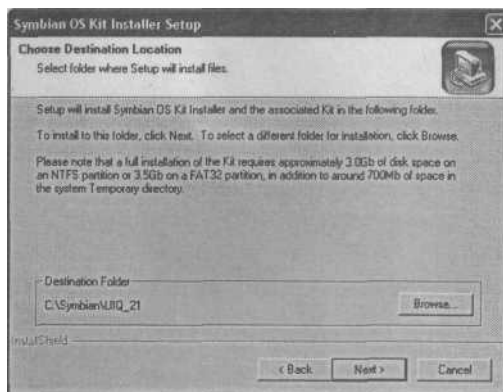


Рис. 4.2. Диалоговое окно Choose Destination Location

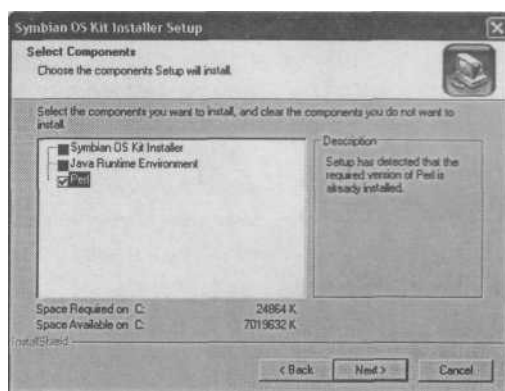


Рис. 4.3. Диалоговое окно Select Components

можно выбрать все элементы для установки, поставив флажки напротив названий **Symbian OS Kit Installer**, **Java Runtime Environment** и **Perl**. Даже если у вас на компьютере установлена Java Runtime, то этот компонент все равно необходим для работы с SDK.

3. При инсталляции Java Runtime в диалоговом окне **Choose Destination Location**, изображенном на рис. 4.4, не изменяйте директорию - именно в папку C:\Program Files\JavaSoft\JRE\1.3.1 должна производиться установка этого компонента, необходимого для работы программ SISAR и AIF Builder.

4. Далее при установке библиотеки Perl проследите за тем, чтобы этот компонент был установлен в корневой каталог на диск C, как показано на рис. 4.5.

Компонент Perl необходим для корректной работы среды программирования C++ BuilderX, а точнее для процесса компиляции и сборки проекта.

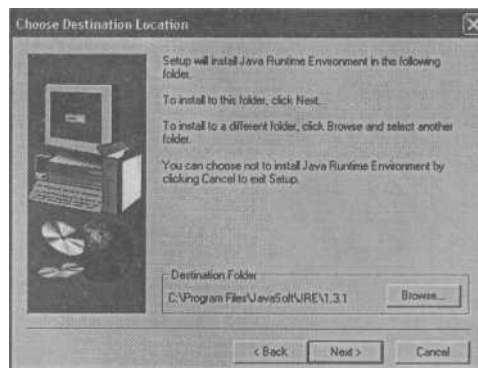


Рис. 4.4. Диалоговое окно **Choose Destination Location** для Java Runtime

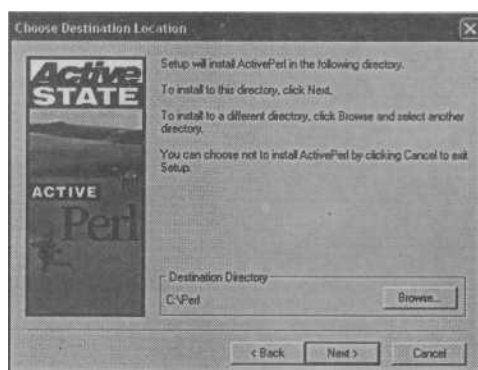


Рис. 4.5. Установка Perl

5. Следующий этап состоит в выборе элементов установки для Symbian OS Kit Installer. Диалоговое окно с одноименным названием изображено на рис. 4.6. Для выбора того или иного элемента необходимо напротив названия установить флажок. К инсталляции предлагаются компоненты:

- UIQ 2.1 Emulator only - устанавливается только эмулятор телефона для платформы UIQ;
- UIQ 2.1 C++ SDK - установка SDK для языка программирования C++;
- UIQ 2.1 Java SDK- установка SDK для языка программирования Java 2 ME;
- UIQ 2.1 MIDP SDK - установка библиотеки MIDP для Java 2 ME программирования;
- Documentation - документация к SDK. Не устанавливайте этот элемент, сняв флажок напротив названия этой опции. На компакт-диске к книге

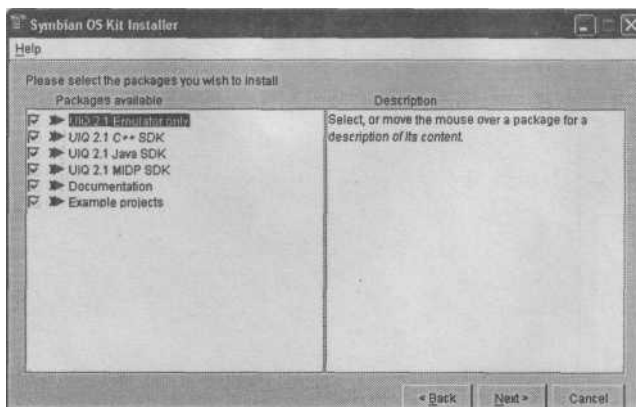


Рис. 4.6. Диалоговое окно Symbian OS Kit Installer

в папке \Sony Ericsson имеется файл, обновляющий документацию и ряд системных библиотек; □ Example project - демонстрационные примеры для платформы UIQ.

6. На этом инсталляция SDK закончена, но нужно обязательно выполнить *обновление установленных компонентов*. В папке \Sony Ericsson на компакт-диске находится ZIP-архив с названием uiq21\_update1\_winscw. Если вы не устанавливали документацию, то просто разархивируйте этот пакет в папку C:\Symbian\UIQ\_21. Если компонент Documentation был инсталлирован, воспользуйтесь средствами операционной системы Windows и через **Установку и удаление программ** удалите Documentation. Только после этого разархивируйте uiq21\_update1\_winscw в папку C:\Symbian\UIQ\_21.

### **4.1.2. Эмуляторы телефонов Sony Ericsson**

В комплекте SDK поставляются эмуляторы телефонов компании Sony Ericsson. Среда программирования Metrowerks CodeWarrior определит SDK автономно. В C++ BuilderX 1.5 Mobile Edition уже входят инструментальные средства разработчика, но если вы устанавливаете дополнительные SDK, то их необходимо подключить самостоятельно, выполнив команду меню **Tools => Symbian SDK Configuration**.

по умолчанию, а для подключения второго эмулятора нужно произвести одну несложную операцию. Зайдите в каталог C:\Symbian\ UIQ\_21\ерос32\data и найдите в этой папке два файла ерос.bmp и ерос.ini. Затем переименуйте оба найденных файла (например, в ероsl.bmp, ероsl.ini) и найдите два других файла под



Рис. 4.7. Эмулятор телефона для платформы UIQ 2.1

названием StandardQ.bmp и StandardQ.ini. Переименуйте эти два файла в ерос.bmp и ерос.ini, произведя тем самым замену одного эмулятора на другой. Более того, в папке C:\Symbian\UIQ\_21\P900 Emulator Bitmap находится эмулятор телефона Sony Ericsson P900. Для подключения этого эмулятора выполните перечисленные выше действия и наслаждайтесь эстетичной картинкой телефона. И еще один бонус. На компакт-диске в папке \Sony Ericsson находится дополнение, где вы найдете эмулятор телефона



## 4.2. Программные средства компании Nokia

Компания Nokia имеет, пожалуй, самое большое количество программного обеспечения. На сайте <http://www.forum.nokia.com> можно найти множество информации о программном обеспечении, созданном компанией Nokia для Symbian OS и платформы Java 2 ME.

Инструментальные пакеты разработчика для Symbian OS делятся на три серии:

- серия 60 - разрешение экрана от 176 x 208 пикселей;
- серия 80 - разрешение экрана от 200 x 640 пикселей;
- серия 90 - разрешение экрана от 320 x 640 пикселей.

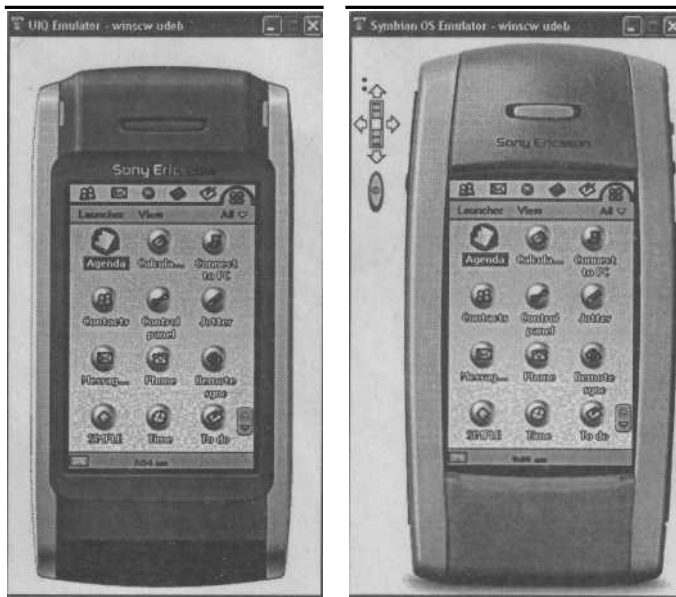


Рис. 4.8. Эмуляторы телефонов Sony Ericsson P900 и Sony Ericsson P800

Forum Nokia - Developer Resources - Microsoft Internet Explorer

www.forum.nokia.com

<b>Forum Nokia</b> Premium Solution Forum Nokia PRO <b>Developer Platforms</b> Series 40 Platform Series 60 Platform Series 80 Platform For CDMA Market <b>Resources</b> Device Specifications Tools and SDK's Documents Technologies Device optimization Porting Learning Path Technical Library Books <b>Services</b> Developer Hub Professional Support Testing Usability Training <b>Communities</b> Discussion Boards Mobile Games Media & Music Enterprise <b>Business Opportunities</b>	<b>Recent Updates</b> Migrating To Symbian OS From Java... This technical case study covers issues faced by a mobile game development team in moving from the Java platform to C++ for Symbian OS. 04-Apr-05	<b>Developer Platform Updates   All Updates</b> <b>Series 60 Platform: Frequently Asked Questions</b> This document provides an excellent introduction to the main aspects of Series 60 Platform from the developer point of view. It covers the platform editions including 3rd Edition, available tools, and guides on finding further information. 29-Mar-05	<b>Newsletter Sign Up</b> Send language Privacy Policy / Archives <b>Spotlight</b> <b>Series 60</b> Developer Platform 3rd Edition Better opportunities, smarter applications, and better production. Series 60 Theme Studio (for Symbian OS) v1.2 <b>Top Downloads</b> 1. Nokia Multimedia Converter 2.0 2. Nokia Developer's Suite 2.2 for J2ME... 3. Series 60 Theme Studio for Symbian OS 4. Series 60 2nd Edition SDK for Symbian OS, Supporting Feature Pack 2, For C++ / Chinese Version 5. Series 60 2nd Edition SDK for Symbian OS, Supporting Feature Pack 2, for MDP <b>Editor's Choice</b> Premium Solution Overview For Developers <b>Key Technologies</b> On Nokia Themes On Bluetooth	
	<b>Series 80 Platform: Frequently Asked Questions</b> This document provides an excellent introduction to the main aspects of Series 80 Platform from the developer point of view. It covers the platform editions including 3rd Edition, available tools, and guides on finding further information. 29-Mar-05	<b>Developing Applications For The Series 80 Developer Platform And The Nokia 7710</b> This document provides detailed information on the most important differences between Series 80 Developer Platform 2.0 and the Nokia 7710 device. Although there are some differences, the Series 80 Developer Platform and the Nokia 7710 device share the same code base and much of the development effort can be utilized to develop applications both for Series 80 and the Nokia 7710. 23-Mar-05	<b>Symbian OS: Threads Programming</b> This document explains the use of threads in Symbian OS. When using threads, synchronization and mutual exclusion must be considered in order to get thread interactions safe. 23-Mar-05	<b>User Experience Checklist For J2ME... Applications</b> This document lists fundamental user experience issues that must be taken into account when designing Java... Applications. 16-Mar-05
	<b>Series 60 Developer Platform 1.02.0: Multiple Threads Example</b> This package includes a Thread example program that demonstrates thread usage and synchronization. A document explaining the use of threads in Symbian OS is included in the package. 16-Mar-05	<b>Introduction To The PM API (With Example) v1.1</b> This updated document is an Introduction to the PM API, including a brief description of the API and the example application that is included in the package. 15-Mar-05	<b>MIDP 2.0: Wireless Messaging API Example</b> This package includes a Wireless Messaging API Example for MIDP 2.0. A document describing the example and its usage is included in the package. 15-Mar-05	<b>Mail download</b>

Серии 80 и 90 используются в дорогих коммуникаторах, и массового распространения этих устройств пока ожидать не приходится. Поэтому основная масса устройств от компании Nokia базируется на серии 60. Например, смартфон Nokia N-Gage QD стоит всего 6,5 тысяч рублей и, учитывая возможность работы на этом смартфоне специализированных мобильных игр с качественной трехмерной графикой, этот вид продукта выдвигается на одно из первых мест в своей категории. Кстати, для создания специальных игр для N-Gage, распространяемых на карте памяти, есть специальный SDK N-Gage. Он распространяется только для лицензированных разработчиков, поэтому на сайте компании и компакт-диске этот SDK вы не найдете. В состав SDK N-Gage входят дополнительные библиотеки, утилиты и масса документации. Дело в том, что смартфоны N-Gage и N-Gage QD базируются на SymbianOS6.1 и множество библиотек из Symbian OS 7.0, улучшающих работу с графикой, недоступны. Это, конечно, не основная причина создания специализированного SDK для смартфона N-Gage, но все же такой SDK существует и доступен он не каждому. А чтобы снять все вопросы по поводу качества трехмерной графики в смартфонах серии 60, советую найти игры Pandemonium (только для N-Gage) или Doom для всех смартфонов серии 60 и просто поиграть в них!

### **4.2.1. Серия 60**

Основной и самой распространенной платформой среди смартфонов является серия 60. На этой платформе базируются смартфоны Nokia, Siemens, Sendo X, Panasonic, Samsung, Lenovo, поэтому, установив SDK этой серии, вы получите возможность создавать программы для телефонов, перечисленных выше. За всю историю существования серии 60 было выпущено несколько версий платформы - это серия 60 версия 0.9 (1.0, 1.2, 2.0, 2.1) и, совсем недавно Nokia анонсировала версию серии 60 2.8, поддерживающую Symbian OS 8.0. На момент написания книги версии 2.8 на сайте компании Nokia еще не было. Из большого количества версий серий 60 самые распространенные — серия 60 версии 1.2 и серия 60 версии 2.1. На компакт-диске к книге находится дистрибутив серии 60 версии 2.1. Серия 60 версия 1.2 предназначена для создания приложений под Symbian OS 6.1, а серия 60 версии 2.1 для Symbian OS 7.0 (7.0s). Серия 60 версии 2.1 имеет два вида установочных пакетов: для среды программирования Metrowerks CodeWarrior (WINSCW) и C++ BuilderX (WINS). Более того, есть SDK и для Microsoft Visual Studio (серия 60 версии 1.2 и 0.9). Тут нас опять спасает компания Borland со своей средой программирования C++ BuilderX, где реализована поддержка обоих видов SDK. Установка SDK серии 60 не должна вызвать у вас затруднений, единственное неперемutable условие - это инсталляция SDK в директорию C:\Symbian\6.1 (или 7.0, 7.0s)\Series60. Это каталог по умолчанию, и не изменяйте пути при установке SDK. В комплекте поставки серии 60 отсутствует Java Runtime Environment и Perl, поэтому вам необходимо самостоятельно установить эти библиотеки до установки SDK серии 60!

При работе с Metrowerks CodeWarrior установленный SDK автоматически идентифицируется средой программирования, а в C++ BuilderX нужно явно



Рис. 4.10. Эмулятор серии 60

подключит SDK, выполнив команду меню **Tools => Symbian SDK Configuration** и в появившемся диалоговом окне добавить установленный SDK (подробности в *главе 3раздел 3.3*). Все рабочие проекты, готовые к упаковке в SIS архив должны располагаться в директории `C:\Symbian\7.0s(6.1)\Series60\Eros32`, это обязательное условие для упаковки программы в SIS архив.

При работе над проектом в одной из сред программирования, после компиляции можно протестировать программу на эмуляторе телефона. В SDK серии 60 имеется эмулятор, изображенный на рис. 4.10, обобщающий все модели телефонов этой серии.

## 4.2.2. Серия 80

Компания Nokia имеет несколько коммуникаторов, работающих на основе серии 80. Это телефоны с разрешением экрана от 200 x 640 пикселей и с полноценной QWERTY клавиатурой, например, Nokia 9210. Описание этой модели вы можете найти в *приложении 2*.

Серия 80 включает в себя несколько версий дистрибутивов, на компакт-диске в папке `\Nokia` находится версия, предназначенная для работы в C++

BuilderX и Visual Studio C++ 6. *Установка SDK* несложна, но есть моменты, на которые необходимо обратить свое внимание.

1. При инсталляции SDK серии 80 в диалоговом окне **Choose Destination Location**, необходимо проследить, чтобы SDK был установлен на диск C в папку `\Symbian`, как это показано на рис. 4.11.

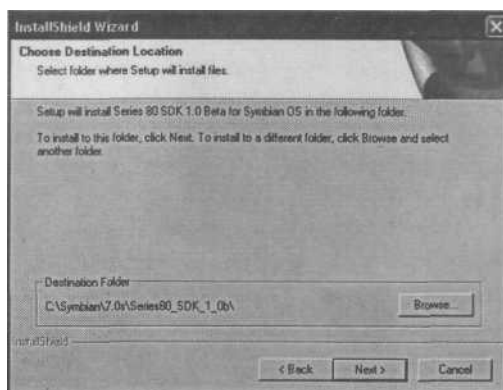


Рис. 4.11. Диалоговое окно Choose Destination Location серии 80

2. При дальнейшей установке SDK выберите комплектацию **Custom** и убедитесь, что в появившемся диалоговом окне **Select Features**, изображенном на рис. 4.12, выбраны все компоненты, и флажки проставлены напротив имеющихся названий.

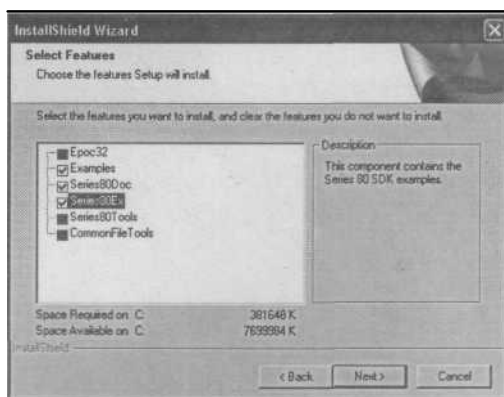


Рис. 4.12. Диалоговое окно Select Features серии 80

3. В конце инсталляции в диалоговом окне **Network Configuration Setup** (рис. 4.13), будет предложено настроить сетевые опции. Если вам не нужна поддержка сети, то нажмите кнопку No, доведя процесс инсталляции SDK серии 80 до логического окончания.

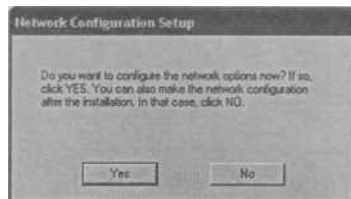


Рис. 4.13. Диалоговое окно Network Configuration Setup

4. И последний момент состоит в конфигурации SDK в среде программирования C++ BuilderX. После установки серии 80, перейдите в файл Series8070s.xml. Затем перейдите в каталог, где установлена среда программирования C++ BuilderX в папку C:\C++ BuilderX\sdktemplate и вставьте скопированный файл из буфера обмена. Далее откройте C++ BuilderX и с помощью команды меню **Tools => Symbian SDK Configurations** подключите SDK в C++ BuilderX. На этапе добавления SDK в диалоговом окне **Add SDK Configurations**, изображенном на рис. 4.14, в списке **SDK Template** выберите название **Series 80 SDK 7.0s (Microsoft version)**.

В дальнейшем работа с C++ BuilderX и SDK серией 80 ни чем не отличается от рассмотренной ранее серии 60. При тестировании программы на эмуляторе серии 80 используйте курсор мыши для нажатия кнопок эмулятора. На рис. 4.15 изображен эмулятор серии 80.

Каталог SDK в папку C:\Symbian\7.0s\Series80 и скопируйте в буфер обмена находящийся там

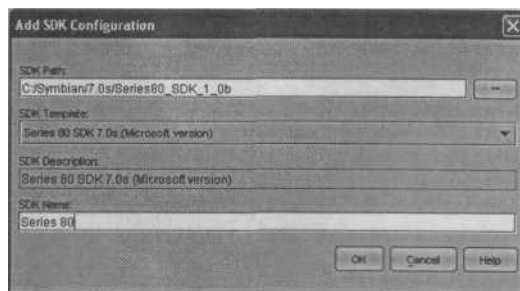


Рис. 4.15. Эмулятор телефона серии 80

### 4.2.3. Серия 90

В телефонах компании Nokia, использующих серию 90, существует гораздо больше моделей в отличие от серии 80, но цена коммуникаторов этой серии достаточно высока (например, телефон Nokia 9500 стоит сейчас 24000 рублей). Телефоны серии 90 имеют полноценную QWERTY клавиатуру и разрешение экрана от 320 x 640 пикселей.

На компакт-диске в папке \Nokia вы найдете SDK серии 90 для среды программирования Metrowerks CodeWarrior. Установка дистрибутива напоминает инсталляцию SDK серии 80. В поставку серии 90 входят Java Runtime Environment и Perl, но если у вас уже установлены эти компоненты, то мастер установки автоматически определит их и не предложит инсталляции. Рассмотрим важные моменты, возникающие в процессе *установки SDK* серии 90.

1. Директория для инсталляции SDK серии 90 определяется в диалоговом окне **Choose Destination Location**, по традиции это папка на диске C:\Symbian (рис. 4.16).



Рис. 4.16. Диалоговое окно Choose Destination Location серии 90

6. В диалоговом окне Select Features, изображенном на рис. 4.17, необходимо проставить флажки напротив всех компонентов, чтобы установить максимально возможное количество элементов SDK. По умолчанию документация и примеры не предлагаются к инсталляции, поэтому проследите за выполнением вышеперечисленных действий.

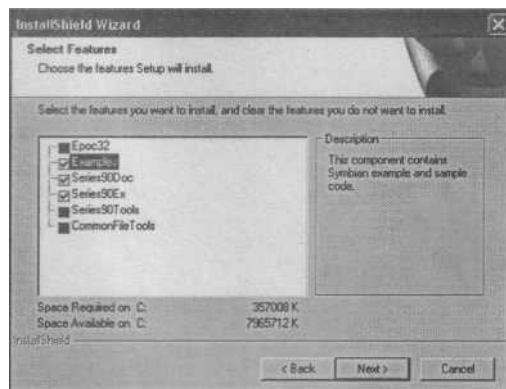


Рис. 4.17. Диалоговое окно Select Features серии 90

7. В конце установки SDK будет предложено сконфигурировать сеть, если вам это необходимо, то нажмите в диалоговом окне **Series 901.0 Beta 2 for Symbian OS Metrowerks Edition**, изображенном на рис. 4.18, кнопку ОК. Если сетевые настройки вам не нужны, нажмите кнопку **Cancel**.

В среде программирования Metrowerks CodeWarrior инструментальное средство разработчика серии 90 определится автоматически, а в C++ BuilderX необ-

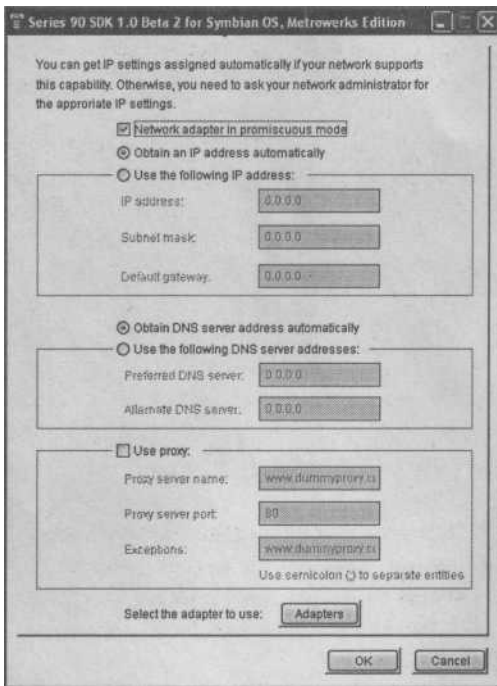


Рис. 4.18. Диалоговое окно **Series 90 1.0 Beta 2 for Symbian OS**

ходимо добавить пакет, выполнив команду меню **Tools => Symbian SDK Configuration**, и в диалоговом окне добавить установленный SDK серии 90 (подробности см. в главе 3, раздел 3.3). При работе с проектом, тестирование приложения производится с помощью эмулятора (рис. 4.19).

#### 4.2.4. Программа **SISAR**

В SDK серий 60 и 90 входит программа **SISAR** для упаковки готовой откомпилированной программы под платформу ARM I. Программу можно открыть, выбрав команду в меню **ПУСК => Все программы => Series**

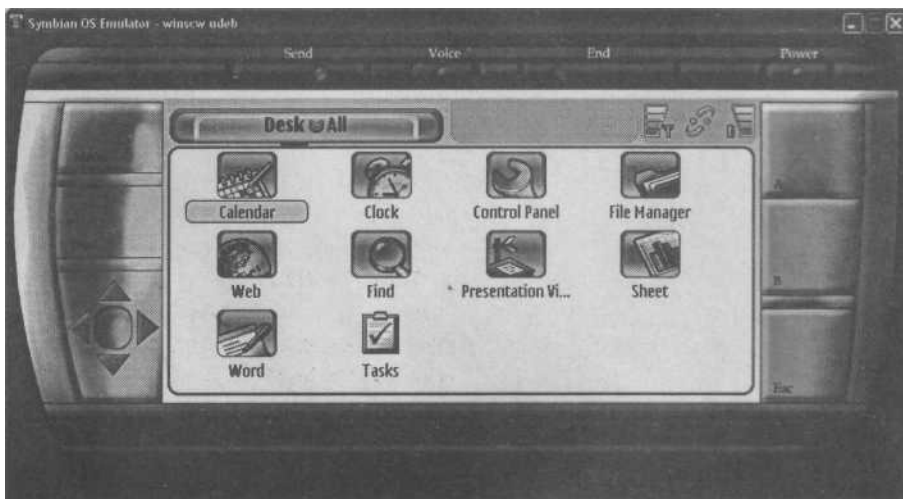


Рис. 4.19. Эмулятор коммуникаторов серии 90

**60 (90) Development Tools => SDK => Tools => Sisar**. Для корректной работы программы нужно, чтобы на компьютере была установлена Java Runtime версии 3.1.1 в папке JavaSoft (см. раздел 4.1 этой главы).



Работа с программой SISAR строится на проектах. Для создания нового проекта выберите в меню программы команду **File => New Project (Ctrl+N)**, и созданный проект откроется в окне программы SISAR, как показано на рис. 4.20. Как только вы создали проект, обязательно сохраните его в папке, где находится ваше приложение готовое к упаковке.

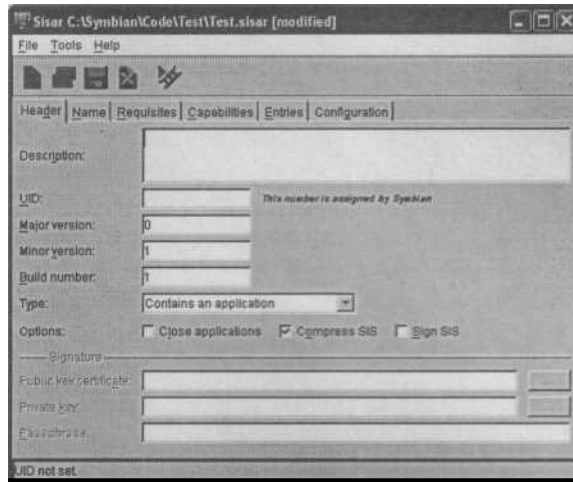


Рис. 4.20. Программа SISAR

Окно проекта разделено на шесть следующих вкладок:

- Header;**
- Name;**
- Requisites;**
- Capabilities;**
- Entries;**
- Configuration.**

В созданный и сохраненный проект нужно импортировать файл PKG, для этого выберите в меню программы команду **Tools => Import PKG**, в появившемся диалоговом окне **Choose PKG file to open** перейдите в каталог с программой и выберите файл PKG вашего приложения, нажав кнопку **Open**. При этом проект должен обязательно находиться в папке C:\Symbian\7.0s(6.1)\Series60(90)\Epos32. Если приложение выполнено правильно и откомпилировано для платформы ARM1, то на всех шести вкладках появятся различные атрибуты с полной информацией о программе. На рис. 4.21 изображена вкладка **Configuration** программы SISAR.

Для упаковки программы воспользуйтесь командой меню **Tools => Build SIS File (Ctrl+B)** и в директории, определенной для сохранения SIS-архива, на вкладке **Configuration** в поле **SIS file** появится готовый дистрибутив для работы на телефоне. Воздержитесь от создания дистрибутива до изучения главы 7.

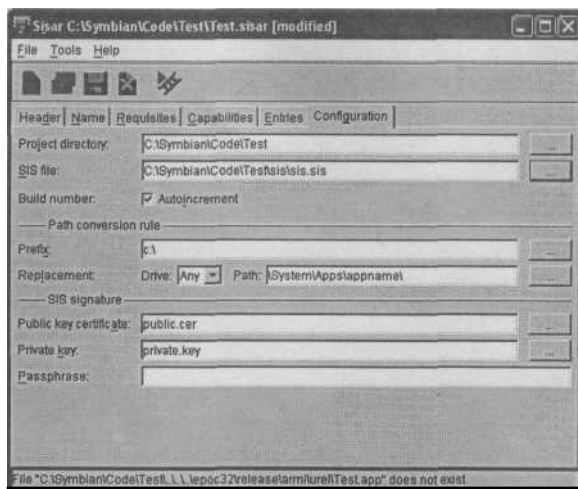


Рис. 4.21. Вкладка Configuration программы SISAR

## Глава 5. Архитектура Symbian OS

Большой класс устройств, представленных на сегодняшний день, позиционируются производителями как мобильные устройства: карманные персональные компьютеры (КПК), телефоны, смартфоны, коммуникаторы. Подразумевается, что эти устройства действительно мобильны как по скорости работы с данными, так и по своим размерам.

Технические характеристики мобильных устройств в силу малых размеров имеют ряд ограничений: маленькая мощность процессора, небольшое количество памяти и отсутствие накопителей большой емкости. Все эти критерии, конечно, влияют на используемое в устройствах программное обеспечение. Именно поэтому операционные системы в мобильных устройствах должны соответствовать определенным системным требованиям. Это вполне очевидно: невозможно установить Symbian OS на компьютер, а Windows XP на телефон. Поэтому при создании любой операционной системы учитывается аппаратная составляющая того или иного устройства, а разработчики программного обеспечения обязаны знать, на какой аппаратной архитектуре предполагается использование создаваемой ими операционной системы. На основе имеющихся сведений происходит разработка операционной системы, поэтому такое тесное сотрудничество компании, разрабатывающей операционную систему, и компании, создающей аппаратную часть устройства, является обязательным условием. Все мы видим плоды сотрудничества компаний Intel и Microsoft и это притом, что компьютеры в принципе могут быть любого размера, особых требований к компактности не предъявляется. Но в мобильных устройствах размеры самого устройства, в том числе аппаратная часть, весьма ограничены, никому не хочется носить в кармане кирпич с большим вентилятором. Используемая операционная система должна быть компактной и тщательным образом оптимизированной для работы на мобильном устройстве. Не стоит забывать и о стабильности в работе системы. В отличие от компьютерных операционных систем, мобильную операционную систему нельзя удалить, заменить или переустановить. Если все требования должным образом учтены, то пользователь получает хорошо отлаженную операционную систему, оптимизированную для определенной аппаратной архитектуры.

Операционная система Symbian OS — это тщательно продуманная многозадачная и объектно-ориентированная система. Множество программистов компании Symbian Ltd. работают над программным кодом операционной системы Symbian OS. Оптимизация программного кода Symbian OS под аппаратную составляющую устройства сказывается на надежности и стабильности работы этой системы. В этой главе мы поговорим как о программной архитектуре Symbian OS, так и об аппаратной части мобильных устройств.

## 5.1. Аппаратная архитектура

Мобильные телефоны по своим размерам невелики и использование мощных процессоров с большой тактовой частотой пока невозможно. Более мощным процессорам требуется соответствующее охлаждение, что ведет к увеличению размеров, стоимости и шума издаваемого устройством. Поэтому в смартфонах и коммуникаторах используются процессоры семейства *ARM* (Advanced RISC Machine), имеющие высокую производительность при малом энергопотреблении. Аббревиатура ARM используется в названии английской компании разрабатывающей так называемые RISC-микроконтроллеры, построенные на ядре ARM. Для простоты будем называть эти микроконтроллеры процессорами, коими, по сути, они и являются. У компании ARM нет своего производства процессоров, она лишь является разработчиком теоретической части процессора: электрических схем, топологии ядра процессора и описания на особом языке процессорного ядра. Все это передается производителям процессоров на основании лицензирования. В число производителей процессоров на базе ARM архитектуры входят такие известные компании, как Intel, Samsung, Motorola, Texas Instruments Inc.

Семейство процессоров ARM делится на несколько групп:

- ARM 4;
- ARM 5;
- ARM 6;
- ARM 7;
- ARM 9;
- ARM 10.

Каждая группа имеет свое ядро и может содержать несколько различных модификаций базового ядра или так называемого макроядра.

Семейство ядер от ARM 7 имеет 32-разрядную RISC архитектуру, обеспечивающую производительность от 130 до 220 миллионов операций в секунду. При этом энергопотребление может составлять не более 0,25 мВт/МГц, что очень хорошо для телефонов с ограниченными энергоресурсами. С другой стороны, такая частота работы процессоров (100-200 МГц и выше) при топологии в 0,35-0,18 мкм, может ввести в заблуждение относительно своей малой мощности. Засчет хорошо продуманной и разработанной системы кэширования макроядра и использования низкочастотной системной шины надобность в высокочастотных запросах отпадает. Применение низкочастотной шины ведет к малому потреблению питания, а пропускная полоса системной шины и памяти достаточна для решения серьезных задач.

Семейство ядер ARM 7 было построено на архитектуре фон-Неймана с общей памятью команд и данных. На сегодняшний день в смартфонах используется мощное ядро ARM 9, построенное на Гавардской архитектуре с разделенными шинами команд и данных.

Семейство ядер ARM характеризуется следующим набором свойств:

- высокопроизводительная 32-разрядная архитектура;
- модулированные операции по перемножению/аккумуляции 16 x 32 за один машинный цикл;

- совместимость с низковольтными CMOS технологиями;
- асинхронная и синхронная однопоточковая конфигурация процессора и шины;
- использование модуля управления памятью MMV, реализующего трансляцию логических адресов в физические и защиту памяти;
- система Thumb, обеспечивающая наилучшую плотность кода;
- поддержка технологии Jazelle для улучшения работы Java 2 ME программ;
- возможность использования специализированного математического процессора для работы с дробными числами (пока не используется);
- контроллеры памяти, клавиатуры и дисплея;
- интерфейсы USB, SDLC, IrDA

Все перечисленные достоинства процессоров семейства ARM при небольшой себестоимости и размерах делают эти процессоры незаменимыми при изготовлении мобильных телефонов. А разумная политика лицензирования компании ARM дает отличную базу для производства процессоров.

## 5.2. Системные библиотеки

При создании приложений под операционную систему Symbian разработчику можно использовать несколько языков программирования. Это C++, Java, Ассемблер, JavaScript, Visual Basic, OPL. Однако следует исходить из того, что родным языком для Symbian OS является все-таки язык C++. Язык программирования Java применяется при создании Java 2 ME приложений, и существует как бы параллельно языку C++, выполняя фактически тот же набор операций. Платформа Java 2 ME, поддерживаемая Symbian OS, реализована в виде отдельного модуля; поэтому создание программ на C++ и Java - это два абсолютно разных подхода, не пересекающихся между собой, но решающих примерно один и тот же круг задач. О программировании Java приложений под платформу Symbian OS вы узнаете из *главы 10*. В последующих главах в качестве основного языка программирования используется язык C++.

Каждая платформа, будь то UNIX, Windows или Linux, имеет определенный набор системных библиотек для упрощения, а иногда даже для единственной возможности работы с системой. Symbian OS в этом плане не исключение. Более того, создание практически всех программ строится на основе применения классов системной библиотеки (API). Операционная система Symbian OS содержит сотни классов, тысячи всевозможных функций для работы с датой, временем, строками, массивами, графикой, списками, диалогами, WAP, Bluetooth, IrDA TCP/IP, обработкой ошибок и многим другим. Работа с системой сводится к работе с библиотечными классами. Symbian OS очень компактная система со своими механизмами, направленными на эффективную работу всей операционной системы на телефоне. Поэтому в Symbian OS включено огромное количество системных классов, помогающих создавать программы. Такой подход одновременно и усложняет, и упрощает работу программиста. Когда мы говорим об усложнении работы в создании программ под Symbian OS, то понимаем и представляем,

что сложность заключается лишь в изучении имеющихся системных библиотек. Но с другой стороны, гораздо проще воспользоваться готовым материалом и построить программу, чем создавать этот материал самому. Для облегчения изучения API Symbian OS в книгу включено *приложение 1*, с подробным описанием основных системных классов для версии Symbian OS 7.0s. Системная библиотека значительно упрощает и облегчает процесс создания конечного программного продукта, поэтому одной из ваших первоочередных задач будет изучение имеющихся системных классов.

Системная библиотека разбита на множество разделов, в каждом из которых собран свой набор функциональных возможностей. Например, раздел **Graphics** (Графика) содержит классы, функции, макросы, структуры и константы, направленные на работу с графикой.

Как вы видите, схема подхода в Symbian OS для реализации собственных проектов схожа с платформой Java 2 ME или с классами MFC в Windows. Это вполне логично — разработчики Symbian OS позаботились о сторонних программистах, а, учитывая возможность возникновения критических ситуаций в работе некорректно созданного приложения, - это видится единственно правильным и взвешенным подходом.

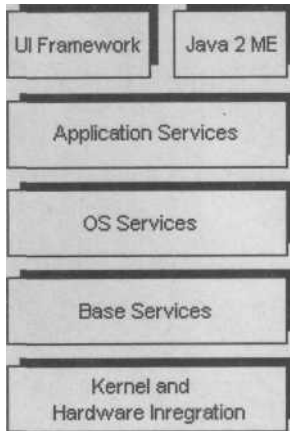
## 5.3. Программная архитектура

На сегодняшний день Symbian OS имеет несколько *версий*, последние из которых v6.1, v7.0, v7.0s, v8.0, v8.0a. Восьмая версия Symbian OS пока используется в одном телефоне (на момент написания книги). Ситуация с шестой и седьмой версиями тоже несколько запутана: на рынке сейчас очень много телефонов под управлением обеих версий Symbian OS и вытеснение версии 6.1 версией 7.x пока не происходит. Производители по-прежнему выпускают телефоны под управлением Symbian OS 6.1 и 7.x. Отчасти это связано с увеличением стоимости телефона, работающего с более новой версией Symbian OS. Но, по всей видимости, такое положение дел на рынке будет существовать еще долго. Наверно, только Symbian OS 8 или 9 в ближайшие годы потихоньку вытеснят предыдущие версии этой операционной системы. В связи с этим при разработке программ, вам предстоит столкнуться с некоторыми трудностями, но каждая версия, по сути, расширяет функциональные возможности предыдущей, поэтому присутствует обратная совместимость с программами для ранних версий Symbian OS. Мы остановимся на анализе архитектуры Symbian OS 7.0s.

Прежде всего, нужно понимать, что Symbian OS для производителей телефонов разделена на две части: ядро и графическую систему. Благодаря этому производители телефонов могут создавать свой вид пользовательского интерфейса, о чем уже не раз упоминалось в книге.

Программная архитектура Symbian OS создана по принципу модульного построения, состоящего из надстраиваемых друг над другом уровней. Посмотрите на рис. 5.1, где показана архитектура Symbian OS.

- Kernel and Hardware Integration — ядро и аппаратная часть системы;
- Base Services - базовые сервисы;



- OS Services - сервисы операционной системы;
- Application Services - пользовательские сервисы;
- UI Frameworks - инфраструктура пользователя тельского интерфейса;
- Java 2 ME - платформа Java 2 ME.

Всего Symbian OS включает в себя пять уровней и модуль поддержки технологии Java 2 ME. В свою очередь, каждый из перечисленных уровней содержит разный набор компонентов, на основе которых и происходит работа всей системы в целом. Описание компонентов платформы Java 2 ME вы найдете в *главе 11*. Давайте перейдем к рассмотрению составляющих каждого уровня.

Рис. 5.1. Архитектура Symbian OS

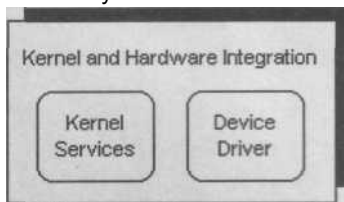


Рис. 5.2. Уровень Kernel and Hardware Integration

### 5.3.1. Ядро и аппаратная часть системы

Абстрактный уровень ядра и аппаратной части системы (Kernel and Hardware Integration) состоит из двух подсистем, изображенных на рис. 5.2.

*Ядро системы* (Kernel Services) оптимизировано для работы на процессорах архитектуры ARM с эффективным управлением всех имеющихся сервисов системы. Ядро системы обеспечи-

вает многопоточность работы, управление памятью и питанием, а также обеспечивает возможность переноса на любое аппаратное обеспечение.

Драйверы устройства (Device Driver) обеспечивают низкоуровневую поддержку программных контроллеров для следующих устройств:

- клавиатура;
- дисплей;
- карта памяти;
- цифровой преобразователь;
- инфракрасный и последовательный порты связи;
- USB1.1.

### 5.3.2. Базовые сервисы

Базовые сервисы системы (Base Services) обеспечивают основной или базовый каркас для последующих компонентов Symbian OS. Уровень базовых сервисов состоит из двух подсистем: Low Level Libraries (Низкоуровневые библиотеки) и Fileserver (Файл сервер). На рис. 5.3. представлен базовый уровень.

Компонент *Low Level Libraries* содержит низкоуровневые библиотеки и утилиты, с помощью которых можно решать задачи в следующих областях:

- криптография;
- базы данных;
- структура управления питанием;
- поддержка кодировок;
- работа с памятью;
- работа с архивами.

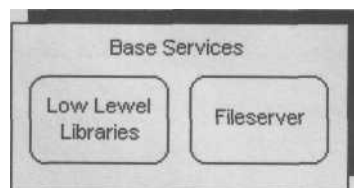


Рис. 5.3. Уровень Base Services

*Файл сервер* требуется для корректной работы с файловыми системами. Поддерживаются типы носителей:

- RAM (Random Access Memory)- оперативно-запоминающее устройство (ОЗУ), применяющееся для чтения и записи данных;
- NOR flash;
- NAND flash;
- MMC-карта памяти;
- SD-карта памяти.

### 5.3.3. Сервисы операционной системы

Сервисы операционной системы (OS Services) содержат набор компонентов инфраструктуры Symbian OS для работы с графикой, мультимедиа, криптографией, связью и так далее. Это полноценные микропрограммы, базовая составляющая которых основывается на предыдущих уровнях операционной системы.

Уровень OS Services разделен на четыре подсистемы, с набором различных компонентов. На рис. 5.4 изображен уровень OS Services.

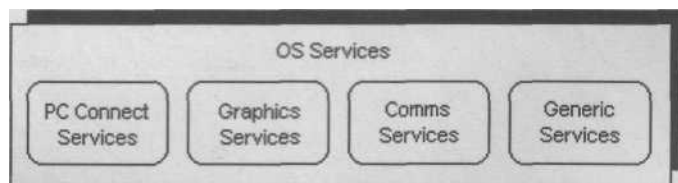


Рис. 5.4. Уровень OS Services

*Сервис связи с компьютером* (PC Connect Services) обеспечивает связь телефона с компьютером посредством специализированного программного обеспечения, а так же инструментальные средства разработчика (Toolkit) для создания программ на компьютере.

*Графический сервис* (Graphics Services) предусматривает работу с экраном и клавиатурой на основе графической подсистемы, предоставляя прямой доступ к экрану, устройству ввода и сглаживанию графики. Все это происходит на основе HAL (Hardware Abstraction Layer - уровень аппаратной абстракции).



*Сервис передачи данных* (Comms Services) обеспечивает для Symbian OS инфраструктуру коммуникаций. Прежде всего, это телефония (Telephony), работа с сетью (Networking Services) и сервис связи с последовательным и инфракрасным портами, USB и Bluetooth.

Система телефонии предоставляет возможность работы со стандартами:

- GSM (Phase2+),
- GPRS (r4, Class B),
- CDMA 2000 (1x),
- EDGE (ECSD, EGPRS),
- WCDMA(r4).

Сетевой интерфейс поддерживает протоколы связи:

- TCP, IPv4, IPv6, MSCHAPv2;
- IPSec;
- TCP/IP;
- WAP;
- множественная адресация.

В свою очередь *сервис связи* предусматривает работу с основными средствами связи:

- IrDA;
- USB;
- Bluetooth.

Подсистема в уровне OS Services - это Generic Services (Общие сервисы), состоит из двух сервисов: Cryptography Services (Криптография) и Multimedia (Мультимедиа). Криптография отвечает за безопасность системы в области криптографии, управления сертификатами и инсталляции программного обеспечения на телефон. В криптографии поддерживаются стандарты:

- DES;
- Q 3DEC;
- AES;
- RC2;
- RC2-128;
- RC4;
- RC5;
- RSA;
- DSA;
- DH;
- PKCS#7.

*Система мультимедиа* необходима для работы со звуком, видео и графикой (как 2D, так и 3D). Работа с этими компонентами осуществляется через соответствующие системные библиотеки. На рис. 5.5 представлена составляющая системы мультимедиа.

Создание трехмерных игр в Symbian OS возможно при соответствующей аппаратной поддержке и на основе OpenGL ES. Работа с 2D графикой построена

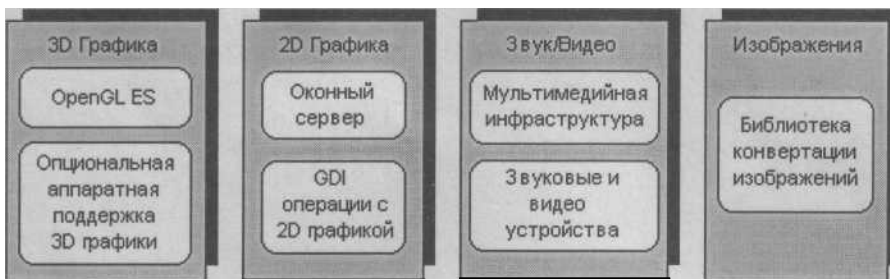


Рис. 5.5. Система мультимедиа

через GDI (Graphics Device Interface - интерфейс графического устройства) системы Symbian. Также все телефоны под управлением Symbian OS поддерживают работу со звуком и видео.

### 5.3.4. Пользовательские сервисы

Уровень пользовательских сервисов (Applications Services) инкапсулирует различные механизмы, обеспечивающие пользователю работу с данными. Symbian OS содержит встроенный пакет приложений, таких как: календарь, заметки, будильник, передача SMS, доступ к электронной почте и так далее. Сервис пользовательских услуг состоит из четырех подсистем, изображенных на рис. 5.6.

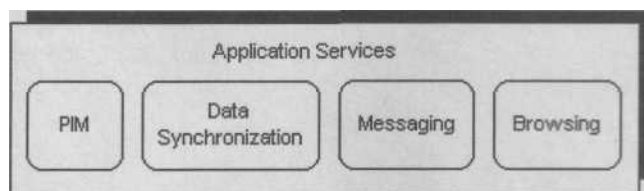


Рис. 5.6. Уровень пользовательских сервисов

*PIM*(Personal Information Manager - персональный информационный менеджер) обеспечивает стандартные механизмы по работе с пользовательскими данными. Примером могут служить простой органайзер, записная книжка или офисные приложения, реализованные в Symbian OS. Имеется большой набор API для создания своих пользовательских программ.

*Синхронизация данных* (Data Synchronization) построена на основе механизма OMA SyncML 1.1, обеспечивающего синхронизацию данных по принципу сервер/клиент.

*Передача сообщений* (Messaging) поддерживает все основные виды сообщений:

- SMS;
- EMS;
- MMS.

Также поддерживается работа по протоколам POP, SMTP/ШАР для передачи и приема, например, электронной почты. Осуществляется поддержка WAP, HTTP, XHTML, а системная библиотека имеет множество классов для создания своих программ.

### 5.3.5. Инфраструктура пользовательского интерфейса

Инфраструктура пользовательского интерфейса (UI Framework) — это система, на основе которой производители мобильных устройств могут создавать свой графический интерфейс на базе механизмов Symbian OS. Это мудрое и взвешенное решение компании Symbian Ltd. по разделению системы на ядро и графику. Инфраструктура пользовательского интерфейса состоит из двух компонентов. Первый - UI Applications Framework (Прикладная инфраструктура UI) - предоставляет возможность в создании собственного пользовательского интерфейса, который вы можете наблюдать на своем телефоне. Второй - UI Toolkit (Инструментальные средства разработчика пользовательского интерфейса). На основе этих средств производители телефонов разрабатывают свои SDK, с которыми вы познакомились в *главе 4* и с помощью которых программисты создают свои программы. Такой подход расширяет круг производителей, заинтересованных в портировании Symbian OS на свои модели телефонов.

## 5.4. Файловая система

В отличие от операционных систем, предназначенных для работы на компьютерах, Symbian OS не может быть переустановлена на телефоне. Компьютерные системы содержат жесткий диск больших размеров, CD-ROM для установки системы, да и в целом весь подход смены и переустановки операционной системы иной. Symbian OS изначально создавалась для работы на телефоне, поэтому основные требования — это стабильность и компактность, так как смена системы невозможна. Естественно, в заводских условиях при наличии соответствующего оборудования и самой Symbian OS сделать это возможно, но такие действия могут осуществлять лишь лицензированные компании, производящие телефоны.

Операционная система Symbian располагается в памяти ROM (Read Only Memory - постоянное запоминающее устройство (ПЗУ)) - это небольшая по размеру микросхема, находящаяся внутри телефона с возможностью только чтения данных. Как уже говорилось, перепрограммировать ее можно только в заводских условиях. Особенности питания телефона все данные (как пользовательские, так и системные) остаются в целости и сохранности. На основе памяти ROM в Symbian OS и формируется *файловая система* на базе логических дисков. Посмотрите на рис. 5.7, где изображена файловая систе-

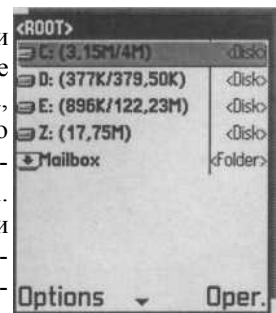


Рис. 5.7. Файловая система N-Gage

ма телефона N-Gage серии 60. С файловой системой нам помогает разобраться файловый менеджер FileMan 1.01 компании SymbianWare.

Логические диски C, D и Z лежат в области ROM памяти, формируя тем самым файловую систему. Диск E - это карта памяти. Symbian OS поддерживает работу с 26 буквами английского алфавита от A до Z. Но чаще всего производителями используются буквы C, D, Z, E и G по аналогии с компьютерными системами.

### 5.4.1. Диск!



Рис. 5.8. Свойства диска Z

Диск Z - это основная и самая большая часть ROM памяти, где прошита Symbian OS. Если запустить файловый менеджер и в установках выставить возможность доступа к ROM памяти, то можно «порыться» в диске Z и посмотреть имеющиеся системные папки. А если в том же FileMan выделить курсором любой диск и нажать на клавиатуре кнопку 5 (Свойства), то появится информационное окно со свойствами данного логического диска. Посмотрите на рис. 5.8, где показаны свойства диска Z на телефоне Nokia N-Gage. Все данные системы, прошитые на диске Z, удалить при помощи файлового менеджера невозможно. Диск Z имеет автономное энергоснабжение.

### 5.4.2. Диск C

Логический диск C в серии 60 имеет размерность от 3,4 Мб до 10 Мб, но в среднем это около 4 Мб. У Sony Ericsson, Motorola размерность несколько выше. Из всех доступных 4 Мб порядка 700 Кб занимают системные файлы, куда входят различные пользовательские приложения, звуковые и видеофайлы, которыми изначально комплектуется телефон. При установке программ лучше не ставить приложения в эту область памяти, а использовать для этих целей карту памяти. Тем более что при установке программ некоторые данные, очень малень-

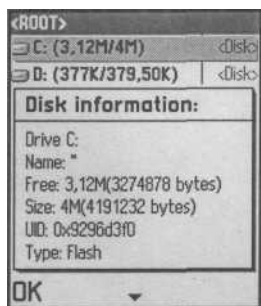


Рис. 5.9. Свойства диска C

кие по размеру, все равно располагаются на логическом диске C. Например, при инсталляции программы с ограниченным сроком работы, эта программа обязательно расположит некий идентификационный файл, на основе которого будет исчисляться время работы этой программы. Конечно, при желании найти такие файлы можно и вопрос здесь скорее нравственный: покупать или не покупать. Как программист знать об этом вы обязаны и, создавая уже свой коммерческий продукт, вам нужно будет спрятать идентификационный файл похитрее и подальше от шаловливых рук пользователей. Диск C так же независим от энергоснабжения. На рис. 5.9 изображены свойства диска C.

### 5.4.3. Диски

Логический диск D - это небольшой по размеру виртуальный диск оперативной памяти (RAM - Random Access Memory), содержащий временные прикладные и системные данные. При выключении телефона все данные на диске D будут утеряны. Диск D инициализируется каждый раз вновь при включении устройства.

### 5.4.4. Диск E

Диск E в телефоне представлен сменным накопителем или картой памяти. Каждый производитель поддерживает тот или иной стандарт карт памяти, но принцип работы один и тот же. На карту памяти пользователь устанавливает программы, копирует музыку, видео, изображения и так далее. При удалении файлов с карты памяти нужно так же быть осторожным. Если вы не знаете, что это за файл, лучше его не удалять. Тем более, что при установке карты памяти на телефон, Symbian OS автоматически создает системные папки на этом накопителе. И, конечно, при установке программ на карту памяти отдельные файлы устанавливаемой программы так же попадают в системные папки карты памяти. Посмотрите на рис. 5.10, где представлена системная папка System, находящаяся на карте памяти.

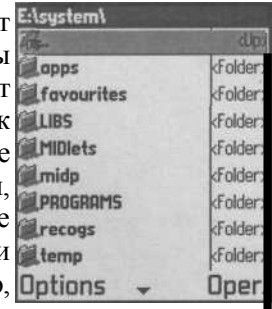


Рис. 5.10. Папка System на смартфоне Nokia N-Gage

Например, могу рассказать вам случай из жизни. Моя жена на своем смартфоне N-Gage устанавливала Java-игру, купленную в Интернете. Для этого на диске E у нее была создана папка E:\Java. После инсталляции игры в установленную директорию, там появились два файла с расширениями \*.jad и \*.jad, что является показателем установленной Java- программы. Удаление Java-приложений необходимо осуществлять через команду меню **Удалить** в окне **Приложения**. Спустя месяц после установки Java-игры, она забыла про установленную игру (и немудрено, на ее телефоне я насчитал около 60 игр и 20 разных программ) и по ошибке удалила папку Java с диска E, предполагая, что эта папка пуста. Через некоторое время она решила установить новую Java-игру на телефон, но не тут-то было. Открыть программу установки Java игр **Приложения** не удавалось - все время появлялось информационное окно, сообщающее о системной ошибке. То есть доступ к программе **Приложения** пропал. Дело в том, что при открытии этой программы, она моментально инициализирует установленные программы по регистрационным данным, которые находятся в папке E:\System\midp. Регистрационные данные - это простой файл специфического формата, где прописаны логи установки, в данном случае Java программ. Кстати, нечто подобное реализовано и при установке программ на C++. Только в этом случае все регистрационные данные размещаются в каталоге C:\System\Install\install.log. Ситуация была пато-

вой — любая попытка установки Java программы через файловый менеджер по Bluetooth и через программу установки Java приводила к системной ошибке. И, только установив удаленную игру на мой смартфон N-Gage QD и передав файлы с расширениями JAR и JAD по каналу Bluetooth на телефон жены в созданную папку Java, удалось решить эту проблему. Поэтому надо быть предельно осторожным при работе с файловыми менеджерами, тем более что в некоторых менеджерах не реализована функция подтверждения действий по удалению файлов. То есть, при выполнении команды «удалить», избранный файл безвозвратно удаляется, и нет никакой корзины. С другой стороны, без периодической очистки системы тоже обходиться нельзя. Механизм удаления программ в Symbian OS сделан на высшем уровне, но все же некоторые программы раскидывают свои «щупальца» по всей файловой системе телефона. Например, платные коммерческие программы делают это специально, размер таких файлов не критичен и может составлять всего несколько килобайт.

### **5.4.5. Оперативная память**

Оперативная память, как вы понимаете, не менее важная часть системы. Не зря придуман лозунг: «Памяти мало не бывает». Symbian OS использует оперативную память, так же как и компьютерные системы. Память интегрирована в телефон и возможности ее увеличения нет. Но, исходя из того, что Symbian OS «мягкая» по характеру система, то и объем оперативной памяти необходимый для корректной работы системы небольшой. По минимальным требованиям Symbian OS - это 8 Мб, но производители телефонов обычно значительно увеличивают эту цифру. Например, в смартфоне Siemens SX1 интегрировано 16 Мб оперативной памяти. Подобно компьютерным системам при включении телефона определенные системные данные загружаются в оперативную память для работы Symbian OS. Когда вы запускаете программу на телефоне, она отбирает себе объем памяти, необходимый уже для работы запущенной программы. Объем памяти, необходимый для работы приложения, обычно не превышает 3 Мб. Программе память доступна в виде стека и кучи (динамически выделенная память). В каждом новом процессе количество памяти ограничено для стека от 1-3 Мб, а для кучи 8-35 Кб. Числа варьируются в зависимости от производителей телефонов. Эти цифры не зависят от количества используемых потоков. При запуске программы стек распределяется статически на время работы приложения, а объем динамической памяти возрастает по мере необходимости. Оперативная память не содержит резервного питания, и при отключении устройства все данные будут утеряны, что аналогично принципу работы компьютерных систем.

В конце главы еще раз хочется упомянуть о размере создаваемой программы. В Symbian OS, к счастью, особой проблемы с объемом программ нет. Поддерживается работа с картами памяти, размер которых достаточно большой для установки десятков различных программ. Это, конечно, очень большой и жирный плюс для всей системы, способствующий созданию все новых и новых шедевров игростроения.

## Глава 6. Основы программирования в Symbian OS

У каждого программиста с течением времени формируется свой стиль написания исходного кода программ. Некоторые используют знаки подчеркивания в названиях классов, функций или переменных, другие эти же названия прописывают с заглавной буквы, игнорируя знаки подчеркивания. Каждый программист пытается выдумать что-то свое, индивидуальное, иногда даже понятное только ему одному. К сожалению, подобный подход в стандартизации названий классов, функций, переменных и структур не всегда оправдан. Разумеется, в реализации исходного кода - это некий индивидуализм и «глоток свежего воздуха» для программиста. Но данный подход возможен, если работа над проектом занимается небольшая группа, где каждый программист знаком со стилем партнера. А если в проекте участвует, предположим, пятьдесят человек и более, что тогда? Именно поэтому компании всего мира определяют для себя различные приоритеты и стандарты в написании исходного кода для упрощения общей реализации проектов. Каждая компания, которая действительно занимается программированием и ведет бизнес, построенный на создании программного обеспечения, имеет соответствующую документацию по стандартизации правил написания исходного кода программ. Например, несколько лет назад я «удаленно» работал на польскую компанию. Так вот первое, что они мне прислали по электронной почте - это требования в формате PDF, состоящие из 14 страниц и регламентирующие написание программного кода. А в одной австрийской компании, где работает мой хороший знакомый, даже предусмотрена система штрафов за неправильное написание названий в исходном коде. Точно такие же правила есть для Windows программирования - это *венгерская система написания*, которая, правда, не всегда применяется программистами.

Стандартизация написания исходного кода необходима, она упрощает разработку проектов. И если, программиста уволят, переведут, то разобраться с исходным кодом, над написанием которого он работал, скажем, в течение полугода, не составит труда любому другому программисту этой компании. Поэтому стандарты важны, их стоит придерживаться. Это одно из первых требований, предъявляемых программисту при устройстве на работу в одну из компаний, связанную с разработкой программного обеспечения под любую платформу.

Компания Symbian Ltd. при создании системных библиотек так же придерживалась определенного стандарта, на основе которого работают программисты всего мира. Этих требований необходимо придерживаться и вам, для того чтобы понимать исходный код, созданный другими людьми и наоборот. Поэтому нач-

нем с освещения системных библиотек, по принципу которых вам надо строить свой программный код, а в конце этой главы будут перечислены основные правила написания «правильного» кода.

## **6.1. Классы**

Системная библиотека платформы Symbian OS содержит множество predefined классов для работы с графикой, диалогами, строками, массивами, функциями печати, контролем над событиями, получаемыми с клавиш телефона и многим другим. Об основных классах API Symbian OS вы можете узнать из *приложения 1*. В названиях классов используются префиксные английские заглавные буквы, подразделяющие классы на категории. Применяя такую систему работы с классами, можно очень легко добиться прозрачности и понятности использования созданных классов в любом месте исходного кода всей программы.

При работе с классами в Symbian OS используются префиксные английские заглавные буквы: C, R, M и T, обозначающие ту или иную область использования данного класса. Еще употребляется буква D, но этот вид классов применяется в низкоуровневом программировании для ядра системы Symbian OS. Необходимо иметь в виду, что при работе с классами с, R, M и T система очистки или механизм освобождения захваченных ресурсов абсолютно разный.

### **6.1.1. Классы C**

Классы с заглавной префиксной буквой C - это любые производные или наследуемые классы от основного класса CBase. При создании производных классов от CBase на основе оператора new (ELeave), все создаваемые члены класса данного объекта автоматически обнуляются. Классы C по умолчанию находятся в динамически распределенной памяти (в куче). По окончании работы с классами C объекты должны обязательно удаляться из памяти.

### **6.1.2. Классы R**

Классы, начинающиеся с буквы R, ассоциируются с классами ресурсов. При создании объектов этих классов, они по умолчанию размещаются в динамически распределяемой памяти.

По окончании работы с классами R необходимо позаботиться о явном освобождении захваченных ресурсов и выгрузке объектов из памяти. Сделать это можно с помощью системной функции Close ().

### **6.1.3. Классы T**

Классы, использующие заглавную букву T, являются базовыми классами для всех типов данных в API Symbian OS. В связи с этим классы T не имеют деструктора, а работа с такими классами подобна работе с типами данных. При создании



объектов классов T их размещение может происходить в любой области памяти. Так как распределение памяти не происходит, то и требования для очистки минимальны.

#### **6.1.4. Классы M**

В API Symbian OS с заглавной буквы M начинаются названия всех интерфейсов. В состав интерфейсов входит набор предопределенных виртуальных функций, доступ к которым можно получить через классы, реализующие данный интерфейс. Все классы, реализующие тот или иной интерфейс, начинаются с заглавной буквы M

#### **6.1.5. Статические классы**

Статические классы не имеют определенных правил написания. В Symbian OS статические классы применяются в основном как контейнеры для библиотечных функций.

### **6.2. Функции**

В названиях функций в Symbian OS применяются англоязычные глаголы, обозначающие направленность действий для функций. Например, функция Draw () рисует на экране телефона некие графические элементы. Название функции начинается с заглавной буквы. Возможно использование и двух-трех слитно написанных слов, например: SetCommandHandler (). Можно придумывать названия и с большим количеством слов, но все же необходимо соблюдать преемственную логику названий для удобочитаемости исходного кода. Много API-функций имеют в окончании своих названий заглавные буквы L и LC, а так же в самих названиях используются английские слова Set (установить) и Get (получать).

#### **6.2.1. Уходящие функции**

В системной библиотеке Symbian OS есть несколько видов функций, два из них - это уходящие функции (leaving function) и неуходящие функции (non-leaving function). Эти технические термины влияют на сущность двух видов функций.

Symbian OS очень компактная система, работа которой рассчитана на устройствах с минимальными системными ресурсами. Вследствие этого в Symbian OS концепция освобождения захваченных ресурсов является важнейшим действием, которое необходимо учитывать при создании программ. Сидя за компьютером при «зависании» операционной системы, вы с легкостью можете перегрузить ее при помощи кнопки Reset на системном блоке, что абсолютно недопустимо для телефона. Поэтому в Symbian OS применяется одна из наиболее продуманных систем освобождения ресурсов. Например, работая с одной из программ в Symbian OS, вы пытаетесь загрузить какой-то файл ресурса этой программы,

и по каким-то независящим от вас обстоятельствам данного ресурса нет. Вызвав функцию, вы создаете поток и соответственно выделяете память. Если в итоге вызов этой функции терпит неудачу, то система обязана позаботиться о корректном ее завершении. В Symbian OS такой метод реализован программно на основе *уходящего вида функций*, которые используют в конце своего названия заглавную букву L, например, NewL (), CreateL (), RunL () и так далее. Но это не значит, что вы можете сделать в программном коде массу ошибок и надеяться, что системные функции все за вас сделают. Этот вид функций заботится лишь об освобождении системных ресурсов, так что исправить свои ошибки сможете только вы сами.

### **6.2.2. Неуходящие функции**

Этот вид функций не имеет каких-либо отличий, но, как правило, это английский глагол, написанный с заглавной буквы и означающий действие, выполняемое функцией. Обычно *неуходящие функции* выполняют различные обобщенные действия, например, Exit () - выход.

### **6.2.3. Функции LC**

Еще один вид уходящих функций системной библиотеки имеет в конце названия заглавные буквы LC. Эти функции необходимы при работе со стеком. *Стек* -это область памяти, где содержится адрес возврата при вызове функции. Такой вид функций размещает определенные данные в стек и впоследствии заботится об освобождении или очистке ресурсов, связанных с вызовами этих функций.

### **6.2.4. Функции Set**

Функции, использующие в своих названиях слово Set, например, SetPenColor (), применяются для установки различных данных, свойств или параметров. То есть этот вид функций, просто задает набор данных для дальнейшей работы с ними. Например, с помощью функции SetColor () можно задать цветовую компоненту для графического элемента.

### **6.2.5. Функции Get**

Функции, использующие в названиях слово Get, например, GetCurrentItem (), призваны обеспечивать получение определенных комплексных данных. Например, ширина и высота дисплея телефона, на основании которых будет происходить дальнейшая работа с графическим контекстом.

## **6.3. Структуры**

В приложениях под Symbian OS применяются структуры C-подобного стиля. В API содержится небольшое количество структур, особенно они актуальны для 6 и 5 версий Symbian OS. Системная библиотека версии Symbian OS 7 уже менее

привязана к структурам, но поскольку имеется обратная совместимость поздних версий API с ранними версиями, то на использование структур ограничений нет. Тем не менее, рекомендуется применять классы T вместо структур в своих проектах.

## 6.4. Макросы

Макросы в программировании под платформу Symbian используют C-подобную конструкцию записи, например, `IMPORT_C`. При отладке можно еще применять макросы с двойным подчеркиванием в начале слова и в конце: `_WINS`.

## 6.5. Имена переменных

В качестве имен переменных и объектов классов всегда лучше использовать близкое по смыслу существительное. Если это дисплей, то пусть он и будет Display, нежели d или Dy. Короткие названия очень быстры для написания, но, к сожалению, не отражают смысла переменной или объекта класса, поэтому лучше придерживаться осмысленных названий. Возможно, не все читатели знают в совершенстве английский язык, и иногда в демонстрационных примерах я использую английскую транскрипцию русских слов. Например, команда выбора может называться Vibor, это сделано для облегчения работы с книгой, но в профессиональной практике лучше не применять такие названия.

При работе с константами в Symbian OS в качестве префикса ставится заглавная буква K. Еще существуют так называемые перечисляемые константы, и в этом случае употребляется заглавная буква E, например, EDecember, EJanuary, EFebruary. Обычно этот вид перечислений используется в работе с ресурсами приложения.

Префиксная буква a используется при работе с аргументами. В названиях нестатических переменных и членов данных (date members) применяется буква i в качестве префикса к названию.

Что касается глобальных переменных, то ярко выраженных требований в написании названий нет, но рекомендуется свести процент использования глобальных переменных к нулю.

## 6.6. Простые типы данных

Для работы с различными типами данных в Symbian OS определены свои простые типы данных, на основе которых создаются переменные в исходном коде. Чтобы простые типы данных были доступны, в программу нужно подключить системный заголовочный файл `e32defs.h`. Рассмотрим типы данных в Symbian OS:

- Tint - целочисленное 32-битное значение размером не менее четырех байт;
- TUInt - целочисленное 32-битное значение со знаком размером не менее четырех байт;
- TInt64 - 64-битное целочисленное значение размером в восемь байт;

- TI n 18 - 8-битное целое значение размером в один байт;
- TInt16 - 16-битное целое значение размером в два байта;
- TInt32 - 32-битное значение размером в четыре байта;
- TInt8 - 8-битное целочисленное значение со знаком размером в один байт;
- TInt16 - 16-битное целое значение со знаком размером в два байта;
- TInt32 - 32-битное целочисленное значение со знаком размером в четыре байта;
- TReal - число с плавающей точкой размером в восемь байт;
- TReal32 - 32-битное число с плавающей точкой размером в четыре байта;
- TReal64 - 64-битное число с плавающей точкой размером в восемь байт;
- TRealX — число с плавающей точкой в диапазоне от  $\pm 1 \times 10^{\sim 9863}$  до  $\sim \pm 1 \times 10^{\sim 9863}$  размером в двенадцать байт;
- TText8 - 8-битное символьное значение размером в один байт;
- TText - символьное значение с использованием уникада (Unicode) размером в два байта;
- TText16 - 16-битное символьное значение с использованием уникада размером в два байта;
- TChar - 32-битное символьное значение с использованием уникада размером в четыре байта;
- TBool - логический тип данных, содержащий переменные ETrue и EFalse размером в четыре байта.

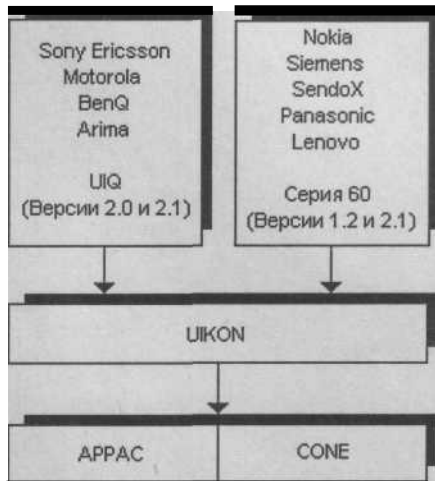
## 6.7. Рекомендации

В этом разделе вы найдете множество рекомендаций по правильному написанию исходного кода программ.

1. Старайтесь писать компактный и понятный код.
2. При компиляции программного кода добейтесь минимума предупреждений от компилятора (Warnings 0).
3. Всегда придумывайте осмысленные названия для классов, функций, объектов, переменных и так далее.
4. Старайтесь не использовать глобальные переменные.
5. Исключайте в названиях знаки подчеркивания, если это не макросы.
6. Все названия старайтесь писать с заглавных букв с присоединением, где необходимо префиксов.
7. Создавая переменную, инициализируйте ее сразу.
8. При создании нескольких переменных одного типа не пишите их через запятую. Каждую переменную объявите в новой строке кода и инициализируйте.
9. Если используете сокращения в названиях, придумайте его так, чтобы это название стало очевидным.
10. Старайтесь избегать длинных названий, они значительно ухудшают общее восприятие кода в целом.

11. Щедро документируйте свой код комментариями, чтобы через полгода можно было вспомнить, что же вы все-таки разрабатывали.
12. При подключении заголовочных файлов убедитесь, что вы подключили только нужные файлы, а не все подряд на всякий случай.
13. Не экспортируйте виртуальные, подставляемые и не библиотечные функции в программу.
14. При создании указателя, оператор звездочка (\*) должен быть написан у названия типа, а не у указателя, например: TBar\* Stol;
15. Системная библиотека Symbian OS включает множество predefined классов на все случаи жизни. Применяйте классы API в альтернативу своим созданным классам.
16. Создавая свой класс, определите объявление класса в заголовочном файле (\*.h), а описание в исходном файле (\*.cpp).
17. Избегайте рекурсии в программах.
18. Будьте очень осторожны при приведении типов.
19. Сложные и большие действия разбивайте на более мелкие задачи.
20. Старайтесь использовать отладочные макросы API при создании программ.
21. Symbian OS создана в лучших традициях ООП (объектно-ориентированное программирование), поэтому проектируйте свои программы исходя из этой концепции.

Здесь перечислены основные, но далеко не все рекомендации по созданию хорошего кода. Старайтесь следовать перечисленным выше инструкциям, создавая свои программы. По мере работы с Symbian OS вы поймете, что лучше потратить больше времени на написание хорошего кода, чем написать плохой код и потратить еще больше времени на его отладку.



классов, обеспечивающих создание графической оболочки системы. На рис. 7.1 представлена схема взаимодействия системных классов Symbian OS с UIQ и серией 60.

Два системных уровня абстракции CONE и APPARAC располагаются на уровне ядра системы Symbian OS.

- APPARAC (Application Architecture - архитектура приложения) содержит каркас системных классов для работы с приложением и прикладными данными;
- CONE (Control Environment - система управления) включает системные классы для графического взаимодействия с программой.

Уровень *Uikon* содержит множество системных классов Symbian OS, а три класса представляют графический пользовательский интерфейс (GUI) - это Application (Приложение), Document (Документ) и App UI (Пользовательский интерфейс приложения). Три перечисленных компонента уровня *Uikon* представлены системными классами с одноименными названиями.

- Класс CEikApplication - приложение;
- Класс CEikDokument - документ;
- Класс CEikAppUi - пользовательский интерфейс приложения.

Все эти три класса уровня *Uikon* являются базовыми классами, над которыми надстраиваются платформы UIQ и серии 60. Иначе говоря, системные классы UIQ и серии 60 являются производными от системных классов CEikApplication, CEikDokument, CEikAppUi уровня *Uikon* операционной системы Symbian.

В этом разделе уже несколько раз использовалась аббревиатура UI (Пользовательский интерфейс или интерфейс пользователя) и GUI (Графический пользовательский интерфейс). Это две разные составляющие одного приложения. Не путайте их между собой!

*Интерфейс пользователя (UI)* - это совокупность различных элементов пользовательского интерфейса приложения, а именно вкладки, диалоги, переключатели, списки, информационные окна, кнопки, флаги, то есть все то, что определяет как общий вид программы, так и дает возможность пользователю взаимодействовать с приложением.

*Графический пользовательский интерфейс (GUI)* - это каркас или база, на котором происходит построение как UI, так и всего приложения в целом. Например, в Windows программировании, прежде всего, необходимо создать и зарегистрировать класс Windows, реализовать обработчик и главный цикл событий. В итоге получится окно, то есть некий каркас для дальнейшего развития программы. Абсолютно тот же самый прием используется и в программировании под Symbian OS, но уже со своим индивидуальным подходом.

Более того, в Symbian OS имеется еще одно похожее название *GDI* (Graphic Device Interface - графический интерфейс устройства). Это системный уровень абстракции Symbian OS, который находится на уровне ядра и отвечает за связь графической системы и аппаратной части устройства. Интерфейсы UI, GUI и GDI очень похожи в своих названиях и, несомненно, связаны между собой, но это

три разных составляющих одной большой системы и ни в коем случае их нельзя путать между собой.

### 7.1.1. Платформа UIQ

Платформа UIQ, созданная компанией UIQ Technology AB и применяемая в телефонах Sony Ericsson, Motorola, BenQ, Arima, является своего рода стандартом для Symbian OS. Как правило, телефоны, использующие UIQ, имеют разрешение экрана как минимум 208 x 320 пикселей и обладают при этом полноценным сенсорным дисплеем. Поэтому интерфейс пользователя UIQ значительно отличается от серии 60 и содержит собственный набор классов *Qikon*, надстраиваемых над системными классами Uikon операционной системы Symbian. Классы UIQ (Qikon) - производные от классов Uikon, они очень похожи в своих названиях. Посмотрите на табл. 7.1, где показана связь классов Uikon и Qikon.

**Таблица 7.1.** Связь классов Symbian OS (Uikon) и платформы UIQ (Qikon)

	Symbian OS - Uikon	UIQ - Qikon
Application	CEikApplication	CQikApplication
Document	CEikDocument	CQikDocument
AppUI	CEikAppUi	CQikAppUi

Можно заметить, что названия классов практически идентичны и лишь префикс Qik указывает на принадлежность классов к платформе UIQ. Создавая программы для UIQ, вы будете использовать именно эти ключевые классы Qikon, которые напрямую взаимодействуют с системными классами Uikon. Естественно, существует огромное количество классов UIQ, но именно на основе этих трех классов формируется каркас (GUI) программы для UIQ.

### 7.1.2. Серия 60

Серия 60 специально адаптирована на меньшее разрешение экранов телефонов - это как минимум 176 x 208 пикселей. Сенсорный дисплей в этой платформе не предусмотрен и взаимодействие с пользователем осуществляется с помощью клавиш телефона. Поэтому серия 60 имеет свои системные классы под названием Avkon. В табл. 7.2 показана связь системных классов Symbian OS Uikon и классов серии 60 Avkon.

**Таблица 7.2.** Взаимодействие классов Symbian OS (Uikon) и классов серии 60 (Avkon)

	Symbian OS - Uikon	Серия 60 - Avkon
Application	CEikApplication	CAknApplication
Document	CEikDocument	CAknDocument
App UI	CEikAppUi	CAknAppUi и CAknViewAppUi



По аналогии с UIQ в серии 60 (Avkon) применяются префиксы Akn для названий системных классов этой платформы. Более того, прибавился еще один дополнительный класс CAknViewAppUi и это как раз связано с меньшим разрешением экрана телефона. То есть через класс CAknViewAppUi происходит своего рода адаптация некоторых частей графической оболочки под телефоны серии 60. Так же как и в UIQ (Qikon), в серии 60 (Avkon) четыре перечисленных класса выступают каркасом, на котором базируются программы для серии 60.

## 7.2. Базовая составляющая приложения

При создании полноценных программ под операционную систему Windows, необходимо реализовать несколько обязательных компонентов, на основе которых строится графический интерфейс пользователя (GUI). Без создания этого каркаса любая графическая составляющая программы будет просто недоступна, и максимум на что вы сможете рассчитывать, это на консольно-ориентированную программу, возвращающую вас во времена DOS. Такой вид программ уже давно не интересен пользователям, да и заменить интуитивно понятный пользовательский интерфейс командной строкой невозможно. Поэтому GUI Symbian OS состоит из каркаса классов, реализовав которые, вы получаете возможность для построения дальнейшей программы.

Когда в *главах 2 и 3* были созданы демонстрационные примеры на базе шаблона HelloWorld с использованием Metrowerks CodeWarrior и C++ BuilderX Mobile Edition, вы, вероятно, заглянули в папки с исходными кодами. И обнаружили там массу непонятных вещей, но четыре автоматически сформированных класса вас наверно заинтересовали в большей степени. Как раз эти четыре класса и составляют графический пользовательский интерфейс приложения для Symbian OS. Реализация этих классов предоставляет программисту минимальную базу для создания приложений с полноценной графической оболочкой. Это классы Application (Приложение), Document (Документ), App UI (Прикладной пользовательский интерфейс) и класс view (Вид).

### 7.2.1. Класс Application

Класс Application определяет основные свойства всего приложения, предоставляя некую стартовую площадку для начала работы всей программы в целом. Он запускает объекты и идентифицирует приложение по уникальному идентификатору (UID3). Класс Application создает новый Document приложения. В системном уровне Uikon классом Application является класс CEikApplication, а для платформ UIQ и серии 60 соответственно классы CQikApplication и CAknApplication, наследующие возможности системного класса CEikApplication.

## 7.2.2. Класс Document

Класс Document содержит различные данные приложения и предоставляет эти данные приложению, создавая тем самым четко отработанную структуру для работы с данными программы. Класс Document так же создает пользовательский интерфейс приложения App UI. В Uikon - это системный класс CEikDokument, в UIQ, (Qikon) - класс CQikDokument. Для серии 60 (Avkon) определен класс CAknDokument.

## 7.2.3. Класс App UI

Созданный классом Document, класс App UI является абсолютно независимым и создает базовые элементы пользовательского интерфейса, то есть предоставляет минимальные элементы для создания полноценной графической оболочки приложения и обработки пользовательских команд. Так же класс App UI создает App view. В Symbian OS на уровне Uikon классом пользовательского интерфейса является CEikAppUi, в UIQ- класс CQikAppUi, а в серии 60 класс CAvkAppUi.

## 7.2.4. Класс App View

Класс App View необходим для контроля за графическим контекстом, представленным на экране телефона, делая возможным взаимодействие с данными на экране телефона. Класс App view в серии 60 играет роль адаптера для графической оболочки приложения и при этом дает дополнительные возможности для обработки событий, полученных с клавиш телефона.

Таким образом, изготовив каркас программы с помощью рассмотренных классов, вы создадите базовую среду для дальнейшего построения приложения, в котором необходимо будет реализовывать различные элементы пользовательского интерфейса, обработку команд, представление графики на экране, работу со звуком и видео.

Что же касается создания каркаса программы, то с этим очень хорошо справляются среды программирования, рассмотренные в *главах 2 и 3*. Став более опытным программистом в вопросах, связанных с Symbian OS, создайте свой шаблон приложения, описывающий GUI (Графический интерфейс пользователя), и используйте его в дальнейшем в своих программах. После создания GUI-приложения для Symbian OS, необходимо создать дополнительные классы, решающие необходимый круг задач и интегрировать их в приложение. На рис. 7.2 схематично изображена архитектура приложения.

При этом вы естественно можете писать программный код и в исходных файлах, составляющих каркас графической оболочки приложения, но это значительно усложнит задачу, да и сами вы в итоге обязательно запутаетесь, создавая свои классы. Поэтому лучше придерживаться модели создания программ, показанной на рис. 7.2, тем более что язык программирования C++ тем и хорош,

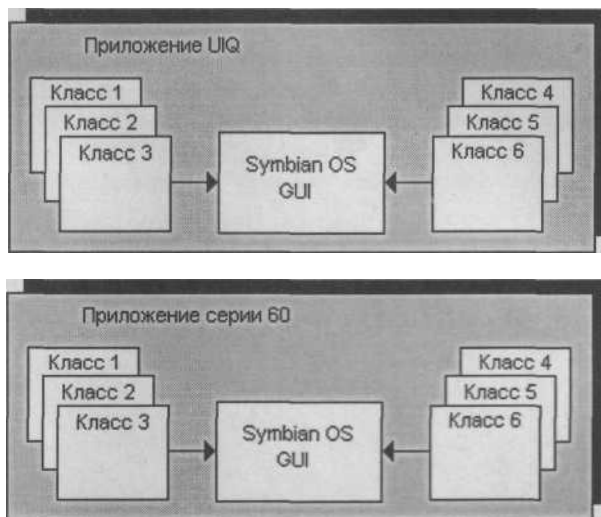


Рис. 7.2. Архитектура приложения

что является объектно-ориентированным языком, упрощающим процесс создания программ.

### 7.3. Первая программа

Теперь давайте перейдем от теоретической части изучения структуры приложения к ее практической реализации и непосредственно к изучению исходного кода демонстрационной программы. В качестве примера была выбрана платформа серии 60. Модель работы для UIQ идентична серии 60, имеются различия только в пользовательском интерфейсе, однако общая схема создания GUI приложения аналогична для обеих платформ.

Чтобы сделать графическую оболочку для программы, необходимо создать свои собственные классы производные от классов Application, Document, App Ui и App View. Формируя новый проект в среде программирования Metrowerks CodeWarrior и C++ BuilderX Mobile Edition, вы получаете такие классы автоматически, они и составляют общий каркас GUI-приложения с минимальной базой для дальнейшего развития программы. На компакт-диске в папке \Code\Test находится проект Test, на примере которого мы изучим составляющую приложения для серии 60. Посмотрите внимательно на рис. 7.3, где показана модель взаимодействия классов приложения Test с системными классами Symbian OS.

Программа Test имеет три класса CTestApplication, CTestDocument и CTestAppUi, производных от системных классов Avkon и один класс CTestAppView, наследующий свойства от класса CCoeControl. Создав и реализовав эти четыре класса, вы получите общий каркас базовой составляющей GUI-приложения.

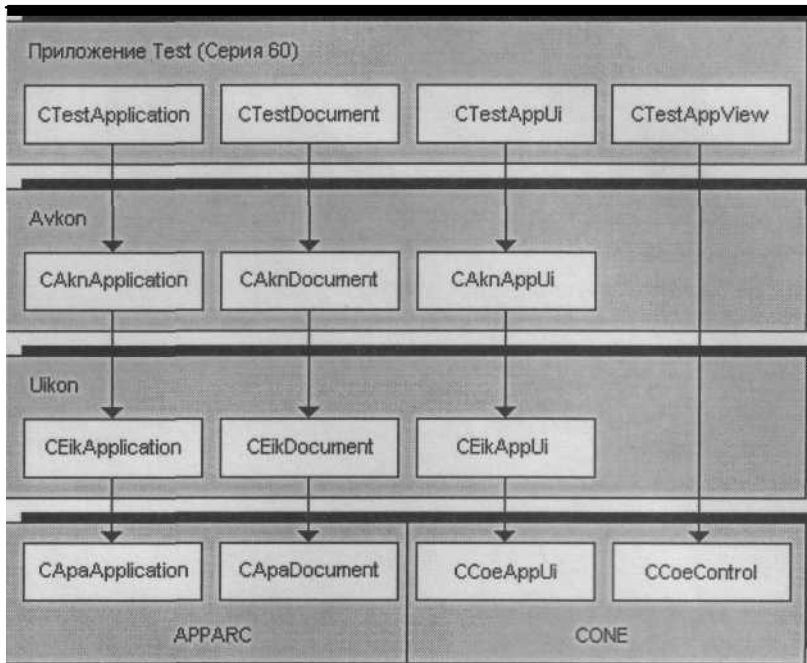
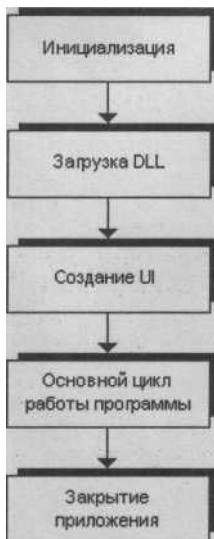


Рис. 7.3. Взаимодействие классов приложения Test

Прежде чем мы перейдем к исходному коду демонстрационного примера Test, необходимо сказать еще несколько слов о работе системы в момент функционирования программы.

### 7.3.1. Работа системы



В тот момент, когда на рабочем столе телефона выбрана иконка приложения, то есть она попадает в фокус курсора, и нажимается клавиша телефона для запуска приложения, на уровне ядра происходят мгновенные системные вызовы соответствующих библиотек и DLL. На рис. 7.4 показана схема системных вызовов в Symbian OS в момент запуска программы.

Как только было запущено приложение, создается новый системный процесс с вызовом функции E32Main() - это входная точка для всех новых процессов системы. Далее происходит процесс инициализации каркаса приложения с созданием нового потока, запуск высокоуровневого менеджмента обработки ошибок, связь с оконным сервером (Window Server) и инициализация ресурсов программы.

Рис. 7.4. Схема системных вызовов Symbian OS

На этапе загрузки DLL происходит создание приложения и бланка документа, затем идет создание интерфейса пользователя (Ш) приложения и управление передается самой программе. Цикл работы программы зависит от ее функционального назначения, но как только пользователь решит выйти из программы, выбрав команду выхода, работа системы переходит к этапу закрытия программы. На системном уровне в момент закрытия программы происходит закрытие всех активных библиотек, удаление из памяти загруженных ресурсов, закрытие сессии Window Server и уничтожение активного процесса, созданного для этой программы. Схема, конечно, несколько упрощена и на самом деле происходят вызовы около полусотни системных функций Symbian OS. Но это уже тема для отдельного разговора и область системных программистов, поэтому давайте перейдем к рассмотрению программы Test.

### **7.3.2. Класс CTestApplication**

Определение класса в языке программирования C++ обычно состоит из двух файлов. Первый файл - заголовочный с расширением \*.h, где задана спецификация класса, то есть объявляется класс, перечисляются данные члены класса, а это конструкторы, деструкторы, функции и переменные. И второй файл - это файл реализации класса с расширением \*.cpp, где соответствующим образом описываются объявленные члены класса в заголовочном файле. Такой подход в объектно-ориентированном программировании считается лучшим и именно по этой схеме рекомендуется работать с классами в Symbian OS.

Класс CTestApplication необходим для создания и инициализации всего приложения. Спецификация класса находится в файле Test\_Application.h, расположенном на компакт-диске в папке \Code\Test\inc\Test\_Application.h. Для написания программ под Symbian OS создана очень удобная система расположения исходных файлов в папках, находящихся в каталоге проекта. Могут использоваться следующие названия папок:

- aif - эта папка предназначена для хранения иконки приложения (\*.bmp) и файла \*.aif описывающего использование AIF ресурсов программы;
- data содержит файлы ресурсов (\*.rss) приложения;
- group - в этой папке хранятся проектные файлы bld.inf, \*.pkg и \*.mmp, а так же среды программирования располагают там свои проектные файлы;
- inc - здесь хранятся заголовочные файлы проекта \*.h, \*.hrh, \*.pan;
- install или sis может использоваться одна из папок для файлов \*.pkg и \*.sis;
- src — папка для файлов исходного кода (\*.cpp) с реализацией объявленных классов.

Это общепринятая структура проекта, правда, не всегда соблюдаемая. Кто-то, например, выбирает меньшее количество папок, обычно это касается ресурсов, но так или иначе лучше придерживаться этой схемы проекта. При упаковке проекта в SIS-архив лучше, чтобы файл PKG находился все-таки в папке \group.

Заголовочный файл CTest\_Application.h содержит спецификацию класса CTestApplication. Давайте рассмотрим исходный код этого класса.

```
y/***** //
заголовочный файл Test_Application.h
// спецификация класса CTestApplication
//*****
// проверяем лексему #ifndef
_Test_APPLICATION_H_ #define
_Test_APPLICATION_H_

// подключаем системную библиотеку #include
<aknapp.h>

// объявляем класс CTestApplication
class CTestApplication : public CAknApplication
{

    public:

        TUid AppDllUidO const;

    protected:

        CApaDocument* CreateDocumentL();

#endif // Test APPLICATION_H
/y*****
```

В первых двух строках исходного кода файла CTest\_Application.h происходит проверка: определена ли данная лексема в программе или нет. За директивами препроцессора #ifndef и #define в конце исходного кода всегда должна следовать директива tendif. Затем в коде происходит подключение системной библиотеки aknapp.h, где объявлен класс CAknApplication (класс приложения) системного уровня Avkon для серии 60. Далее происходит объявление класса: class CTestApplication : public CAknApplication

С помощью ключевого слова class декларируется спецификация класса CTestApplication и оператор «:» указывает на открытое наследование от класса CAknApplication, предоставляя тем самым классу CTestApplication все возможности системного класса уровня Avkon. В самом конце исходного файла происходит объявление двух функций: TUid: :AppDllUidO и CreateDocumentL() класса CApaDocument.

Теперь давайте перейдем к файлу CTest\_Application.cpp, где дана реализация объявленного класса CTestApplication. Исходный файл так же можно найти на компакт-диске в папке \Code\Test\src\Test\_Application.cpp.

```

/у*****
// файл Test_Application.cpp
// реализация класса CTestApplication
/у*****
// подключаем заголовочные файлы
#include "Test_Application.h"
#include "Test_Document.h"

// константа со значением уникального идентификатора
const TUid KUidTest = { 0x10000000 };
// создаем бланк документа
CAppDocument* CTestApplication::CreateDocumentL ()
{
    CAppDocument* document = CTestDocument::NewL (*this);
    return document;
}
// функция возвращает значение уникального идентификатора
приложения
// уникальный идентификатор приложения определен в файле MMP
TUid CTestApplication::AppDllUid() const
{
    return KUidTest;
}

```

В строке исходного кода файла `CTest_Application.cpp` `const TUid KUidTest = { 0x10000000 };` происходит объявление константы и присвоение ей значения уникального идентификатора (UID3) приложения. В Symbian OS уникальные идентификаторы - это ключевое понятие и довольно сложное по своей спецификации, поэтому эта тема вынесена в отдельный раздел главы 7.4. (*Уникальные идентификаторы*). А пока просто запомните, что это уникальный идентификатор приложения и обозначается он как UID3, то есть в приложении имеются еще как минимум два UID.

В спецификации класса `CTestApplication` были объявлены еще две функции. С помощью функции `CreateDocumentL ()` создается бланк документа. Оператор « : » очень часто используется в программировании под Symbian OS и определяет пространство имен. Вторая функция `AppDllUid ()` идентифицирует приложение и возвращает уникальный идентификатор приложения (UID3). Как видите, класс `CTestApplication` предназначен для инициализации приложения и создания бланка документа (`Document`).

### **7.5.3. Класс *CTestDocument***

Класс `CTestDocument` создает общий механизм для работы с прикладными данными программы - это работа с файлами, операции по сохранению и чтению

пользовательских данных. Обращение к памяти происходит при помощи потока. *Поток* - это последовательность двоичных данных. Кроме того, класс CTestDocument создает AppUi, применяя для этого функцию CreateAppUiL (). Спецификация класса CTestDocument находится на компакт-диске в папке \Code\Test\inc\Test\_Document.h, рассмотрим этот заголовочный файл.

---

```
// заголовочный файл Test_Document.h
// спецификация класса CTestDocument
//*****

// проверяем лексему tifndef
_Test_DOCUMENT_H_ #define
_Test_DOCUMENT_H_

// подключаем системную библиотеку tinclude
<akndoc.h>

// ссылки на системные классы и классы приложения class
CTestAppUi; class CEikApplication;

// объявляем класс CTestDocument
class CTestDocument : public CAknDocument

public:
    // двухфазный конструктор
    static CTestDocument* NewL(CEikApplicationS aApp);
    static CTestDocument* NewLC(CEikApplications aApp);
    // деструктор
    ~CTestDocument();
    CEikAppUi* CreateAppUiL();
private:
    void ConstructL();
    // конструктор
    CTestDocument(CEikApplication& aApp);

#endif // _Test_DOCUMENT_H_
```

В начале исходного кода следует стандартное определение лексемы и подключение системной библиотеки akndoc.h для работы с системным классом CAknDocument. Затем в исходном коде обозначаются две ссылки на системный класс Symbian OS и класс приложения.



```
class CTestAppUi; class  
CEikApplication;
```

Эти объявления, ссылающиеся на классы, необходимы в процессе компиляции программы, иначе возникнут ошибки. Далее идет объявление класса CTestDocument, который наследует возможности системного класса CAknDocument.

В спецификации класса CTestDocument используется стандартный *двухфазный* конструктор, работающий на основе функций NewL (), NewLC () и ConstructL (). При формировании класса в Symbian OS почти всегда используется такая конструкция исходного кода для создания класса. В Symbian OS очень важно следить за объектами, предотвращая утечку памяти. Если объект уже удален, а ссылка на него осталась в стеке, происходит сбой в системе и немедленное закрытие программы. Системные функции NewL (), NewLC () и ConstructL () создают объект класса и осуществляют контроль над объектом, находящемся в стеке, выполняя роль так называемого двухфазного конструктора.

Так же декларируется простой конструктор класса CTestDocument () со ссылкой на объект класса CEikApplication. В классе CTestDocument определен деструктор и функция CreateAppUiL () для создания AppUi.

Теперь перейдем к файлу реализации класса CTestDocument, который находится на компакт-диске в папке \Code\Test\src\Test\_Document.cpp. Давайте посмотрим на исходный код и разберемся с его составляющей.

```
/******* * * *****/  
// файл Test_Document.cpp  
// реализация класса CTestDocument.h  
/*******  
// подключаем заголовочные файлы  
◆include "Test_Document.h"  
◆include "Test_AppUi.h"  
// двухфазный конструктор  
CTestDocument* CTestDocument::NewL(CEikApplicationS aApp)  
{  
    CTestDocument* self = NewLC(aApp);  
    CleanupStack::Pop(self);  
    return self; }  
// двухфазный конструктор  
CTestDocument* CTestDocument::NewLC(CEikApplicationS aApp)  
{  
    CTestDocument* self = new (ELeave) CTestDocument(aApp);  
    CleanupStack::PushL(self);  
    self->ConstructL ();  
    return self;
```

```

// двухфазный конструктор
void CTestDocument::ConstructL()

// простой конструктор без реализации
CTestDocument::CTestDocument(CEikApplicationS aApp)
: CAknDocument(aApp)

// деструктор
CTestDocument::~CTestDocument()

// создает интерфейс пользователя
CEikAppUi* CTestDocument::CreateAppUiL() {
    return new (ELeave) CTestAppUi; }

```

В начале исходного кода файла `Test_Document.cpp` происходит подключение заголовочных файлов приложения `Test`. Затем описываются двухфазные конструкторы, использующие функции `NewL()`, `NewLC()` и `ConstructL()` для работы со стеком. Деструктор и простой конструктор `CTestDocument()` пока остаются без реализации. Функция `CreateAppUiL()` способствует созданию каркаса интерфейса пользователя посредством класса `CTestAppUi`. Класс `CTestDocument` может использоваться и для сохранения и чтения данных приложения, применяя функции `StoreLO` и `RestoreLO`, которые нужно объявить в заголовочном файле и описать в файле реализации.

### **7.3.4. Класс *CTestAppUi***

Класс `CTestAppUi` завершает создание каркаса интерфейса пользователя и является получателем различных уведомлений, которые инициализируются структурой приложения, обрабатывая тем самым события, полученные от пользователя.

Заголовочный файл `Test AppUi.h` содержит объявления класса `CTestAppUi` и находится на компакт-диске к книге в папке `\Code\Test\inc\Test_AppUi.h`. Рассмотрим исходный код этого файла.

```

/у*****
// заголовочный файл Test_AppUi.h //
спецификация класса CTestAppUi

```

```

// проверяем лексему
#ifndef Test_APPUI_H
#define _Test_APPUI_H_

// подключаем системную библиотеку
#include <aknappui.h>

// ссылка на класс
class CTestAppView;

// объявляем класс CTestAppUi
class CTestAppUi : public CAknAppUi
{
public:

    void ConstructL0 ;
    // конструктор
    CTestAppUi () ; //
    деструктор -
    CTestAppUi ();

public:
    // события
    void HandleCommandL(Tint aCommand);

private:

    CTestAppView* iAppView;
};

#endif // _Test_APPUI_H_

```

В третьей строке кода происходит подключение системной библиотеки aknappui.h для работы с классом CAknAppUi. В объявлении класса CTestAppUi используется оператор «:», указывающий на наследование классом CTestAppUi всех возможностей системного класса CAknAppUi уровня Avkon серии 60. В классе происходит объявление двух конструкторов, деструктора и функции HandleCommandL () для обработки пользовательских событий. Также создается указатель на объект класса CTestAppView,

Перейдем к файлу Test\_AppUi.cpp, который находится на компакт-диске в папке \Code\Test\src\Test\_AppUi.cpp и содержит реализацию класса CTestAppUi.

```

/y*****
// файл Test_AppUi.cpp
// реализация класса CTestAppUi

// подключаем системные библиотеки
#include <aknotewrappers.h>
#include <avkon.hrh>
// подключаем файл ресурса
◆include <Test.rsg>
// подключаем заголовочные файлы
◆include "Test AppUi.h"
◆include "Test AppView.h"
◆include "Test.hrh"
◆include "Test.pan"
// конструктор .
void CTestAppUi::ConstructL()
{
// основной конструктор завершающий создание UI
BaseConstructL() ; // создаем клиентскую область
iAppView = CTestAppView::NewL(ClientRect()); //
добавляем в стек AddToStackL(iAppView); }
// конструктор
CTestAppUi::CTestAppUi()
{ }
// деструктор CTestAppUi::~-
CTestAppUi() {
    if (iAppView) {
        // удаляем из стека
        iEikonEnv->RemoveFromStack(iAppView); //
        удаляем объект delete iAppView; //
        обнуляем iAppView = NULL;

// обработка команд
void CTestAppUi::HandleCommandL(Tint aCommand)

```

```

switch (aCommand)
{
// ВЫХОД
case EEikCmdExit:
case EAknSoftkeyExit:
    ExitO ;
break; //
паника
default:
    User::Panic ( L("Test"), ETestBasicUi);
break;

// *****

```

Первоначально в исходном коде файла Test\_AppUi.cpp происходит подключение системных библиотек. Avkon.hrh — это своеобразный заголовочный системный файл уровня Avkon для всех ресурсов программы, работающих с меню, диалогами и списками. Далее происходит подключение заголовочных файлов создаваемого приложения. В файле Test.hrh содержится спецификация ресурсов нашей программы, а в файле Test.pan определен исходный код для обработки внештатных ситуаций. В Symbian OS существует специальный термин Panic (*Паника*), его мы и будем придерживаться. Содержание этих двух заголовочных файлов будет рассмотрено в этой главе позже.

Функция ConstructLO, играющая роль конструктора, состоит из трех строк исходного кода формирующего каркас UI. В первой строке кода происходит вызов системного конструктора BaseConstructL (), создающего Ш. Затем с помощью функции ClientRect () создается или определяется клиентская область экрана телефона. Полученное значение сохраняется в объекте iAppView, который в свою очередь посредством функции AddToStackL () добавляется в стек.

*Деструктор* класса CTestAppUi содержит полный цикл по удалению активного объекта из стека

```

iEikonEnv->RemoveFromStack(iAppView);
delete iAppView; и обязательно обнуляется.
iAppView = NULL;

```

Не забывайте после удаления объекта явно присваивать ему значение NULL.

Функция HandleCommandL () реализует обработку событий, полученных с подэкранных клавиш телефона. С помощью оператора switch происходит обработка имеющихся вариантов действий пользователя. На этом этапе задействована единственная команда **Exit**, расположенная под правой подэкранной клавишей телефона. В телефоне имеются две клавиши, расположенные под экраном телефона с левой и с правой стороны. Это две программно определенные клавиши (Soft key), необходимые для удобства работы с меню, диалогами, списками.

В функции `HandleCommandL ()` команда `EEikCmdExit` создает команду **Exit** на подэкранной клавише, а команда `EAKnSoftKeyExit` задает программную клавишу для использования команды **Exit** путем вызова функции `Exit ()`, которая на системном уровне включает механизм корректного завершения программы.

При работе с командами, когда применяется конструкция операторов `case`, надо не забывать использовать ключевое слово `break`, иначе после выполнения определенных действий, прописанных за оператором `case`, управление будет передано следующему оператору `case`. Оператор `default` обрабатывает любые другие действия, не описанные в функции `HandleCommandL ()`. В этом случае используется функция `User: :Panic ()`, отслеживающая внештатные ситуации или панику, связанную с событиями, полученными от пользователя, но не описанными в функции `HandleCommandL ()`. Например, если сейчас запустить на эмуляторе программу `Test`, то на двух подэкранных клавишах вы увидите две команды: слева - **Options**, а справа - **Exit**. Команда **Exit** была обработана в приложении, при ее выполнении последует выход из программы. А вот команда **Options** нами пока не использовалась, но при ее выполнении должно что-то произойти, потому что команда эта есть и соответственно ее можно выполнить, нажав подэкранную клавишу. И вот для того, чтобы не произошло ошибки или внештатной ситуации, программа «не запаниковала» (если так можно выразиться), используется функция `User: :Panic ()`. Эта функция имеет два параметра. Первый — это текстовое сообщение, которое будет представлено в уведомлении, и где, как правило, используется название программы. Вторым параметром в функции `User: :Panic ()` - это перечисление `ETestBasicUi`, определенное в файле `Test.pan` для обработки простейшей паники, связанной с пользовательскими данными. Если вы сейчас запустите программу `Test` и выполните команду **Options**, то в ответ появится небольшое по размеру информационное окно с сообщением о возникшей ошибке, что и является реакцией программы на возникшую внештатную ситуацию.

### 7.3.5. Класс `CTestAppView`

Класс `CTestAppView` определяет представление имеющихся данных приложения относительно экрана телефона, создавая при этом окно и отображая в нем прикладные данные. Спецификация класса `CTestAppView` находится в файле `Test_AppView.h` на компакт-диске в папке `\Code\Test\inc\Test_AppView.h`, рассмотрим исходный код этого класса.

```
// заголовочный файл Test_AppView.h //
спецификация класса CTestAppView
```

```
// проверяем лексему
#ifdef Test_APPVIEW_H
#define Test_APPVIEW_H
```

```

// подключаем системную библиотеку
#include <coectrl.h>

class CTestAppView : public CCoeControl
{
public:
    // двухфазный конструктор
    static CTestAppView* NewL(const TRect& aRect);
    static CTestAppView* NewLC(const TRects aRect);
    // деструктор
    ~CTestAppView();

public:

    void Draw(const TRectS aRect) const;

private:

    void ConstructL(const TRects aRect);
    CTestAppView();

#endif // _Test_APPVIEW_H_

```

В начале исходного кода происходит подключение системной библиотеки coectrl.h для возможности работать с системным классом CCoeControl, потенциал которого наследует класс CTestAppView. При объявлении класса CTestAppView создается стандартная конструкция с применением двухфазного конструктора и деструктора, а также декларируется использование функции Draw () для рисования графики в клиентской области экрана, чтобы нарисовать на экране пару прямоугольников для красоты. Сейчас создается каркас программы, в который затем можно интегрировать графику, шрифт, элементы интерфейса пользователя. Для того, чтобы не смотреть на пустой белый экран, задействуется функция Draw ().

Теперь перейдем к файлу Test\_AppView.cpp, на компакт-диске он находится в папке \Code\Test\src\Test\_AppView.cpp.

---

```

// файл Test_AppView.cpp
// реализация класса CTestAppView
//*****
// подключаем системные библиотеки
#include <coemain.h>

```

```

#include <aknnotewrappers.h>
// подключаем файл ресурса
◆include <Test.rsg>
// подключаем заголовочный файл
◆include "Test AppView.h"
// двухфазный конструктор
CTestAppView* CTestAppView::NewL(const TRects aRect)
{
    CTestAppView* self = CTestAppView::NewLC(aRect);
    CleanupStack::Pop(self) ;
    return self;
}
// двухфазный конструктор
CTestAppView* CTestAppView::NewLC(const TRectS aRect)

    CTestAppView* self = new (ELeave) CTestAppView;
    CleanupStack::PushL(self);
    self->ConstructL(aRect);
    return self;
}
// двухфазный конструктор
void CTestAppView::ConstructL(const TRectS aRect)
{
    // создаем окно
    CreateWindowL() ;
    // определяем клиентскую область
    SetRect(aRect);
    // активизируем механизм прорисовки
    ActivateL(); }
// конструктор
CTestAppView::CTestAppView()

// деструктор
CTestAppView: : ~CTestAppView ()

// рисуем на экране
void CTestAppView::Draw(const TRectS /*aRect*/) const
{
    // получаем графический контекст окна
    CWindowGc& gc = SystemGcO;
    // очищаем клиентскую область окна

```



```

gc.Clear();
// определяем область прорисовки
TRect drawRect = Rect();
// задаем размер прямоугольнику
drawRect.Shrink(10,10);
// рисуем прямоугольник
gc.DrawRect(drawRect);
// задаем размер второму прямоугольнику
drawRect.Shrink(15,30);
// рисуем второй прямоугольник
gc.DrawRect(drawRect);
}
//*****

```

В реализации класса CTestAppView описывается стандартный двухфазный конструктор на основе функций NewL () и NewLC () с использованием стека. В функции ConstructL (), играющей роль конструктора, сначала при помощи функции CreateWindowL () создается окно, затем определяется клиентская область экрана, на которой можно рисовать графику или задействовать различные элементы пользовательского интерфейса. Что же такое клиентская область? Как вы знаете из главы 1, окно или экран телефона разделены на три панели: Status Pane, Main Pane и Control Pane. По умолчанию клиентская область определена как Main Pane, ее мы и используем. Если выразаться языком Windows-программирования - это оконный режим. И в конце функции ConstructL () происходит активизация всех ранее созданных элементов вызовом функции ActivateL().

Функция Draw () рисует два прямоугольника на экране телефона. Первым делом в функции Draw () нам необходимо получить графический контекст окна, чтобы было на чем рисовать графику, шрифты и элементы интерфейса пользователя, для этого задействуется функция SystemGc (). CWindowGc& gc = SystemGc();

Дальше очищаем клиентскую область экрана телефона при помощи функции Clear (), у которой в этом случае нет аргументов, а значит, в качестве цвета очистки используется белый цвет. Затем определяем область прорисовки в строке исходного кода: TRect drawRect = Rect();

В следующей строке кода происходит создание прямоугольника с заданными размерами: drawRect.Shrink(10,10);

Функция Shrink () создает в пространстве область определенного размера и имеет для этого два параметра. Первый - это координата по оси X, второй - координата по оси Y. Отсчет происходит с левого верхнего угла экрана. Откладываются отрезки в 10 пикселей по оси X и в 10 пикселей по оси Y. Начальная точка прорисовки прямоугольника находится в верхнем левом углу. Остальные линии

дорисовываются автоматически относительно левого верхнего угла, создавая тем самым четыре точки с координатами 10 пикселей по каждой оси, равномерно отложенные от каждого угла клиентской области экрана телефона. Чтобы нарисовать заданный прямоугольник вызывается функция DrawRect ().  
gc.DrawRect(drawRect);

Построение и прорисовка второго прямоугольника происходит по той же схеме, но в качестве координаты верхнего левого угла применяется значение в 15 пикселей по оси X и 30 пикселей по оси Y. На экране рисуются два прямоугольника, причем второй прямоугольник рисуется внутри первого.

### 7.3.6. Файл Test\_Main.cpp

В файле TestMain.cpp происходит создание и запуск приложения, которое мы описали в предыдущих разделах. Для конкретно запущенной программы создается свой новый поток, запускается механизм arrgun.exe на системном уровне Eikon и начинается работа всей программы. Исходный код в Test\_Main.cpp находится на компакт-диске в папке \Code\Test\src\Test\_Main.cpp.

```
// файл Test_Main.cpp
// создание и запуск приложения

// подключаем заголовочный файл
#include "Test_Application.h"

// загружаем DLL системы
GLDEF_C Tint E32D11(TDllReason /*aReason*/)
{
    return KErrNone; }
// создаем приложение
EXPORT_C CAppApplication* NewApplication() {
    return (new CTestApplication);
} y/*****
```

В исходном коде имеются две функции. При помощи функции E32D11 () производится загрузка системных динамических библиотек DLL. Если загрузка прошла успешно, возвращается значение KErrNone, обозначающее отсутствие ошибок в системе. И в конце в функции NewApplication () создается и запускается приложение, выполняя тем самым полный цикл системных вызовов, необходимых для работы программы с момента ее запуска и до выполнения команды **Exit** (выход из приложения).

### 7.3.7. Файл Testpan

Программа Test содержит еще несколько важных файлов, один из них - Test.pan. На компакт-диске он находится в папке \Code\Test\inc\Test.pan.

```
y/*****  
// файл Test.pan  
// паника  
//*****  
// проверяем лексему  
#ifndef _Test_PAN_  
◆define _Test_PAN_  
  
// перечисляемый тип  
enum TTestPanics {  
    ETestBasicUi = 1  
  
#endif TestPAN  
/y*****  
****
```

В исходном коде файла Test\_AppUi.cpp в функции HandleCommandL () была задействована конструкция кода для обработки паники, возникающей во время внештатных ситуаций. В частности, использовалась функция Panic () с двумя параметрами. Первый - это строка текста, которая не должна превышать шестнадцать символов. Второй параметр - это 32-битное целое число. Комбинация обоих параметров составляет идентификатор программной ошибки. Во втором параметре использовалась константа ETestBasicUi со значением 1, созданная в перечисляемом типе TTestPanics файла Test.pan. Значение создаваемой константы не должно равняться 0, в данном случае число 1 является неким кодом, идентифицирующим ошибку приложения. В справочной системе Symbian OS в разделе *System panic reference* можно найти полный перечень кодов для работы с внештатными ситуациями. В перечисляемый тип TTestPanics файла Test.pan можно добавлять константы, присваивая им коды (числа от 1 и выше), определенные в категории *System panic* справочной системы Symbian OS для работы с внештатными ситуациями, возникающими в программе.

### 7.3.8. Файл Test.hrh

Файл Test.hrh - это специализированный заголовочный файл, содержащий объявления констант для использования в файлах \*.cpp и файлах ресурсов \*.rss. Это специальные константы, необходимые для работы с меню, диалогами и командами. Файл Test.hrh находится на компакт-диске в папке \Code\Test\inc\Test.hrh, рассмотрим исходный код файла.

```

// заголовочный файл Test.hrh
// содержит спецификацию ресурсов
/y*****
// проверяем лексему
#ifndef Test HRH
#define Test HRH

// перечисляемый тип
enum TTestlds
{
    ETestCommand1 = 1

#endif// Test HRH
/y*****

```

В исходном коде объявляется перечисляемый тип TTestlds с одной константой, которую мы пока в программе не используем, но в дальнейшем при модернизации программы обязательно задействуем.

Что касается исходного кода в файлах \*.rap и \*.hrh, то здесь можно использовать только объявления перечисляемых типов (enum) для определения перечислений, любая другая конструкция исходного кода вызовет ошибку при компиляции.

### 7.3.9. Файл Test\_Caption.rss

*Файлы ресурсов* — это одна из важнейших составляющих в программировании приложений под Symbian OS. С помощью файлов ресурсов осуществляется создание и работа с меню, диалогами, списками и редакторами. Все файлы с расширением \*.rss (Resource Source Script) в программе относятся к файлам ресурсов. Файл Test\_Caption.rss определяет в приложении название программы, написанное под иконкой приложения на рабочем столе телефона и в самой программе на панели Status Pan.

Файл Test\_Caption.rss находится на компакт-диске в папке \Code\Test\group\Test\_Caption.rss и содержит следующий программный код:

```

// файл Test_caption.rss
// заголовки программы
y/*****

```

```

#include <apcaptionfile.rh>

```

```

RESOURCE CAPTIONDATA

```

```
caption = "Test";
shortcaption = "Test";
```

В начале исходного кода происходит подключение системной библиотеки `arcaptionfile.rh`, содержащей большое количество перечисляемых констант. Две из них и были задействованы в файле `Test_Caption.rss`.

Константа `caption` содержит название программы, которое отображается на экране телефона под иконкой. Вторая константа `shortcaption` использует меньший текст для удобного использования названия программы, например, в списке. Структура `RESOURCE CAPTION_DATA`, где определены константы, является стандартной структурой, создающей заголовок приложения. Файл ресурсов `CAPTION` (в нашей программе это файл `Test_Caption.rss`) используется опционально, по желанию программиста, и аналогичные функции можно возложить на информационные файлы ресурсов программы `AIE`

### **7.3.10. Файл *Testrss***

Это, пожалуй, самый главный файл ресурса, необходимый для создания меню, диалогов и списков. В программе `Test` мы используем этот вид ресурсов не на полную мощность, но уже в следующей главе усложним программный код. Файл `Test.rss` можно найти на компакт-диске в папке `\Code\Test\group\Test.rss`, рассмотрим исходный код файла ресурсов программы `Test`.

```
//*****
// файл Test.rss
// ресурсы программы
//**** *****
// имя ресурса (ID)
NAME SETS

// подключаем системные библиотеки
#include <eikon.rh>
◆include <avkon.rh>
◆include <avkon.rsg>
// подключаем заголовочный файл
◆include "Test.hrh"

//*****
// сигнатура
//*****

RESOURCE RSS SIGNATURE
```

```

/y*****
// имя документа

RESOURCE TBUF r default document name {
    buf="";
}
y/*****
// команды

RESOURCE EIK_APP_INFO { cba =
    R_AVKON_SOFTKEYS_OPTIONS_EXIT;
}
y/*****

```

В первой строке исходного кода файла Test.rss определяется имя ресурса: NAME SETS

Ключевое слово NAME определяет имя для ресурса из четырех заглавных букв в файле ресурса. *Имя ресурса* - это идентификатор, определяющий принадлежность файла ресурса к этой программе. Имя состоит из четырех произвольно заданных букв и для каждого приложения лучше использовать новое имя.

Затем происходит подключение системных библиотек и файла Test.hrh для работы с ресурсами. Все дальнейшие объявления ресурсов применяют следующий синтаксис: RESOURCE ИМЯ\_СТРУКТУРЫ

Ключевое слово RESOURCE объявляет тип структуры и дальше следует название создаваемой структуры. В названиях структур ресурсов применяется множество ключевых слов для стандартных различных определений и дополнительно можно создавать свои структуры ресурсов. В системной библиотеке в файле avkon.th объявлены все структуры и ключевые слова для работы с ресурсами в серии 60. В файле Test.rss задействованы пока три названия:

- RSS\_SIGNATURE - это сигнатура, регулирующая в приложении неправильные подключения файлов ресурсов, панику, загрузку других бинарных ресурсов. Но эта структура игнорируется, просто определите ее и оставьте пустой;
- TBUF - этот вид ресурса задает имя для документа приложения. Исползуется по усмотрению, но он обязательно должен быть объявлен;

- EIK\_APP\_INFO - информационный ресурс приложения (Application information resource).

Все три объявленные ресурса в исходном коде программы должны присутствовать обязательно, вне зависимости от того используются они или нет. Подробнее с файлами ресурсов вы познакомитесь в *главе 8*.

### 7.3.11. Файл *bld.inf*

Файл *bld.inf* создается автоматически при формировании проекта и включает в себя всего две строки кода: `PRJ_MMPFILES Test.mmp`

Ключевое слово `PRJ_MMPFILES` типичное для всех проектов - это ссылка на проектный файл `*.mmp`, название которого прописывается во второй строке кода.

Файл *bld.inf* необходим для процесса сборки и компоновки приложения, как указатель на соответствующие проектные данные. Импорт проекта в среду `C++ BuilderX` происходит через файл *bld.inf*. В среде программирования `Metrowerks CodeWarrior` импорт проекта осуществляется через `*.mmp` файл, но наличие файла *bld.inf* обязательно.

### 7.3.12. Файл *Test.mmp*

Проектный файл с расширением `*.mmp` содержит полное описание свойств проекта и является одним из ключевых файлов программы. Поговорка «Что объявишь, то и будешь иметь» как нельзя лучше подходит к файлам `MMP`. В демонстрационном примере `Test` файл *Test.mmp* находится на компакт-диске в папке `\Code\Test\group\Test.mmp`, рассмотрим структуру файла.

```
/*.....  
// файл Test.mmp  
y/*****  
  
TARGET                Test.app  
TARGETTYPE            app  
  
UID                   0xЮ003ЭСЕ  0x10000000  
  
TARGETPATH            \system\apps\test  
  
LANG                  01  
  
SOURCEPATH            ..\src  
SOURCE                Test Main.cpp  
SOURCE                Test Application.cpp  
SOURCE                Test_AppView.cpp
```

```

SOURCE          Test AppUi.cpp
SOURCE          Test Document.cpp

SOURCEPATH      ..\group

RESOURCE        Test.rss
RESOURCE        Test caption.rss

USERINCLUDE     ..\inc

SYSTEMINCLUDE   \epoc32\include

LIBRARY         euser.lib
LIBRARY         apparc.lib
LIBRARY         cone.lib
LIBRARY         eikcore.lib
LIBRARY         avkon.lib
LIBRARY         eikcoctl.lib
/y*****

```

Синтаксис и семантика описания проекта в файле \*.mmp строго predetermined и обычно написание осуществляется строго по ранжиру. В программе Test задействованы не все возможные атрибуты. Поэтому предлагаю рассмотреть все существующие атрибуты, попутно разбирая присутствующие в файле Test.mmp.

Проектный файл \*.mmp состоит из двух столбцов. С левой стороны располагаются ключевые слова, обозначающие определенный атрибут, а с права заданы значения для атрибутов проекта.

- TARGET - это имя программы, устанавливающее вид создаваемой программ: \*.app, \*.dll или \*.exe. В файле Test.mmp назначено имя Test. app — это стандартная программа для Symbian OS.
- TARGETTYPE - тип создаваемой программы: \*.app, \*.dll, \*.exe (должен совпадать со значением в TARGET). Опционально может использоваться уникальный идентификатор (UID1).
- UID - уникальный идентификатор GUI Symbian OS (UID2) и уникальный идентификатор приложения (UID3). Подробно об уникальных идентификаторах мы поговорим в разделе 7.4. *Уникальный идентификатор*.
- TARGET PATH - здесь определяется место расположения программы в момент инсталляции ее на телефон. В файле Test.mmp задана директория \system\apps\test без указания диска (C: или E:). В этом случае пользователь может сам выбирать, на какой диск устанавливать программу, но по указанному адресу. Операционная система Symbian инсталлирует программы в папку \system\apps.
- LANG - это язык, используемый в приложении. Для обозначения используются заглавные буквы или код из цифр, обозначающий выбранный язык. В главе 8 вы найдете подробное описание этой темы. Язык по умолчанию



чанию - английский. В этом случае можно вообще не использовать данный атрибут.

- SOURCEPATH - местонахождение или путь к файлам исходного кода, здесь указывается папка с исходными кодами проекта.
- SOURCE - полное имя файлов исходного кода проекта с расширением \*.cpp. Обязательно необходимо указывать все имеющиеся файлы проекта.
- RESOURCE - перечисляются имена всех файлов ресурсов с расширением \*.rss и \*\_caption.rss.
- UERINCLUDE - папка с подключаемыми заголовочными файлами проекта. В программе Test это .. \inc.
- Q SYSTEMINCLUDE - директория на компьютере, содержащая библиотечные и заголовочные файлы Symbian OS, это всегда \epoc32\include.
- LIBRARY - подключаемые вами в приложение библиотечные файлы. При работе с системными классами Symbian OS вы должны подключить заголовочный файл. Делается это в начале исходного кода, и многие классы требуют подключения соответствующей системной библиотеки. В атрибуте LIBRARY и прописываются все без исключения библиотеки, задействованные вами в программе.
- AIF — это информационный файл приложения (Application Information File), содержащий иконку программы и файл ресурса (\*.aif), который описывает использование информационных ресурсов программы. В конце главы мы задействуем этот атрибут и добавим иконку в приложение.

Это полное описание атрибутов файла \*.mmp. В программе Test мы задействовали почти все атрибуты, а в конце этой главы и в начале следующей используем оставшиеся атрибуты \*.mmp файла. При импорте проекта в любой из сред программирования, на основе указанных данных в файле MMP создается общая структура проекта, принимающая непосредственное участие в компиляции, сборке, отладке, упаковке всей программы. Поэтому важно прописывать все данные проекта в файле MMP до его импорта в одну из сред программирования. В CodeWarrior на основе файла MMP создается свой проектный файл с расширением MCP, но это точная копия атрибутов файла MMP, оптимизированная для работы с CodeWarrior.

### **7.3.13. Файл Test.pkg**

В тестировании программ на эмуляторах файл \*.pkg не участвует. Этот файл задействуется системой при создании установочного пакета (SIS) для переноса на телефон, и на основе описаний \*.pkg файла создается архив SIS.

Синтаксис описания \*.pkg файла несложен. Давайте рассмотрим файл Test.pkg, который находится на компакт-диске в папке \Code\Test\sis\Test.pkg.

```
; файл Test.pkg  
; необходим для создания установочного пакета SIS
```

```

; название, UID3, версия (major, minor, build), тип упаковки
#{ "Test" }, (0x10000000), 1,0,0, TYPE=SISAPP

; серия 60 версия 2.0
(0x101F7960), 0, 0, 0, {"Series60ProductID"}

; пути к откомпилированным файлам программы
"..\\..\\..\\epoc32\\release\\armi\\urel\\Test.app" -
"!:\\system\\apps\\Test\\Test.app"

".\\Л.Л.\\epoc32\\data\\z\\system\\apps\\Test\\Test.r01"-"!:\\system\\apps\\Test\\Test.rsc"

".\\Л.Л.\\epoc32\\data\\z\\system\\apps\\Test\\Test_caption.r01"-
"!:\\system\\apps\\Test\\Test_caption.rsc"

```

В коде можно применять ассемблерный вид комментариев, которые следуют после точки с запятой. Компилятором они будут игнорироваться.

В строке кода

```

#{ "Test" }, (0x10000000), 1,0,0, TYPE=SISAPP

```

указаны имя приложения, заключенное в фигурные скобки, уникальный идентификатор приложения (UID3), номер версии программы (major, minor и build) и тип создаваемого установочного пакета. Затем прописывается строка кода, указывающая на платформу, под которую создается программа: (0x101F7960), 0, 0, 0, {"Series60ProductID"}

Это уникальный идентификатор платформы (Platform UID), подробности смотрите в этой главе, в *разделе 7.4*.

И последние шесть строк исходного кода файла Test.pkg определяют пути к откомпилированным и готовым компонентам программы.

```

"..\\..\\..\\epoc32\\release\\armi\\urel\\Test.app" -
"!:\\system\\apps\\Test\\Test.app"

".\\Л.Л.\\epoc32\\data\\z\\system\\apps\\Test\\Test.r01"-"!
:\\system\\apps\\Test\\Test.rsc"

".\\Л.Л.\\epoc32\\data\\z\\system\\apps\\Test\\Test_caption.r01"-
"!:\\system\\apps\\Test\\Test_caption.rsc"

```

Перед созданием SIS-архива вы обязаны откомпилировать программу под платформу ARMI UREL или THUMB UREL в зависимости от модели телефона. После компиляции исходных кодов и ресурсов приложения в каталоге, где установлен используемый SDK, появятся файлы с расширением \*.app, \*.rsc и \*\_caption.rsc. Это готовые компоненты программы, которые можно упаковывать в SIS-архив. Файл Test.app расположен в папке:

```
" . . \ . . \ . . \epoc32\release\armi\urel\Test.app"
```

Вторая строка с восклицательным знаком "

```
!\system\apps\Test\Test.app"
```

указывает на то, что во время инсталляции пользователем программы на телефон, он может ее установить на любой диск своей файловой системы, но в каталог \system\apps\Test\Test.app. Ресурсы приложения находятся в других папках, как видно из исходного кода файла Test.pkg, поэтому прописывается следующий путь:

```
".\..\..\epoc32\data\z\system\apps\Test\Test.r01"-"!\system\apps\Test\Test.rsc"
```

```
".\..\..\epoc32\data\z\system\apps\Test\Test_caption.r01"-
```

```
!\system\apps\Test\Test_caption.rsc"
```

Если вы используете в файле MMP своего проекта декларацию языка LANG, то файлы ресурсов после компиляции будут иметь расширение \*.r01 (для английского языка). При использовании других языков соответственно будет изменяться и расширение откомпилированного файла ресурса, например, для русского языка это \*.r16. Подробно о локализации приложений рассказывается в следующей главе.

Все установки с описанием путей к готовым файлам программы, использованные в файле Test.pkg, верны только для SDK серии 60 версии 2.x, а в SDK серии 60 версии 1.x необходимо прописывать следующие пути:

```
".\..\..\epoc32\release\armi\urel\Test.app" -w !
```

```
:\system\apps\Test\Test.app"
```

```
".\..\..\epoc32\release\armi\urel\Test.r01"-
```

```
!\system\apps\Test\Test.rsc"
```

```
".\..\..\epoc32\release\armi\urel\Test_caption.r01"-
```

```
!\system\apps\Test\Test_caption.rsc"
```

После компиляции проекта под SDK серии 60 версии 1.x готовые файлы программы сохраняются в этих папках. Как видите, для ресурсов программы путь к готовым элементам приложения в SDK серии 60 версии 1.x уже совсем другой. Поэтому если во время упаковки программы появляется ошибка, указывающая на отсутствие тех или иных файлов готовой программы, то нужно пройти в каталог SDK и убедиться в наличии соответствующих файлов.

Для платформы THUMB откомпилированные файлы находятся в папках для SDK серии 60 версии 2.x:

```
".\..\..\epoc32\release\thumb\urel\Test.app" -"
```

```
!\system\apps\Test\Test.app"
```

```
".\Л.Л..\epoc32\data\z\system\apps\Test\Test.r01" -" !:\system\apps\Test\Test.rsc"
```

```
".\Л.Л..\epoc32\data\z\system\apps\Test\Test_caption.r01" -  
"!\system\apps\Test\Test_caption.rsc"
```

А для SDK серии 60 версии 1.x:

```
".\Л.Л..\epoc32\release\thumb\urel\Test.app" -"  
!\system\apps\Test\Test.app"
```

```
".\Л.Л..\epoc32\release\thumb\urel\Test.r01" -"  
"!\system\apps\Test\Test.rsc"
```

```
".\Л.Л..\epoc32\release\thumb\urel\Test_caption.r01" -"  
"!\system\apps\Test\Test_caption.rsc"
```

При работе со средой Metrowerks CodeWarrior можно настроить проектные пути под свой проект. При формировании или экспорте проекта, среда программирования Metrowerks CodeWarrior в каталоге проекта создает несколько папок, одна из них `НаЗВанМеПрограмМН_Data`. В этой папке имеются дополнительные вложенные папки с названиями платформ `ARMU UREL`, `ARMU UDEB` и так далее для всех платформ. Эти папки предназначены для хранения откомпилированных компонентов программы для каждой из платформ, но настройки по умолчанию в Metrowerks CodeWarrior направляют готовые элементы программы в каталог SDK. Произведя определенные настройки в проекте, можно именно в эти папки перенаправить размещение откомпилированных файлов программы.

Для этого в активном проекте выберете в меню команду **Edit** ⇒ **ARMU UREL Setting** (Alt+F7) или нажмите на инструментальной панели Metrowerks CodeWarrior крайнюю справа кнопку **ARMU UREL Setting**. Откроется диалоговое окно **ARMU UREL Setting**. На панели **Target Setting** этого окна, в поле **Output Directory** (Выходная директория для откомпилированных файлов проекта), укажите путь к папке, например, для программы `Test`, это `C:\Symbian\Code\Test\Test_Data\ARMU UREL`. Точно такой же путь нужно указать на вкладке **Symbian Installation** диалогового окна **ARMU UREL Setting** в поле **Content Search Location** (Поиск файлов проекта).

## 7.4. Уникальные идентификаторы UID

*Уникальный идентификатор* (UID) - это 32-битное уникальное число в формате `0x00000000` или `0x0FFFFFFF`, идентифицирующее определенный компонент, используемый в создании программ для Symbian OS. Уникальных идентификаторов в одной программе может использоваться четыре, пять или более, но основными являются `UID1`, `UID2`, `UID3` и `Platform UID`.

### **7.4.1. Идентификатор UID1**

Этот уникальный идентификатор применяется в приложении опционально и прописывается в файле \*.mmp в атрибуте TARGETTYPE. Как правило, *UID1* употребляется редко и в проектном файле \*.mmp в атрибуте TARGETTYPE проще всего применить англоязычные обозначения для типа создаваемой программы \*.app, \*.dll или \*.exe.

### **7.4.2. Идентификатор UID2**

Уникальный идентификатор *UID2* - это специальный идентификатор для GUI-приложения в Symbian OS, содержащий обобщенное описание каркаса приложения (GUI). Он прописывается в файле \*.mmp в атрибуте UID и обязательно идет первым в этой строке. В программе Test в файле Test.mmp применялся идентификатор со значением 0xЮ0003ЭСЕ - это общий идентификатор для серии 60 и UIQ.

### **7.4.3. Идентификатор UID3**

Уникальный идентификатор *UID3* - идентификатор приложения, это очень важный идентификатор, который необходимо использовать с особой осторожностью! Идентификатор UID3 вы не можете создать сами, его необходимо заказать в компании Symbian Ltd. Для этого надо выслать письмо по электронной почте по адресу uid@symbiandevnet.com с просьбой выслать уникальный идентификатор приложения. В письме нужно обязательно указать свое имя и фамилию, адрес электронной почты и страну, в которой вы проживаете. За один раз можно попросить не более десяти идентификаторов. Не думаю, что на начальной стадии вам столько нужно, поэтому будьте скромнее - одного-двух будет достаточно. Каждая готовая программа, предназначенная для инсталляции на телефон, обязана иметь свой уникальный идентификатор приложения UID3.

*Никогда не используйте одинаковый уникальный идентификатор приложения UID3 в двух и более различных программах!*

*Никогда не используйте старые идентификаторы в своих новых программах!*

*Как только вы создали программу и готовы ее продавать или распространять бесплатно, закажите в Symbian уникальный идентификатор приложения UID3 для своей программы и больше никогда не используйте его в других программах!*

*Если две разные программы с одним уникальным идентификатором UID3 попытаться установить на один и тот же телефон, возникнет системная ошибка, и обе программы никогда больше не будут работать на этом телефоне!*

Однако на этапе создания и тестирования программы на компьютере вы можете использовать свой промежуточный идентификатор. В программе Test мы так и поступили- использовался идентификатор UID3 со значением 0x10000000. В файле Test\_Application.cpp в строке const TUid KUidTest = {0x10000000};

объявлена константа со значением UID3 и в функции AppDI IUID () возвращалось значение UID3. В файле Test.mmp атрибут UID так же содержит UID3 со значением 0x10000000. И, наконец, в файле Test.pkg в строке #{"Test"}, (0x10000000), 1, 0, 0, TYPE = SISAPP; прописывается UID3. Во всех трех перечисленных файлах должен фигурировать один и тот же уникальный идентификатор приложения UID3. Но как только вы подготовили программу, чтобы покорить весь мир, закажите в Symbian Ltd. идентификатор UID3 и замените промежуточный UID3 на новый, специально полученный для программы.

В использовании промежуточных идентификаторов приложения UID3 для тестирования на компьютере тоже есть свои нюансы.

***Нельзя использовать один и тот же идентификатор UID3 в различных приложениях и демонстрационных примерах, которые вы тестируете на компьютере!***

То есть если мы с вами создали программу Test с UID3 со значением 0x10000000, то следующую программу (даже демонстрационную работающую только на компьютере), вы обязаны наградить новым UID3! Дело в том, что в процессе тестирования программы, а именно в момент компиляции и сборки, UID3 идентифицируется в SDK и последующее его использование уже невозможно. То есть две разные программы с одним UID3 отлично компилируются, но работать не будут.

Использование промежуточных идентификаторов приложения UID3 очень удобно. Вы можете контролировать их сами, но как только вы будете готовы распространять программу, вам потребуется действительно уникальный UID3, полученный в компании Symbian Ltd.

## **7.4.5. Идентификаторы платформы**

Уникальный идентификатор платформы {Platform UID} декларируется только в файле приложения \*.pkg и служит для создания корректного SIS-пакета. При использовании идентификатора платформы в момент инсталляции программы телефон идентифицирует программу по Platform UID. Если программа предназначена для этого устройства, то производится установка. Если программа написана для другой платформы, то телефон сообщит пользователю о возникшей проблеме и предложит выйти из процесса установки программы.

В демонстрационном примере Test в файле Test.pkg был использован Platform UID: 0x101A7960- это уникальный идентификатор серии 60 версии 2.0. В табл. 7.3 перечислены уникальные идентификаторы различных платформ.

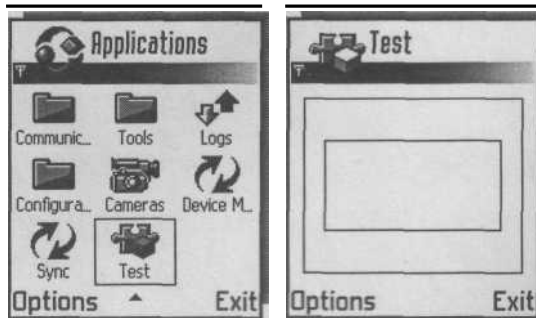
**Таблица 7.3. Уникальные идентификаторы платформ**

<b>Платформы</b>	<b>Идентификаторы</b>
Серия 60 версия 0.9	0x101F6F88
Серия 60 версия 1.0	0x101F795F
Серия 60 версия 1.2	0x101F8202
Серия 60 версия 2.0	0x101F7960
Siemens SX1	0x101F9071

**Таблица 7.3. Уникальные идентификаторы платформы (продолжение)**

Платформы	Идентификаторы	
Nokia 3650 0x101		F7962
Nokia N-Gage	0x101F8A64	
Nokia 6600 0x101F7963		
Nokia 6620 0x1020216B		
Nokia 6630 0x101		F7964
Nokia 7610 0x101		F6F87
Nokia 7650 0x101F617B		
UIQ2.0	0x101FD5DB	

Откомпилировав программу Test и запустив эмулятор, на рабочем столе телефона отобразится иконка и название программы, как это показано на рис. 7.5.



**Рис. 7.5. Работа программы Test**

В своей программе мы не использовали пока иконку, и поэтому сервис телефона добавляет иконку по умолчанию. Любое установленное на телефон приложение добавляется на рабочий стол телефона в конец списка. Перейдя с помощью джойстика или клавиш телефона курсором по списку программ, наведите курсор на иконку программы. Выполните команду **Запуск**, **Выбрать** или **Launch** (в зависимости от модели телефона). Запустится рабочий цикл программы. В программе Test на экране телефона рисуются два прямоугольника, изображенные на рис. 7.5. Для выхода из программы используйте подэкранную клавишу **Exit**.

Таким образом, было создано простое GUI-приложение, выступающее в роли каркаса для всех остальных программ. Теперь рассмотрим добавление иконки в программу.

## **7.5. Добавляем иконку в приложение**

В Symbian OS существует вид ресурсов AIF (Application Information File), отождествляющийся как раз с иконкой и названием программы. Применяя отработанные механизмы Symbian OS, добавим в программу Test иконку. В дальнейшем мы будем модернизировать программу Test.

Прежде чем перейти к проекту, нам понадобятся иконки. Для создания иконок можно использовать любой удобный для вас графический редактор. Стандартный формат Symbian OS для иконок и изображений - это \*.mbm (Multi-Bitmap), но все графические элементы сохраняются в формате Windows - \*.bmp и при компиляции программы конвертируются в формат \*.mbm автоматически.

В Symbian OS для AIF-ресурсов предусмотрено два вида *иконок*: размером 44 x 44 пикселя для рабочего стола телефона и 42x29 пикселей для всевозможных списков и названий программ. Так как в Symbian OS используются два вида дисплеев с различной цветностью (битность) экрана, иконки могут быть двух видов. Цветность в 4096 цветов - это 12-битный цвет ( $2^{12} = 4096$ ) и цветность в 65536 цветов - это 16 бит ( $2^{16} = 65536$ ). Так же в Symbian OS имеется поддержка и 24-битного цвета.

При создании иконки необходимо сделать маску (mask bitmap) для иконки, которая примет непосредственное участие в процессе конвертации \*.bmp файла в \*.mbm формат во время компиляции программы. В программе AifTest на иконке изображена микросхема с надписью «Тест», а маска для иконки — это тот же самый рисунок с микросхемой, содержимое которого (микросхема) просто залито черным цветом. Это и есть маска. Во время конвертации изображения происходит процесс создания файла \*.mbm с использованием исходного изображения и маски.

В программе мы применим две 12-битные иконки: размером 44 x 44 пикселя - для рабочего стола телефона и 42 x 29 пикселей - для отображения в заголовке программы на панели состояния (Status Pane).

Теперь перейдем к исходному коду. Для удобства был создан новый проект. Точнее, старый проект был помещен в папку AifTest, а в исходном коде использовался уже другой промежуточный уникальный идентификатор приложения UID3 (во избежание возможного конфликта). На компакт-диске программа AifTest находится в папке \Code\AifTest.

Создадим в проекте AifTest новую папку, где будут располагаться иконка и маска. В программе AifTest иконка - это файл Test.bmp, а маска - MaskTest.bmp размером 44 x 44 пикселя. А так же иконка для заголовка программы - Test1.bmp и маска - MaskTest1.bmp размером 42 x 29 пикселей. Создадим новый файл ресурса AifTest.rss, исходный код которого вы найдете в папке \Code\AifTest\AifTest.rss, там же находятся и иконки.

```
// файл AifTest.rss

#include <aiftool.rh>

// данные aif ресурса
RESOURCE AIF_DATA
{
    // идентификатор UID3
```



```

app_uid = 0x11000000;
// список заголовков
caption_list =

        CAPTION

        code = ELangEnglish;
        caption = "AifTest";

num icons = 2;
embeddability = KAppNotEmbeddable;
newfile = KAppDoesNotSupportNewFile;
}
//*****

```

В начале исходного кода происходит подключение системного заголовочного файла ресурсов Symbian OS `aiftool.rh`, в котором находятся спецификации всех элементов, использованных в исходном коде `AifTest.rss`.

Ресурс `AIF_DATA` содержит описание данных используемых программой. Переменной `app_uid` присваивается значение нового уникального идентификатора приложения (UID3), который вы так же должны изменить в проекте `AifTest` в файлах `Test.mmp`, `TestApplication.cpp` и `Test.pkg`.

С помощью ресурса `caption_list` объявляется список заголовков для подписочной надписи иконки на рабочем столе и на панели состояния в самом приложении. В этом случае при использовании `AIF`-ресурса необходимость в файле `Test_caption.rss`, в котором также определен заголовок программы, отпадает. Но приоритетность у файла `Test_caption.rss` выше, чем у `AIF`-ресурса, поэтому в названиях программы можно использовать содержимое файла `Test_caption.rss`, оба файла взаимозаменяемы. Чтобы показать разностороннее применение одних и тех же элементов программы, файл ресурса `Test_caption.rss` не используется в проекте `AifTest`, и его функции возложены на `AIF`-ресурсы.

В списке `CAPTION` используются два значения для `code` и `caption`. Переменная `caption` содержит название программы, заключенное в кавычки, а вот `code` определяет код языка, употребляемого для названия программы. В программе `AifTest` был применен код для английского языка `ELangEnglish`, подробней об этой теме мы поговорим в *главе 8*.

Затем перейдите в файл `Test.mmp` проекта `AifTest` и задекларируйте новые `AIF`-ресурсы приложения. Это всего две строки кода, добавленные в самый конец файла `*.mmp`. Не забывайте изменять идентификатор `UID3` для каждого нового приложения.

```

//*****
// файл Test.mmp

```

```

//J*****
TARGET          Test.app
TARGETTYPE      app

UID             0x00039CE 0x11000000

TARGETPATH      \system\apps\test

LANG            01

SOURCEPATH      ..\src
SOURCE          Test Main.cpp
SOURCE          Test Application.cpp
SOURCE          Test_AppView.cpp
SOURCE          Test_AppUi.cpp
SOURCE          Test_Document.cpp

SOURCEPATH      ..\group

RESOURCE        Test.rss

USERINCLUDE     ..\inc

SYSTEMINCLUDE   \epoc32\include

LIBRARY         euser.lib
LIBRARY         apparc.lib
LIBRARY         cone.lib
LIBRARY         eikcore.lib
LIBRARY         avkon.lib
LIBRARY         commonengine.lib

AIF             Test.aif..\Aif AifTest.rss \
cl2 Test.bmp MaskTest.bmp Testl.bmp MaskTestl.bmp
//*****

```

После ключевого слова AIF следует название для откомпилированного файла AIF-ресурса Test.aif. Новый проект называется AifTest, но мы используем содержимое программы Test (классы, ресурсы), поэтому и применяется название Test.aif. Затем определяется название папки проекта - Aif, где находится файл ресурса AifTest.rss. Во второй строке кода идет перечисление через пробел всех иконок и масок. Причем основная иконка для рабочего стола, должна прописываться первой. Атрибут c12 указывает на битность, то есть 12 бит для экрана с 4096 цветами.

И последнее дополнение нужно сделать в файле Test.pkg проекта AifTest. Изменяем уникальный идентификатор UID3 и указываем путь к откомпилированному файлу AIF.

```
/Test.pkg
; Файл необходим для создания установочного пакета

; Название программы, UID3, версия и тип упаковки программы #{"Test"b
(0x11000000), 1,0,0, TYPE=SISAPP

;Серия 60 версия 2.0
(0x101F7960), 0, 0, 0, {"Series60ProductID"}

"..\\..\\..\\epoc32\\data\\z\\system\\apps\\Test\\Test.aif"-
"!:\\system\\apps\\Test\\Test.aif"

"..\\..\\..\\epoc32\\release\\armi\\urel\\Test.app" -"
!:\\system\\apps\\Test\\Test.app"

*■. Л. Л. •\\epoc32\\data\\z\\system\\apps\\Test\\Test.r01"-
!:\\system\\apps\\Test\\Test.rsc"

".Л.\\..\\..\\epoc32\\data\\z\\system\\apps\\Test\\Test_caption.r01"-
*!:\\system\\apps\\Test\\Test_caption.rsc"
```

Вот и все изменения, которые коснулись проекта AifTest. Откомпилируйте программу и запустите эмулятор. Вы увидите, что на рабочем столе у приложения AifTest появилась иконка, как показано на рис. 7.7. А при запуске программы на панели состояния отобразится меньшая по размеру иконка. Теперь мы рассмотрим альтернативное или автоматизированное добавление AIF-ресурсов в программу.

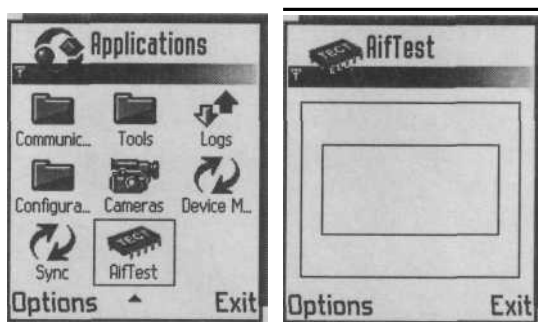


Рис. 7.6. Работа программы AifTest

## 7.5.1. Добавление AIF-ресурсов в C++ BuilderX

Откройте среду программирования C++ BuilderX и сформируйте или импортируйте новый проект. Затем создайте в директории проекта новую папку с названием aif и поместите в нее иконки для приложения. Выберите в меню команду **File => New** и в появившемся диалоговом окне **Object Gallery**, изображенном на рис. 7.7, выделите опцию **New Symbian AIF wizard** и нажмите кнопку **OK**, запустив тем самым, мастер добавления AIF-ресурса в проект.

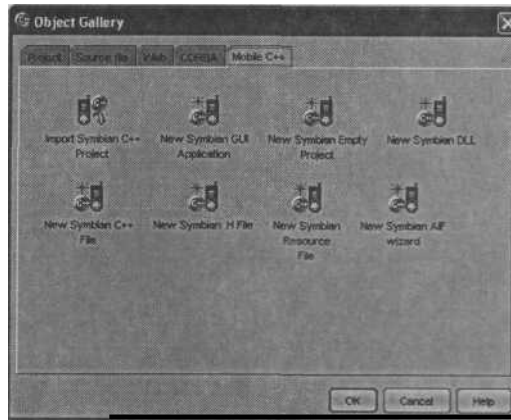


Рис. 7.7. Диалоговое окно Object Gallery

Появится новое диалоговое окно **New Symbian AIF wizard - Step 1 of 3**, показанное на рис. 7.8. В поле **AIF source directory** укажите путь к папке aif в директории проекта, в котором происходит добавления AIF-ресурса. Если это проект Test, то путь должен быть таким: C:\Symbian\Code\Test. Затем нажмите кнопку **Next**.

В следующем окне **New Symbian AIF wizard - Step 2 of 3**, изображенном на рис. 7.9, в поле **Bitmap file name** необходимо последовательно перечислить названия всех иконок, включая маски. После этого нажмите кнопку **Add** для каждого указанного названия. В области **Bitmaps** в столбик будут отображаться добавленные файлы. Для перехода к последнему этапу работы с мастером нажмите кнопку **Next**.

В последнем диалоговом окне **New Symbian AIF wizard - Step 3 of 3** мастера добавления AIF-ресурса, изображенном на рис. 7.10, нужно задать имя заголовку программы и выбрать код для используемого языка. Для этого в списке **Language** выберите код языка, а в поле **Caption** на выбранном языке дайте названия программе, нажав кнопку **Add**. В области **Defined captions** все добавленные коды и языки будут представлены в виде таблицы. Для окончания работы с мастером нажмите кнопку **Finish**.

После этого в активном проекте среды программирования C++ BuilderX на панели **Project** в папке \aif отобразятся проектные файлы изображения и AIF-

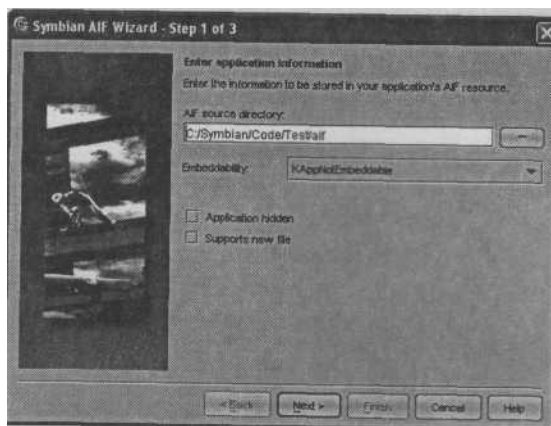


Рис. 7.8. Диалоговое окно New Symbian AIF wizard - Step 1 of 3

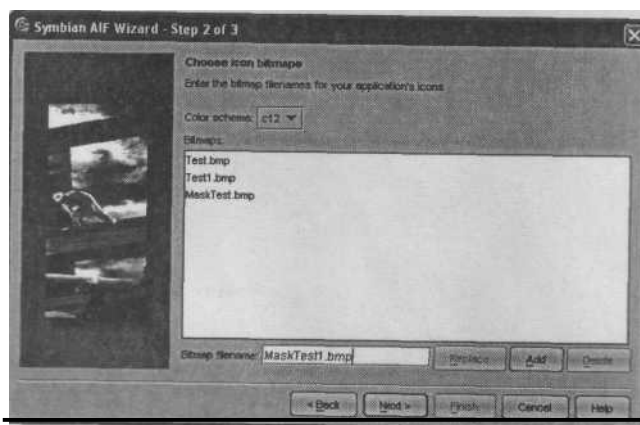


Рис. 7.9. Диалоговое окно New Symbian AIF wizard - Step 2 of 3

ресурсов. Посмотрите на рис. 7.11, где показан проект Test в C++ BuilderX с открытым файлом Test.rss. Среда C++ BuilderX автоматически сформировала AIF-ресурс с необходимыми атрибутами, а в проектном файле \*.mmp все созданные AIF-ресурсы будут так же прописаны соответствующим образом.

## 7.6. Сборка проекта компилятором

Что же происходит при компиляции и сборке проекта на системном уровне? Посмотрите на рис. 7.12, где схематично показаны процессы, происходящие во время компиляции и сборки проекта.

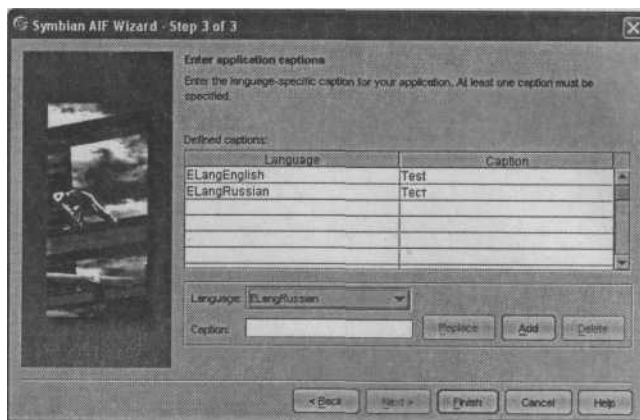


Рис. 7.10. Диалоговое окно New Symbian AIF wizard - Step 3 of 3

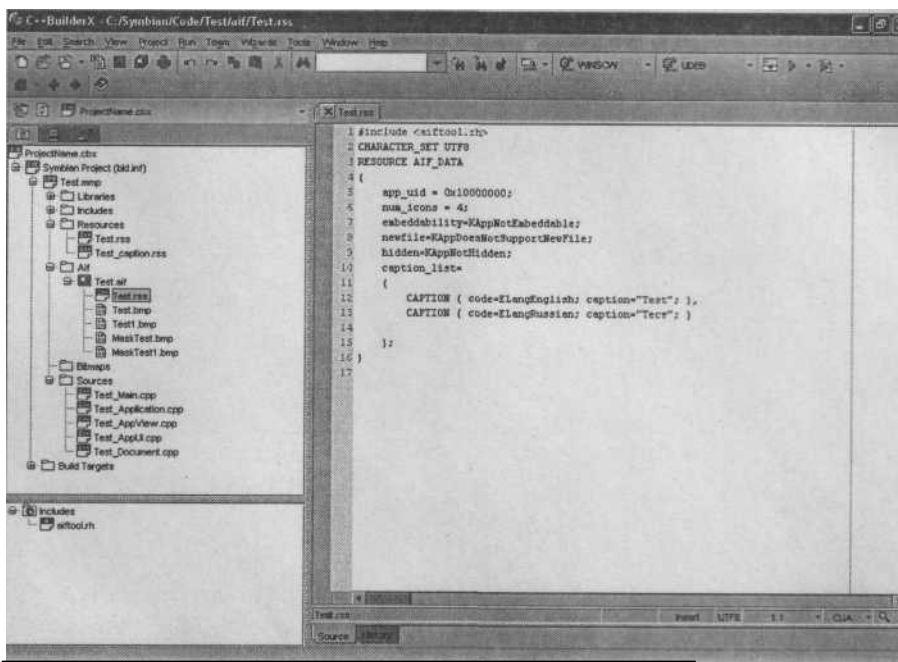


Рис. 7.11. Среда программирования C++ BuilderX с открытым проектом Test

На рис. 7.12 схематично изображено все то, что вы изучили в этой главе, за исключением процесса локализации (поддержки различных языков) приложения, который мы рассмотрим в следующей главе. После компиляции проекта получаются готовые компоненты приложения, в которые входят файлы:

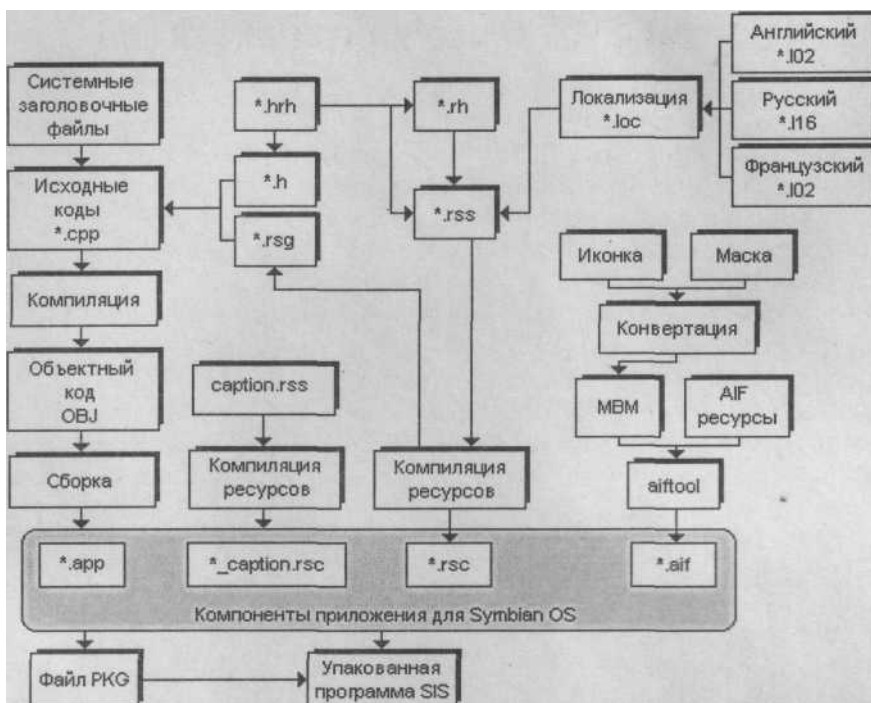


Рис. 7.12. Схема компиляции, сборки и упаковки проекта

- \*.app - готовый откомпилированный исходный код программы;
- Q \*\_caption.rsc - файлы ресурсов, содержащие заголовки;
- \*.rsc - файлы ресурсов программы;
- \*.aif - информационные файлы ресурсов приложения.

Все эти компоненты и составляют готовую программу, пригодную для работы на телефоне, при условии что компиляция и сборка проекта происходила для процессоров с архитектурой ARM (платформа ARMI). Далее все перечисленные компоненты программы упаковываются в специальный архив Symbian System Installation (SIS), оптимизированный под инсталляционную систему Symbian OS. То есть все компоненты программы с расширением \*.app, \*\_caption.rsc, \*.rsc и \*.aif запаковываются специальным образом и уже на телефоне распаковываются на выбранный пользователем диск в директорию \system\app\НаЗВаНHe\_програММbi. Вы можете перенести файлы \*.app, \*\_caption.rsc, \*.rsc и \*.aif в системную папку телефона \system\app\НаЗВаНHe\_програММbi на любой из дисков. Делать это не рекомендуется, однако большое количество программ из Интернета распространяется именно таким образом, нанести вред телефону таким способом инсталляции программ сложно, но лучше не экспериментировать. А теперь на основе всей полученной информации создадим установочный пакет для Symbian OS, учитывая все изложенные нюансы.

## 7.7. Создание установочного пакета SIS

Я уверен, что вы уже не раз пробовали создать установочный файл SIS и, скорее всего, у вас ничего не получилось, а если и получилось, то на телефоне это приложение работать отказывалось. Поэтому рассмотрим правильный способ создания дистрибутива для Symbian OS вне зависимости от использования среды программирования, командной строки или SISAR.

После того, как вы многократно протестировали программу на эмуляторе телефона, можно приступить к созданию установочного пакета.

***Папка с проектом должна обязательно находиться в каталоге SDK в папке Epos32 для любой платформы!***

В случае с проектом AifTest, его папка должна находиться в каталоге C:\Symbian\Series60\Epos32\AifTest! Только из папки Epos32 возможно создание дистрибутива SIS. Если вы используете C++ BuilderX, то после размещения папки с проектом в директории Epos32 выполните компиляцию проекта, как описано в *главе 3*. В среде программирования CodeWarrior необходимо выполнить в точности все действия из *главы 2*. Для программы SISAR порядок действий описан в *главе 4*.

Если вы тестировали и компилировали программу под платформы WINCSW, ARMI или THUMB и ошибок не было, а при упаковке программы возникает проблема, то эта ошибка точно связана с указанием путей к готовым компонентам программы в файле PKG. Как уже упоминалось, в разных SDK и платформах готовые или откомпилированные компоненты программы могут находиться в разных папках, в *разделе 7.3.13* указаны правильные пути к файлам.

Если вы не уверены в местонахождении файлов, всегда можно перейти в каталог SDK и найти их самостоятельно, указав правильные пути в файле PKG. Сам файл PKG вашего проекта при создании SIS-архива лучше расположить в папке \group рабочего проекта вместе с файлом MM P. Это гарантия безошибочной упаковки программы, и, конечно, не стоит забывать о правильных идентификаторах UID2, UID3 и Platform UID.



## Глава 8. Интерфейс пользователя

В Symbian OS реализовано огромное количество системных классов для создания пользовательского интерфейса приложения. Это шаблонные классы, использование которых приводит к созданию меню, диалогов, таблиц, редакторов списков и других элементов интерфейса, упрощающих работу пользователя.

Платформы UIQ и серия 60 в силу разных параметров дисплеев имеют два различных пользовательских интерфейса, поэтому имеет смысл коснуться общих характеристик для обеих платформ.

### 8.1. Платформа UIQ

Интерфейс пользователя UIQ используется в телефонах Sony Ericsson, Motorola, Arima и BenQ. Дисплей телефонов имеет достаточно большой размер

208 x 320 пикселей и предусматривает сенсорный ввод данных для облегчения работы пользователя. Все рабочее пространство дисплеев телефонов на платформе UIQ состоит из нескольких панелей. На рис. 8.1 изображен экран телефона UIQ с обозначением имеющихся панелей.

- Application Picker - панель, содержащая иконки приложений;
- Menu bar - панель меню;
- Main pane - клиентская область экрана;
- Toolbar — панель инструментальных средств;
- Status bar - панель состояния.



Рис. 8.1. Панели UIQ

#### 8.1.1. Панель Application Picker

На панели *Application Picker* в виде иконок отображаются доступные пользователю приложения. Когда приложение запущено, иконка этой программы выделена графически, как показано на рис. 8.1. Для работы с программами пользователь выбирает необходимую иконку приложения на панели *Application Picker*, запуская тем самым рабочий цикл программы.

#### 8.1.2. Панель Menu bar

В тот момент, когда приложение запущено, возникает необходимость выполнения различных команд. Для удобства работы с приложениями предусмотрена стандартная панель меню *Menu bar*. Работа с *Menu bar* напоминает работу с обыч-

новенной линейкой меню любой программы.

### 8.1.3. Клиентская область экрана

*Клиентская область экрана* служит для представления основных данных программы. Например, если это календарь, то вы увидите числа и дни недели, если это текстовый редактор, то можно будет редактировать, читать и набирать текст.

### 8.1.4. Панель Toolbar

Панель *Toolbar* используется в программах опционально и может быть не реализована в приложении. На панели *Toolbar* элементы представлены в виде кнопок и служат для определения различного рода командных действий и других специфических операций, например, для отображения цифр или букв алфавита.

### 8.1.5. Панель Status bar

На панели состояния почти всегда отображается кнопка виртуальной клавиатуры, нажав на которую пользователю становится доступной клавиатура для сенсорного ввода данных. Так же на панели *Status bar* может находиться индикатор заряда батареи, дата и время.

## 8.2. Серия 60

Интерфейс пользователя телефонов, использующих серию 60, очень отличается от интерфейса в UIQ. Телефоны серии 60 имеют разрешение 176 x 208 пикселей, и сенсорный ввод не поддерживается. Экран телефона серии 60 разделен на три панели (см. рис. 8.2, где изображен дисплей телефона этой серии с обозначением панелей).

- Status Pane - панель состояния;
- Main Pane - основная панель;
- Control Pane - панель контроля.



Рис. 8.2. Панели серии 60

### 8.2.1. Панель Status Pane

Панель состояния в серии 60 находится в верхней части дисплея. В играх обычно *Status Pane* скрывают, реализуя тем самым полноэкранный режим работы программы. Панель состояния состоит из шести панелей:

- Signal Pane* - сигнальная панель, содержащая индикатор сети;
- Context Pane* - контекстная панель, предназначенная для отображения иконки приложения (42 x 29 пикселей). В предыдущей главе с помощью AIF-ресурса мы нарисовали иконку программы Test как раз в контекстной

- панели;
- *Title Pane* - титульная панель для представления заголовка приложения;
- *Navigation Pane* - навигационная панель, служит для удобства навигации в программах и выполняется в виде вкладок;
- *Indicator Pane* - панель индикатора предназначена для отображения индикатора заряда батареи.

## 8.2.2. Панель *Main Pane*

Основная часть экрана служит для отображения данных приложения, рабочего стола телефона, графики и так далее. В серии 60 размер *Main Pane* - это вся клиентская область экрана - 174 x 132 пикселя.

## 8.2.3. Панель *Control Pane*

Панель контроля (*Control Pane*) располагается в нижней части экрана над подэкранными клавишами телефона с командами. Как правило, это стандартный набор команд: **Exit, Back, Cancel, Options**. С помощью команд, расположенных на панели контроля происходит работа с меню, диалогами, списками и редакторами.

## 8.3. Ресурсы

Большая часть составляющих интерфейса пользователя в серии 60 строится на основе использования ресурсов. Как правило, это файлы исходного кода с расширениями \*.rss, \*.rh, \*.hrh, \*.loc и caption.rss, где описываются различные элементы программы. Ресурсы в Symbian OS могут использовать следующие типы данных:

- Q BYTE - 1-байтовое число без знака;
- WORD - 2-байтовое число;
- LONG - 4-байтовое число;
- DOUBLE - 8-байтовое число с плавающей точкой;
- TEXT - строка текста;
- Q LTEXT - строка текста с уникодом (Unicode);
- BUF - строка текста с уникодом (Unicode);
- BUF8 - 8-битная символьная строка текста;
- BUF <p> - строка текста с использованием уникада, длина которой задается в параметре <p>;
- Q LINK - 16-битный идентификатор (ID) для специальных ресурсов;
- LLINK - 32-битный идентификатор (ID) для специальных ресурсов;
- SRLINK - ссылка на идентификатор (ID) специализированного ресурса;
- SRUCT - определяет имя структуры;
- ENUM - перечисляемый тип;
- RESOURCE - определяет ресурсы приложения;
- NAME - 20-битный идентификатор (ID) ресурсов приложения.

Как видите, типы данных, применяемых в Symbian OS, используют вполне стандартные определения, встречающиеся и в программировании для других систем.

Для того чтобы создать свою структуру, строится обыкновенная конструкция исходного кода:

```
STRUCT MYNAME

BYT bvalue = 0;
LON lvalue = 0;
BUF buf;
```

Точно такая же ситуация обстоит и с перечисляемыми типами. Объявляется перечисляемый тип с помощью ключевого слова ENUM и далее следует название самого типа. При определении ресурсов приложения задействуется другая система. RESOURCE <определение\_ресурса> <ваше\_имя>

Посмотрите, как может выглядеть, например, определение ресурса для меню:

```
RESOURCE MENU_BAR r_mymenu_bar
{
// реализация ресурса
```

Ключевое слово MENU\_BAR объявляет меню для программы, а в качестве названия данного ресурса служит определение r\_mymenu\_bar. В названиях ресурсов обычно используются знаки подчеркивания, и начальной буквой является прописная буква г английского алфавита.

Системная библиотека Symbian OS содержит заголовочные файлы с определением ресурсов, перечисляемых типов, структур и констант. При работе с ресурсами надо помнить о подключении системных файлов ресурсов Symbian OS (Uikon). #include <eikon.rh> tinclude <eikon.hrh>

Системные файлы ресурсов Symbian OS подключаются независимо от использования UIQ или серии 60. При создании приложения под конкретную платформу, нужно подключить дополнительные файлы ресурсов, например, для серии 60 это: #include <avkon.rh> ◆include <avkon.hrh>

## 8.4. Меню

Во всех программах (вне зависимости от операционной системы) используется механизм *меню*. Как правило, это список команд, при выборе которых пользователь может совершать определенные действия. В Symbian OS почти все

программы имеют систему меню. В серии 60 меню назначается для подэкранных клавиш, обычно это левая клавиша **Options**, но при желании меню можно назначить и на правую клавишу. Давайте на примере рассмотрим систему меню серии 60, но сначала установим требования к будущей программе и предложим алгоритм ее решения. В качестве каркаса послужит приложение AifTest, созданное в *главе 7*.

При запуске программы AifTest, подэкранными клавишам назначаются команды **Options** и **Exit**. Команду Exit вы уже умеете реализовывать, а для команды **Options** мы назначим меню в виде списка из трех команд. Одна из команд будет иметь вложенное или вторичное меню со списком еще из двух команд. При выборе пользователем одной из команд должны происходить назначенные действия. В качестве подобного действия была задействована работа с одним из видов диалогов под названием Note (заметка, примечание). В каждом таком информационном примечании будет содержаться текстовое сообщение с названием выбранной команды. Для решения этой задачи нам потребуется создать дополнительные команды в файле Test.hrh, а в файле ресурса Test.rss сформировать меню с помощью заданных команд и уже в файле Test\_AppUi.cpp в функции HandleCommandL () непосредственно описать действия для каждой команды.

Демонстрационный пример MenuTest находится на компакт-диске в папке \Code\MenuTest. Перейдем к реализации поставленной задачи. Первым делом надо создать команды. Для этого в файле Test.hrh в перечисляемом типе TTestCommands произведем необходимые объявления. Файл Test.hrh вы найдете на компакт-диске в папке \Code\MenuTest\inc\Test.hrh, рассмотрим исходный код.

```
//. заголовочный файл Test.hrh
// содержит спецификацию ресурсов
у/*****
// проверяем лексему
#ifndef Test HRH
#define Test HRH

// определяем команды для меню
enum TTestCommands
{
    ETestCommand = 1,
    ETestConfirmation,
    ETestInformation,
    ETestNotes,
    ETestError,
    ETestWarning
```

```
#endif// _____ Test_HRH_
/y*****
```

Команда ETestCommand не используется в программе, она оставлена с прошлого приложения, но любая первая команда, идущая в перечислении, обязана объявляться со значением равным 1 или 0x600. Это связано с работой системы. В названиях созданных команд используются виды диалоговых информационных уведомлений:

- Configuration Note сообщает о завершении какого-либо действия, например, о завершении скачивания файла из Интернета или об окончании сохранения игры. Это информационное окно содержит текстовое сообщение и графический элемент в виде галочки;
- Information Note - сообщение информационного характера, обычно связанного с ошибками сделанными пользователем, например, ввод неправильного пароля или имени. Информационное окно содержит текст и графический элемент в виде прописной английской буквы i;
- Error Note — этот вид примечания связан с серьезными ошибками, вызванными сбоем программы. Окно содержит текст три восклицательных знака;
- Warning Note - информационное сообщение, связанное с предупреждением, в окне рисуется большой восклицательный знак и текст сообщения.

Команда ETestNote в нашей программе будет иметь вторичное меню со списком из двух команд. Теперь перейдем к файлу Test.rss. На компакт-диске он находится в папке \Code\MenuTest\group\Test.rss.

```
/y*****
// файл Test.rss
// ресурсы приложения
// it*****
II (ID)
NAME MENS

// подключаем системные библиотеки
◆include <eikon.rh>
tinclude <avkon.rh>
◆include <avkon.rsg>
// подключаем заголовочный файл
#include "Test.hrh"

// сигнатура
/y*****

RESOURCE RSSSIGNATURE
```



```

MENU_ITEM
{
    command = ETestNotes;
    txt = "Notes";
    cascade = r_test_submenu; Ъ

```

```

MENU_ITEM
{
    command = ETestInformation;
    txt «■ "Information Note";

```

**Ъ**

```

MENU_ITEM
{
    command = EAknCmdExit;
    txt = "Exit";

```

```

// вторичное меню
/y*****

```

```

RESOURCE MENU_PANE    r_test_submenu
{
    items =
    {
        MENU_ITEM
        {
            command = ETestError;
            txt = "Error Note";
        },
        MENU_ITEM
        {
            command = ETestWarning;
            txt = "Warning Note";
        }
    }

```

```

/y *****

```

В самом начале исходного кода происходит изменение идентификатора ресурса на новое название MENS. Затем в ресурсе EIK APP INFO объявляется о создании меню для подэкранной клавиши.



```
menubar - r_test_menubar;
```

Слово `menubar` - это зарезервированное слово, объявленное в системном заголовочном файле `avkon.rh` и указывающее на создание меню в программе. Дается название создаваемому меню `r_test_menubar` - это произвольное название, которое вы можете выбрать сами. Семантика в названии может быть любой, но рекомендуется применять английскую букву `r` в начале названия и знаки подчеркивания в самом названии. В нашей программе мы объявили меню с названием `r_test_menubar`, теперь надо описать этот вид меню. Делается это следующим образом:

```
*****  
// создаем меню  
-----  
RESOURCE MENU_BAR r_test_menubar {  
    titles = {  
        MENU_TITLE  
        {  
            menu_pane = r_test_menu;  
        }  
    }  
}  
  
*****
```

Ресурс `MENU_BAR` описывает вид создаваемого меню. С помощью конструкции из слов `titles` и `MENU_TITLE`, а также ключевого слова `menu_pane` определяется меню в виде панели со списком команд под названием `r_test_menu`. И уже в ресурсе `MENU_PANE` описываются команды для списка меню `r_test_menu`.

Как видите, применяется чередующаяся друг за другом конструкция исходного кода, где сначала объявляется меню, потом описывается вид создаваемого меню и в конце задается список команд в ресурсе `MENU_PANE`.

```
// определяем команды для меню "Options"  
//*****  
RESOURCE MENUPANE    rtestmenu  
  
    items  
  
        // Test.hrh  
        MENUITEM
```

```

        command = ETestConfirmation;
        txt = "Confirmation Note";
Б
MENU_ITEM {
    command = ETestNotes;
    txt = "Notes";
    cascade = r_test_submenu;

MENU_ITEM {
    command = ETestInformation;
    txt = "Information Note";
},
MENU_ITEM {
    command = EAknCmdExit;
    txt = "Exit";

```

При описании команд меню применяется ключевое слово MENU\_ITEM с фигурными скобками, внутри которых содержится исходный код для каждой команды. Ресурс command служит для идентификации определенной команды, которые объявлены в файле Test.hrh. С помощью ресурса txt задается текст, устанавливающий вид команды. Когда вы нажмете подэкранную клавишу **Options**, то появится меню со списком команд, названия которых задаются в MENU\_PANE. Названия заданы на английском языке, только он отображается корректно. Для того чтобы использовать русский или другие языки в меню или во всей программе, нужно использовать систему локализации, о которой мы проговорим в конце этой главы в разделе 8.5. (Локализация). В списке присутствует команда выхода и три дополнительные команды, а одна из них, ETestNotes, содержит еще вторичное меню.

Для определения вторичного меню создан следующий код:  
 cascade = r\_test\_submenu;

То есть, содержится дополнительный список вторичного меню, команды которого так же нужно описать:

```

// вторичное меню
//*****

```

```

RESOURCE MENU_PANE      r_test_submenu {
    items = {
        MENU_ITEM
        {
            command = ETestError; txt =
            "Error Note"; },
        MENU_ITEM
        {
            command = ETestWarning; txt =
            "Warning Note"; }
    }; }

```

Как видите, описание вторичного меню ничем не отличается от описания основного. Тот же набор идентификаторов, схема описания и лишь ресурс `r_test_submenu` указывает на принадлежность к вложенному меню.

На этом формирование меню и его команд заканчивается. Если сейчас откомпилировать и запустить программу `MenuTest`, то при нажатии на клавишу **Options** станут доступными созданные основное и вторичное меню (посмотрите на рис. 8.3).

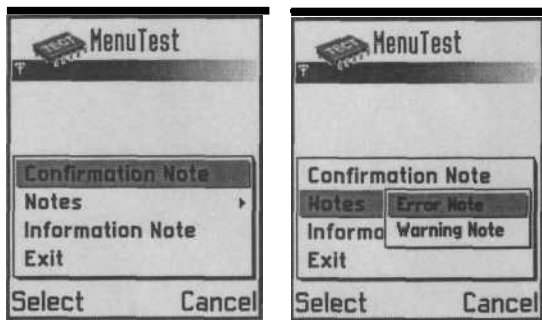


Рис. 8.3. Меню в программе `MenuTest`

Но при выборе одной из команд запустится механизм обработки внештатных ситуаций (паника) и приложение будет закрыто, поэтому надо переместиться в файл `Test_AppUi.cpp` и написать исходный код для сформированного меню.

В программе `MenuTest` файл `Test_AppUi.cpp` находится на компакт-диске в папке `\Code\MenuTest\src\Test_appUi.cpp`, рассмотрим исходный код этого файла.

```

// файл Test_AppUi.cpp

J
// реализация класса CTestAppUi
/y*****

i
II подключаем системные библиотеки
◆include <aknnotewrappers.h>
◆include <avkon.hrh>

// подключаем файл ресурса
tinclude <Test.rsg>

// подключаем заголовочные файлы
#include "Test AppUi.h"
#include "Test AppView.h"
#include "Test.hrh"
◆include "Test.pan"
// конструктор
void CTestAppUi::ConstructL()
{
    // основной конструктор, завершающий создание UI
    BaseConstructL();
    // создаем клиентскую область
    iAppView = CTestAppView::NewL(ClientRect());
    // добавляем в стек
    AddToStackL(iAppView);
}
// конструктор
CTestAppUi::CTestAppUi() {
1
// деструктор
CTestAppUi::~CTestAppUi()
{
    if (iAppView)
    {
        // удаляем из стека
        iEikonEnv->RemoveFromStack(iAppView) ; //
        удаляем объект delete iAppView; //
        обнуляем iAppView = NULL;
    }
}

```

```

// обработка команд
void CTestAppUi::HandleCommandL(Tint aCommand)
{
    switch(aCommand) {
        // меню Confirmation:
        case ETestConfirmation:
        {
            LIT (text1, "Confirmation Note");
            CAknConfirmationNote* NoteConfirmation;
            NoteConfirmation = new (ELeave)
                CAknConfirmationNote;
            NoteConfirmation ->ExecuteLD(text1);
            break; }
        // меню Information case
        ETestInformation: {
            LIT (text2, "Information Note");
            CAknInformationNote* NoteInformation;
            NoteInformation = new (ELeave)
                CAknInformationNote;
            NoteInformation ->ExecuteLD(text2);
            break; }
        // вторичное меню Error
        case ETestError: {
            LIT (text3, "Error Note");
            CAknErrorNote* NoteError;
            NoteError = new (ELeave) CAknErrorNote;
            NoteError ->ExecuteLD(text3);
            break; }
        // вторичное меню Warning
        case ETestWarning:
        {
            LIT (text4, "Warning Note");
            CAknWarningNote* NoteWarning;
            NoteWarning = new (ELeave) CAknWarningNote;
            NoteWarning ->ExecuteLD(text4);
            break;
        }
    }
}

```

```

// ВЫХОД
case EAknSoftkeyExit:

    case EEikCmdExit:
    {
        Exit ();
    break; }
    // паника
    default:
        User::Panic ( L("MenuTest"), ETestBasicUi);
break; }

```

\*\*\*\*\*

/у

Все действия по обработке команд происходят в функции HandleCommandL (). В качестве *ключей* (case) используются команды, определенные в файле Test.hrh, это: ETestConfirmation, ETestInformation, ETestError и ETestWarning. Обратите внимание, между ключами содержится исходный код команды, заключенный в фигурные скобки, что редко применяется в программировании на C++. Дело в том, что Metrowerks CodeWarrior при компиляции программы под платформу ARM UREL или THUMB UREL перестраховывается и видит в этом ошибку. Только использование фигурных скобок дает возможность запаковать программу в SIS-архив. При эмуляции программы на компьютере и в C++ BuilderX такой проблемы не возникает. Для команды ETestConfirmation сначала задается текстовое сообщение, которое будет написано внутри информационного сообщения. Это текст Confirmation Note, указывающий на вид используемого примечания. Затем в двух строках кода: CAknWarningNote\* NoteWarning; NoteWarning = new (ELeave) CAknWarningNote;

происходит создание и инициализация объекта, представляющего информационное примечание типа Confirmation Note. И в строке NoteWarning - >ExecuteLD(text); созданное информационное примечание выводится на экран телефона.

Обработка всех последующих команд происходит аналогичным образом, но с использованием классов CAknInformationNote, CAknErrorNote и CAknWarningNote для каждого типа информационного примечания. На рис. 8.4 изображена работа программы MenuTest со всеми вариантами сообщений.

Таким вот образом и происходит работа с меню в Symbian OS для серии 60, что позволяет программисту создавать хорошую систему обработки команд и навигацию в приложении.

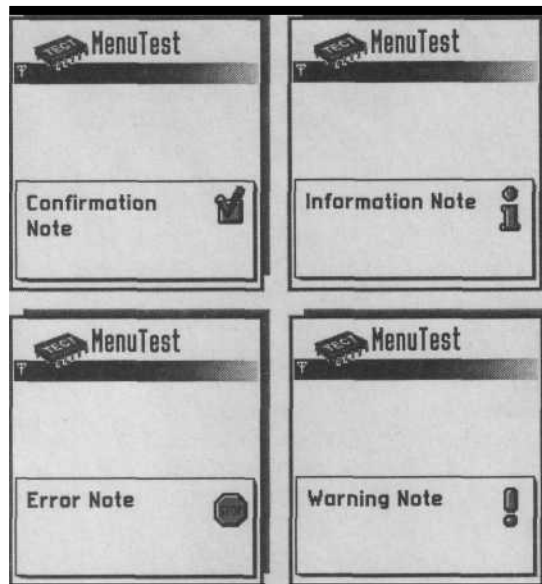


Рис. 8.4. Информационные примечания в программе MenuTest

## 8.5. Локализация

Создавая программу для массового использования, надо обязательно предусмотреть возможность поддержки максимального количества языков. Для этого в Symbian OS реализована система *локализации* приложения или поддержки языков.

Каждый язык в Symbian OS имеет свой код - это ключевое слово и целочисленное значение. Например, для русского языка это ELangRussian = 16. В табл. 8.1 даны коды для всех языков, поддерживаемых Symbian OS.

Таблица 8.1. Коды языков

Языки	Коды
Английский Великобритания	ELangEnglish = 1
Французский	ELangFrench = 2
Немецкий	ELangGerman = 3
Испанский	ELangSpanish = 4
Итальянский	ELangItalian = 5
Шведский	ELangSwedish = 6
Датский	ELangDanish = 7
Норвежский	ELangNorwegian = 8
Финский	ELangFinnish = 9
Американский	ELangAmerican = 10
Швейцарский французский	ELangSwissFrench = 11

**Таблица 8.1. Коды языков (продолжение)**

<b>Языки</b>	<b>Коды</b>
Швейцарский немецкий	ELangSwissGerman = 12
Португальский	ELangPortuguese = 13
Турецкий	ELangTurkish = 14
Исландский	ELangIcelandic = 15
Русский	ELangRussian = 16
Венгерский	ELangHungarian = 17
Голландский	ELangDutch = 18
Бельгийский флемеиш	ELangBelgianFlemish = 19
Бельгийский французский	ELangBelgianFrench = 21
Австралийский английский	ELangAustralian = 20
Австрийский немецкий	ELangAustrian = 22
Новозеландский английский	ELangNewZealand = 23
Международный французский	ELangInternationalFrench = 24
Чешский	ELangCzech = 25 ELangSlovak
Словацкий	= 26 ELangPolish = 27
Польский	ELangSlovenian = 28
Словенский	ELangTaiwanChinese = 29
Тайваньский китайский	ELangHongKongChinese = 30
Гонконгский китайский	ELangPrcChinese = 31
Китайский	ELangJapanese = 32
Японский	ELangThai = 33 ELangAfrikaans
Тайский	= 34 ELangAlbanian = 35
Африкаанс	ELangAmharic = 36
Албанский	ELangArabic = 37
Амхарик	ELangArmenian = 38
Арабский	ELangTagalog = 39
Армянский	ELangBelarussian = 40
Тагалог	ELangBengali = 41
Белорусский	ELangBulgarian = 42
Бенгальский	ELangBurmese = 43
Болгарский	ELangCatalan = 44
Бирманский	ELangCroatian = 45
Каталонский	ELangCanadianEnglish = 46
Хорватский	ELangInternationalEnglish = 47
Канадский английский	ELangSouthAfricanEnglish = 48
Международный английский	ELangEstonian = 49 ELangFarsi
Южноафриканский английский	= 50 ELangCanadianFrench =
Эстонский	51 ELangScotsGaelic = 52
Фарси	ELangGeorgian = 53
Канадский французский	ELangGreek = 54
Галльский язык	ELangCyprusGreek = 55
Грузинский	ELangGujarati = 56
Греческий	ELangHebrew = 57
Кипрский греческий	
Гуярати	
Еврейский	



Таблица 8.1. Коды языков (продолжение)

Языки	Коды
Хинди	ELangHindi = 58
Индонезийский	ELangIndonesian = 59 ELangIrish = 60 ELangSwissItalian = 61
Ирландский	ELangKannada = 62 ELang
Швейцарский итальянский	Kazakh = 63 ELangKhmer = 64
Канадский	ELang Korean = 65 ELangLao = 66 ELang Latvian = 67
Казахский	ELangLithuanian = 68
Кхмерский	ELangMacedonian = 69
Корейский	ELangMalay = 70
Лаосский	ELangMalayalam = 71,
Латвийский	ELangMarathi = 72
Литовский	ELangMoldavian = 73
Македонский	ELangMongolian = 74
Малайский	ELangNorwegianNynorsk = 75
Мал ялам	ELangBrazilianPortuguese = 76
Маратхи	ELangPunjabi = 77
Молдавский	ELangRomanian = 78
Монгольский	ELangSerbian = 79
Норвежский	ELangSinhalese = 80
Бразильский португальский	ELangSomali = 81
Пенджабский	ELangInternationalSpanish = 82
Румынский	ELangLatinAmericanSpanish = 83
Сербский	ELangSwahili = 84
Сенегал	ELangFinlandSwedish = 85
Сомали	ELangReserved1 = 86
Международный испанский	ELangTamil = 87
Латиноамериканский испанский	ELangTelugu = 88
Суахили	ELangTibetan = 89
Шведский финский	ELangTigrinya = 90
Зарезервирован для будущего использования	ELangCyprusTurkish = 91
Тамильский	ELangTurkmen = 92
Телугу	ELangUkrainian = 93
Тибетский	ELangUrdu = 94
Тигриния	ELangReserved2 = 95
Кипрский турецкий	ELangVietnamese = 96
Туркменский	ELangWelsh = 97
Украинский	ELangZulu = 98
Урду	ELangOther = 99
Зарезервирован для будущего использования	
Вьетнамский	
Уэльский	
Зулу	
Для других языков	

Зная коды, можно очень легко локализовать программу. Для этого в всем используемым в программе командам, текстовым сообщениям и строкам надо за-

дать идентификаторы или ключевые слова. То есть назначить замену названий на определенную аббревиатуру. Например, вместо слова Exit можно использовать LANG\_MENU\_EXIT. После того, как вы замените весь текст в программе на макросы, в приложении создаются файлы с расширениями \*.Н6 (русский язык), \*.140 (белорусский язык), \*.193 (украинский язык) и так далее. Ровно столько, сколько языков вы пожелаете поддерживать в своей программе. В каждом файле вы перечисляете все макросы, применяемые для замены текста, и около каждого макроса прописываете название команды или текст на поддерживаемых языках. Если это файл \*.101, то перечисленные макросы должны быть описаны на английском языке, а если это \*.И6, то на русском и так далее. Например, это может выглядеть так:

```

//*****
// файл Language.116
// локализация приложения
//*****
// заголовок для языка

#define ELanguage      ELangRussian

// команды меню

◆define      LANG_MENU_CONFIRMATION      "Подтверждение"
#define      LANG_MENU_INFORMATION      "Информация"
#define      LANG_MENU_NOTES      "Примечание"
#define      LANG_MENU_ERROR      "Ошибка"
◆define      LANG_MENU_WARNING      "Предупреждение"
#define      LANG_MENU_EXIT      "Выход"

//*****

```

В файле Language.116 локализованы команды программы MenuTest. Точно так же осуществляется локализация для любых языков, поддерживаемых Sym-bian OS. После этого создается файл с расширением \*.loc, например, MyLanguage.loc, где подключаются все имеющиеся файлы локализации программы, например:

```

//*****
// файл MyLanguage.loc
in
II английский язык
#ifdef      LANGUAGE 01
◆include "Language.101"

```

```

#endif

// французский язык
#ifdef LANGUAGE 02
#include "Language.102"
#endif

// немецкий язык #ifdef
LANGUAGE 03 #include
"Language.103" #endif

// русский язык #ifdef
LANGUAGE 16 ◆include
"Language.116" #endif

// белорусский язык
#ifdef LANGUAGE 40
◆include "Language.140"
#endif

// украинский язык
#ifdef LANGUAGE 93
#include "Language.193"
#endif

```

---

Все созданные файлы помещаются в каталог приложения, обычно в папку \inc, но принципиального значения это не имеет. Чтобы программа увидела все ваши старания по локализации, в файлах \*.rss, \*\_caption.rss и \*.aif, подключается файл спецификации, который описывает использование файлов локализации; в нашем примере это My Language, loc и системный файл avkon.loc. ◆include "MyLanguage.loc"

И последнее действие заключается в декларировании используемых языков для локализации программы в файле MMP. Например, если вы в локализации применяете английский, французский, немецкий, русский, белорусский и украинский языки, то в файле MMP после ключевого слова LANG должны быть перечислены числовые значения для каждого языка. LANG 01 02 03 16 40 93

Коды пишутся через пробелы, без каких-либо знаков препинания и так для всех языков локализации. В файле PKG вашего проекта тоже должно присутствовать обозначение языков в виде простой отдельной строки с кодами языков, например:

После того как вы создадите SIS-архив для работы на телефоне в момент его установки, инсталляционная система Symbian OS определит по кодам имеющиеся файлы локализации и предложит пользователю выбрать язык для интерфейса программы. Единственная проблема, возникающая с файлами локализации, это работа на эмуляторах телефонов. К сожалению, не всегда локализованные программы функционируют правильно. В SDK серии 60 в папке Example находится программа Language, иллюстрирующая процесс локализации программ. На основе полученной информации вам не составит труда разобраться с приложением самостоятельно.

## 8.6. Получение данных от пользователя

Создавая меню в программе MenuTest, для обработки команд полученных от пользователя была задействована функция HandleCommandL (). Эта функция позволяет обрабатывать команды для меню или диалогов, которые были назначены для двух подэкранных клавиш телефона. основополагающий принцип работы этой функции заключается в использовании оператора switch для перебора имеющихся вариантов в обработке событий, полученных от пользователя. Функция HandleCommandL () имеет ограниченный диапазон работы и предназначена для работы с меню, диалогами с помощью двух подэкранных клавиш телефона. В системной библиотеке Symbian OS существует еще ряд функций, призванных упростить процесс обработки событий, полученных с различных клавиш телефона. Вот основные из них:

- Q CCoeAppUi: :HandleKeyEventL () - обрабатывает события, полученные с клавиш телефона от пользователя;
- Q CCoeAppUi: :HandleForegroundEventL () - функция предназначена для обработки команды находящейся в фокусе курсора;
- CCoeAppUi: :HandleSystemEventL () - обрабатывает системные события, сгенерированные сервером окна (Window Server);
- Q CCoeAppUi: :HandleApplicationSpecificEventL() - необходима для обработки специфических событий приложения;
- CCoeControl: :OfferKeyEventL () - обработка событий с клавиш телефона при помощи ключевых кодов.

В заголовочных файлах e32keys.h и Uikon.hrh находится большинство ключевых кодов для работы с вышеперечисленными функциями. Например, при использовании функции CCoeControl: : OfferKeyEventL () можно создать такую конструкцию кода обрабатывающую нажатие клавиш телефона.

```
TKeyResponse CMyTestControl::OfferKeyEventL( const
TKeyEventS aKeyEvent, TEventCode aType)
{
    // узнаем, нажата любая клавиша или нет
    if (aType == EEventKey)
```

```

// одна из клавиш нажата, значит, получаем код if
(aKeyEvent.iCode)
{
    case EKeyDownArrow: // вниз
    // действия на нажатую клавишу
    return EKeyWasConsumed;

    case EKeyUpArrow: // вверх //
    // действия на нажатую клавишу
    return EKeyWasConsumed;

    case EKeyLeftArrow: // влево
    // действия на нажатую клавишу
    return EKeyWasConsumed;

    case EKeyRightArrow: // вправо
    // действия на нажатую клавишу
    return EKeyWasConsumed;

    default:
    return EKeyWasConsumed;
}

return EKeyWasConsumed;

```

В справочной системе SDK серии 60 в разделе *C++ API Reference Ю Window Server* существует перечисляемый тип `TKeyCode`, где представлены все ключевые коды для клавиш телефона.

## 8.7. Списки

Одним из основных элементов пользовательского интерфейса являются списки. *Список* - это перечисление определенных команд, опций, текста, перемещаясь по которым пользователь в праве выбирать один или несколько компонентов в зависимости от вида списка.

В Symbian OS существуют три вида списков: `Grid` (Сетка), `Vertical List` (Вертикальный список) и `Setting List` (Список настроек). В каждом из них имеются дополнительные разновидности списков.

Все списки формируются примерно одинаково. Чтобы создать список в программе, необходимо сначала обратиться к файлам ресурсов приложения и задекларировать объявления списка, примерно так, как мы делали это для меню, но со своей спецификой. Затем в исходном коде программы описать код для создания класса списка и обработки программ.

### 8.7.1. Вертикальный список

Вертикальный вид списка (Vertical List) располагается на экране телефона вертикально строка за строкой. Посмотрите на рис. 8.5, где изображен вертикальный вид списка.

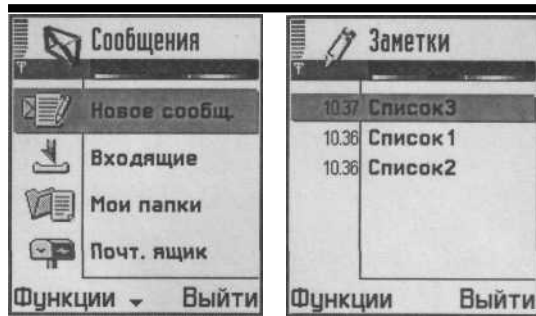


Рис. 8.5. Вертикальный список

При переходе на страницу со списком курсор всегда выделяет самую первую верхнюю строку вне зависимости от нумерации. На рис. 8.5. это хорошо видно. В качестве строки списка может выступать как команда, так и просто строка текста или же набор опций, которые и составляют компоненты списка. Выбрать один из компонентов списка можно, выделив его курсором и нажав клавишу ОК, либо открыв меню и выбрав соответствующую команду для этого компонента списка.

Вертикальный список делится на четыре разновидности списков: Selection List, Menu List, Markable List и Multiselection List.

#### **Список Selection**

Это простейшая разновидность списка. В списке *Selection* на экране отображаются построчно компоненты списка, и пользователь может выбирать из предложенных ему данных. После выбора одного из элементов списка и выполнения команды, открывающей данный компонент, как правило, происходит переход на новый экран с определенными данными для этого элемента списка.

#### **Список Menu**

Список *Menu* формируется на основании простого списка Selection, но для подэкранной клавиши Options создается дополнительное меню, через которое пользователь выполняет заданную команду для выбранного элемента списка.

#### **Список Markable**

Эта разновидность списка тоже построена на основе простого списка Selection, но с возможностью маркировки одного или более элементов списка. Посмотрите на рис. 8.6, где изображен список *Markable*.

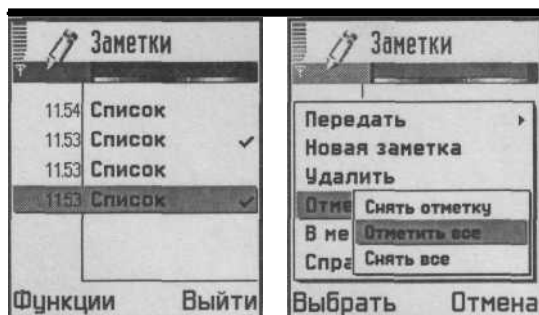


Рис. 8.6. Список Markable

Выбрав один из компонентов списка Markable, с правой стороны от названия элемента появляется графическое изображение в виде галочки, которая сигнализирует о маркированном элементе списка. Маркировка компонента осуществляется с помощью стандартных команд меню: **Mark** (Отметить), **Mark All** (Отметить все), **Unmark** (Снять) или **Unmark All** (Снять все).

### Список Multiselection

Список *Multiselection* очень похож на список Markable и формируется точно таким же образом, но графический элемент, сигнализирующий об отмеченном или снятом элементе списка, находится с левой стороны от названия компонента. Это квадратик небольшого размера с галочкой внутри (если элемент списка выбран). На рис. 8.7 показан список Multiselection. Маркировка списка происходит аналогичным образом, как и в списке Markable.

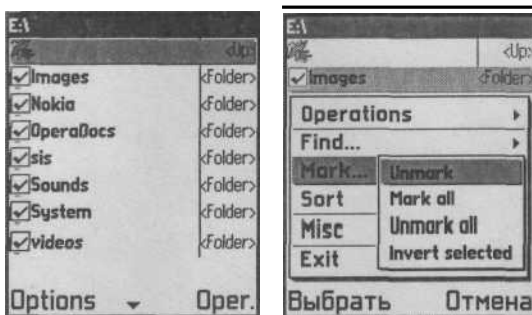


Рис. 8.7. Список Multiselection

### Создание списка

Для того чтобы *создать* вертикальный список, в файле ресурса вашего проекта \*.rss делается объявление:

```

RESOURCE LISTBOX r_list_vertical {
    array_id = r_list_string;
    flags = EAknListBoxSelection;
}

```

Перечисление array id содержит массив текстовых строк типа LBUF, которые определены как компоненты или элементы всего списка. В своем файле ресурса вы должны описать r\_list\_string, создав необходимое количество текстовых строк в списке.

Флаг flags в вертикальном списке декларирует разновидность создаваемого списка, а это четыре перечисленных ранее списка. Для каждой разновидности вертикального списка задан собственный флаг:

- Selection List - EAknListBoxSelectionList;
- Menu List - EAknListBoxMenuList;
- Markable - EAknListBoxMarkableList;
- Multiselection - EAknListBoxMultiselectionList.

Затем в программе создаются дополнительные классы для формирования вертикального списка. Демонстрационный пример, иллюстрирующий работу со списками, будет рассмотрен в конце главы.

Один из дополнительных классов в программе создается специально для того, чтобы определить стиль создаваемого вертикального списка. Посмотрите на рис. 8.5, где показаны два вертикальных списка, но с абсолютно разным оформлением. В системной библиотеке Avkon определено двенадцать классов для оформления стилей вертикальных списков. Создавая свой класс, наследующий возможности одного из системных классов, вы тем самым выбираете оформление для списка. Поскольку существует двенадцать классов, им соответствуют и двенадцать видов оформления. Принцип, по которому оформляются списки, имеет свои определенные закономерности. Посмотрите на рис. 8.8, где схематично показан шаблон для оформления списков.

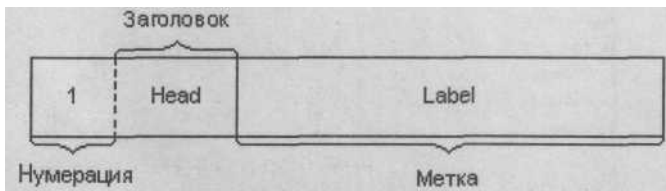


Рис. 8.8. Оформление списков

Если внимательно посмотреть на рис. 8.5, где изображены два списка с различным оформлением, то вы заметите вертикальную линию, разделяющую экран на две части. С правой стороны расположен текст и метка (Label) элемента списка, как показано на рис. 8.8. Вторая часть экрана для каждого списка содержит



две колонки. Нумерация, где проставляются числа, определяющие порядок элементов списка, либо загружается графическое изображение. Заголовок может содержать короткое дополнительное название для одного из элементов. В некоторых системных классах и заголовки, и нумерация интегрируются в один столбик. Для того чтобы задать текст элементу списка, используется следующая конструкция: "\tHead\tLabel"

- число «1» - нумерация для элемента списка;
- \t - непечатаемый символ, разделяющий между собой колонки;
- Header - заголовок для элемента списка, находится в левой части экрана;
- Label - это основной текст, располагается в правой и большей части экрана.

Теперь давайте рассмотрим все двенадцать системных классов Avkon, с помощью которых создается оформление для вертикального списка.

- CAknSingleStyleListBox содержит только текст в области метки, определяется по типу "\tLabel".
- CAknSingleGraphicsStyleListBox - в колонках нумерации и заголовка загружается иконка размером 13x13 пикселей, а область метки содержит текст. Определяется по типу "1\tLabel".
- CAknSingleNumberStyleListBox - в колонках нумерации и заголовка загружается числовое значение для нумерации элементов списка. Область метки содержит текст и определяется по типу "1\tLabel".
- CAknSingleHeadingStyleListBox - в двух колонках нумерации и заголовка находится один заголовок, а в области метки — текст. Элемент списка задается: "\tHead\tLabel".
- CAknSingleGraphicsHeadingStyleListBox - в колонку нумерации загружается иконка 13x13 пикселей, в заголовок — содержание заголовка и в метку - строка текста. Задается как "O\tHead\tLabel".
- CAknSingleLargeStyleListBox - в нумерацию и заголовок загружается иконка размером 42x29 пикселей, метка содержит строку текста. Определяется "\tLabel".
- CAknSingleNumberHeadingStyleListBox - нумерованный элемент списка с заголовком, заданный по типу "\tHead\tLabel".

Q CAknDoubleStyleListBox - этот стиль оформления списка содержит две строки текста - одна под другой для каждого элемента списка. Задается "\tLabel\tLabel2".

- CAknDoubleStyle2ListBox - оформление элемента списка происходит как при использовании предыдущего класса, но с большим по размеру шрифтом текста.
- CAknDoubleNumberStyleListBox оформляется как же, как и класс CAknDoubleStyleListBox, но дополнительно содержит нумерацию элементов списка. Задается "\tLabel\tLabel2".

U CAknDoubleTimeStyleListBox оформляется как предыдущий элемент списка, но вместо нумерации содержит время. Задается "08.00\tAM (илиPM)\tLabel\tLabel2".

О CAknDoubleLargeStyleListBox использует точно такой же стиль оформления, но вместо времени загружается иконка размером 42x36 пикселей. Определяется по типу "\tLabel\tLabel2".

## 8.7.2. Список Grid

Английское слово Grid обозначает сетку, нечто подобное используется при создании этого вида списка. При создании списка Grid в клиентской области экрана формируется набор графических элементов в виде квадратов, равномерно распределенных по всей области экрана, как, например, простой календарь на месяц. Перемещаться по такому виду списка можно с помощью джойстика или клавиш **Up**, **Down**, **Left** и **Right**. Когда один из элементов списка находится в фокусе курсора, пользователь нажимает клавишу OK или с помощью меню выполняет

команду, это приводит к заданным действиям. Так же как и вертикальный список, Grid имеет четыре разновидности:

- D Selection List - флаг EAknListBoxSelecti-onGrid;
- Menu List - флаг EAknListBoxMenuGrid;
- Markable List- флаг EAknListBoxMarkableGrid;
- Multiselection List- флаг EAknListBoxMarkableGrid.

Для оформления стиля списка Grid задействуются три системных класса уровня Avkon.

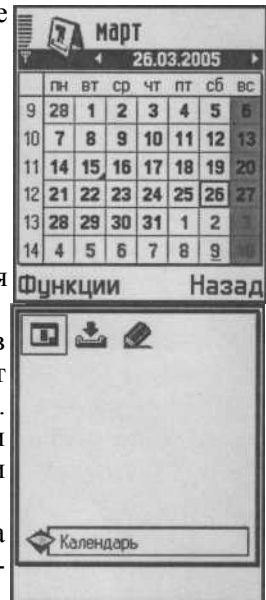
Класс CAknCaleMonthlyStyleGrid создает список в виде календаря. Класс CAknPinbStyleGrid формирует список по принципу программы **Актив**, как на рис. 8.8. Класс CAknGMSStyleGrid создает список в виде сетки из графических иконок. На рис. 8.9 изображены все три стиля оформления для списка Grid.

Чтобы *создать* список вида Grid, в файле ресурса приложения \*.rss необходимо описать вид и стиль создаваемого списка Grid, например:

```
RESOURCE GRID    r_grid_list
{
    array_id = r_grid_string;
    flags = EAknListBoxMarkableGrid;
    style = r_grid_style; }

```

Рис. 8.9. Стили оформления списка Grid



Первые два параметра имеют аналогичные свойства, как и при создании вертикального списка. Параметр `style` задает, как на экране будут представлены элементы списка `Grid`. И еще необходимо описать созданный ресурс `r_grid_style`, например, следующим образом:

```
RECURSE GRID_STYLE    r_grid_style
{
primaryscroll = EAknGridFollowsItemsAnLoops;
secondaryscroll = EAknGridFollowsItemsAnLoops;
layoutflags = EAknGridHorizontalOrientation|
              EAknGridLeftTopRight|
              EAknGridTopTopBottom;
itemsinprimaryorient = 3;
itemsinsecondaryorient = 4;
gapwidth = 4; gapheight = 4; height
= 45; width = 45;
```

- `primaryscroll` определяет способ горизонтальной навигации по элементам списка;
- `secondaryscroll` — вертикальная навигация по списку `Grid`;
- `layoutflags` - это флаги, на основании которых происходит построение элементов списка `Grid`. Например, в представленном примере построение происходит горизонтально, то есть построчно. Построение так же происходит слева на право от верхнего правого угла списка `Grid`;
- `itemsinprimaryorient` - число элементов списка по горизонтали;
- `itemsinsecondaryorient` - число элементов списка по вертикали;
- `gapwidth` и `gapheight` - расстояние в пикселях между элементами списка, то есть между иконками;
- `height` и `width` — высота и ширина загружаемой иконки элемента списка.

### 8.7.3. Спуск *Setting*

На основании списка `Setting` в программе создается механизм настроек опций или конфигураций для приложения. Список вида `Setting` имеет вложение в виде дополнительного экрана, где и происходит настройка конфигурации для программы. Выбрав один из элементов списка `Setting`, пользователь попадает в дополнительное окно, где может произвести определенные опции. Посмотрите на рис. 8.10, где изображен список вида `Setting`.

На рис. 8.10 происходит настройка точки доступа в Интернет по каналу GPRS. Выбрав из списка `Setting` компонент **Канал данных**, вы попадете на экран с возможностью выбора канала данных из списка. То есть существует титульная



Рис. 8.10. Список Setting

панель для элемента списка и вложенное окно, на которое попадает пользователь при выборе этого элемента списка. Для двух подэкранных клавиш назначаются команды ОК и Отмена. В настройках точки доступа в Интернет использовался один из видов списка Setting, а всего существуют десять типов списка для всевозможных настроек. Это тип Volume control, Slider control, Enumerated text, Text editor, Time editor, Date edition, IP editor, Binary switch, Password numeric editor и Password alphabetic editor. Уделим внимание каждому типу списка Setting, тем более, что это один из сложных элементов пользовательского интерфейса. Сначала познакомимся со всеми типами списка Setting, на примере демонстрационной программы Setting List. Эта программа поставляется с SDK серии 60 версии 2.1 и находится в каталоге SDK в папке Series60Ex. И в конце главы на основе этой же программы научимся создавать список Setting.

### Список Volume control

Этот тип списка Setting чаще всего используется для установки громкости звука. Посмотрите на рис. 8.11, где изображен список Volume control.

Вы видите два экрана. Первый — это титульная панель, в этом элементе списка отображается текущая опция, установленная на данный момент. Список Volume control в качестве графических элементов использует небольшие по разме-

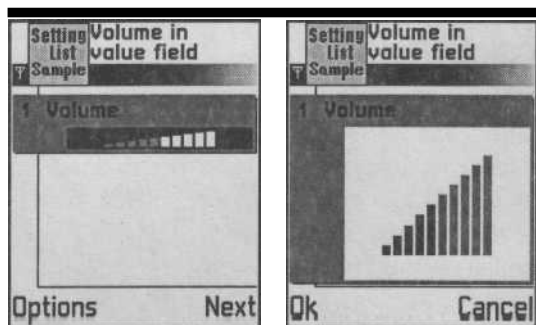


Рис. 8.11. Volume control

рам прямоугольники, показывающие, например, уровень установленной громкости звука. При выборе титульного элемента списка пользователь попадает на вторичный экран, где уже непосредственно и происходит настройка свойств.

В списке Volume control увеличение или уменьшение индикатора уровня производится с помощью джойстика или клавиш **Right** или **Left**. После того, как уровень задан, нажимается клавиша ОК. Заданный уровень индикатора теперь отображается и на титульной панели элемента списка.

Значения для Volume control могут задаваться только в диапазоне от 0 до 10, а любое большее значение будет сброшено к нулю.

### **Список Slider control**

Этот тип списка служит для увеличения или уменьшения какого-либо параметра с помощью бегунка. Посмотрите на рис. 8.12, где изображен список Slider control.

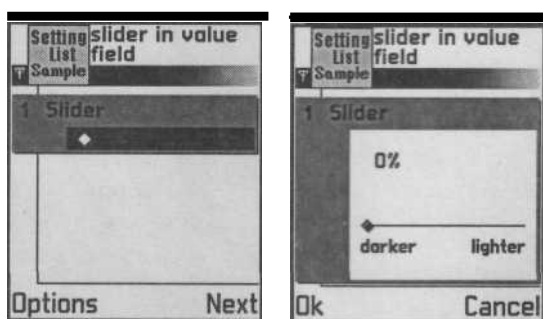


Рис. 8.12. Slider control

Пользователь перемещает бегунок между минимальным и максимальным значением, устанавливая подходящее ему значение. Минимальное и максимальное значения устанавливаются программистом на этапе создания приложения и не имеют определенных ограничений. Перемещение бегунка происходит с помощью джойстика или клавиш телефона **Right** и **Left**.

### **Список Enumerated text**

Список Enumerated text содержит перечисления, одно из которых можно выбрать. Принцип работы основан на использовании переключателей как в Windows. На рис. 8.13 изображен Enumerated text.

Количество текстовых перечислений не ограничено и определяется программистом. Перемещаться по перечислениям можно джойстиком или клавишами **Up** и **Down**. При выборе перечисления, курсор помещается на необходимый текст, нажимается клавиша ОК и происходит установка заданного параметра. Титульный лист типа Enumerated text показывает установленное на данный момент значение, что закономерно для всех типов списка Setting.

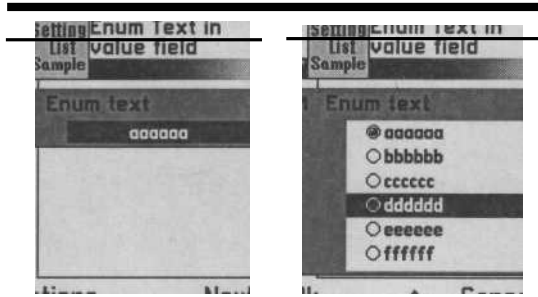


Рис. 8.13. Enumerated text

### Список Text editor

Этот тип списка представляет обыкновенный текстовый редактор, но со значительными ограничениями длины текста. Список Text editor обычно используется для создания коротких названий, которые даются каким-то опциям в программе. На рис. 8.14 показан Text editor. Для набора символов в текстовом редакторе задействуются все клавиши телефона и доступны любые операции по редактированию текста. В том случае, если в Text editor не содержится ни одного символа, на титульной панели будет фигурировать слово None.

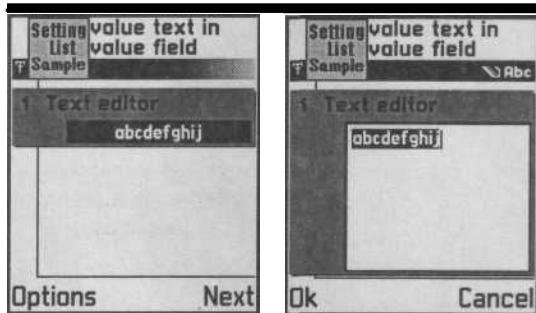


Рис. 8.14. Text editor

### Список Time editor

Как видно из названия этого типа списка, он используется для редакци и времени в программе. На рис. 8.15 показан Time editor. В области редакции Time editor располагаются десятичные цифры, разделенные между собой на блоки для часов, минут, секунд и PM или AM. Для редакции используются все клавиши, переход по значениям осуществляется джойстиком или клавишами **Right** или **Left**.

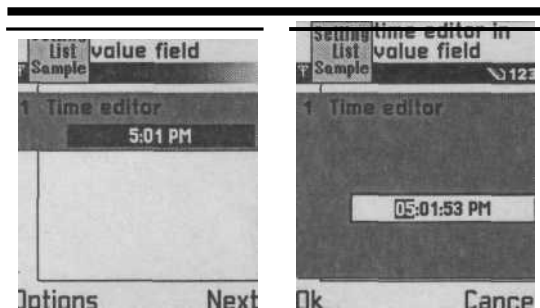


Рис. 8.15. Time editor

### **Список Date editor**

Абсолютно идентичен списку Time editor, с той лишь разницей, что установка значений происходит для числа, месяца и года. Область редакции разделена на три независимых блока, перемещение по которым и редактирование происходит с помощью клавиш телефона. Титульная панель списка *Date editor* показывает установленную пользователем дату. На рис. 8.16 показан список Date editor.

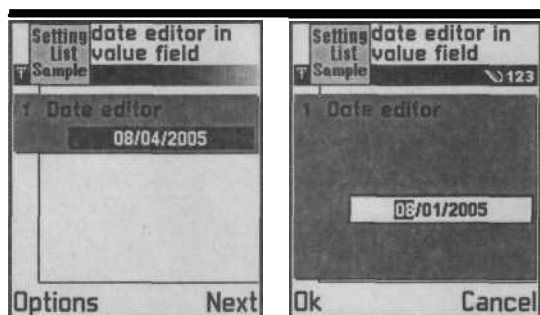


Рис. 8.16. Date editor

### **Список IP editor**

Область редакции этого типа списка основана на двух предыдущих типах списков. Но вместо двухзначных значений используются трехзначные числа. Блок чисел разделен точкой и всего имеются четыре таких блока. С помощью списка IP editor, задается IP-адрес. Редакция и перемещение в IP editor производится всеми клавишами телефона. На рис. 8.17 показан список IP editor.

### **Список Binary switch**

Список *Binary switch* применяется для определения логических значений по принципу Да-Нет, Включить-Выключить или Верно-Не верно и так далее. Посмотрите на рис. 8.18, где показан список Binary switch. Этот тип списка не имеет

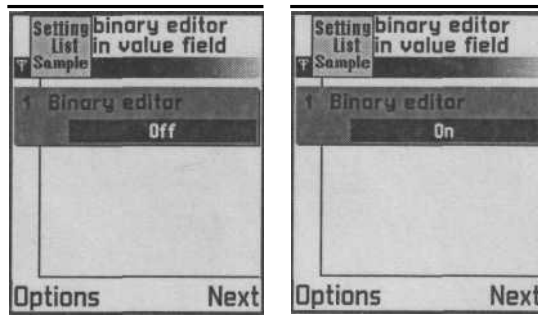


Рис. 8.17. IP editor

дополнительно вложенного экрана редакции. Всего могут быть только два значения в списке Binary switch, поэтому одно из них задано по умолчанию. Для изменения значения в списке Binary switch курсором выделют необходимый элемент списка и нажимают клавишу **ОК** или **Выбор**, после чего значение изменяется на противоположное.

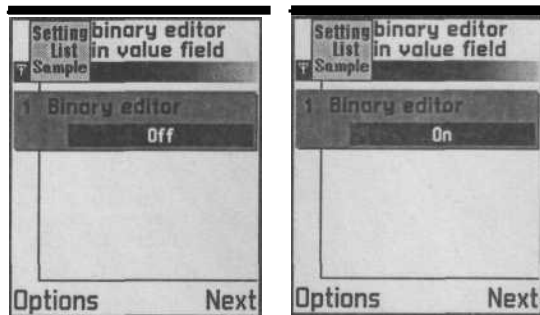


Рис. 8.18. Binary switch

## **Список Password editor**

Этот тип списка используется для задания пароля. Список Password editor может быть двух разновидностей: numeric и alphabetic. Это два идентичных редактора, но при использовании alphabetic, вводя пароль, вы можете на несколько секунд увидеть набираемую на клавиатуре букву или цифру. Затем она камуфлируется звездочкой. В случае с numeric такой возможности нет, и при наборе пользователь не видит введенных символов. Все цифры или буквы изначально камуфлируются звездочками. На титульной панели списка Password editor показаны звездочки, скрывающие набранный пароль. На рис. 8.19 изображен Password editor.



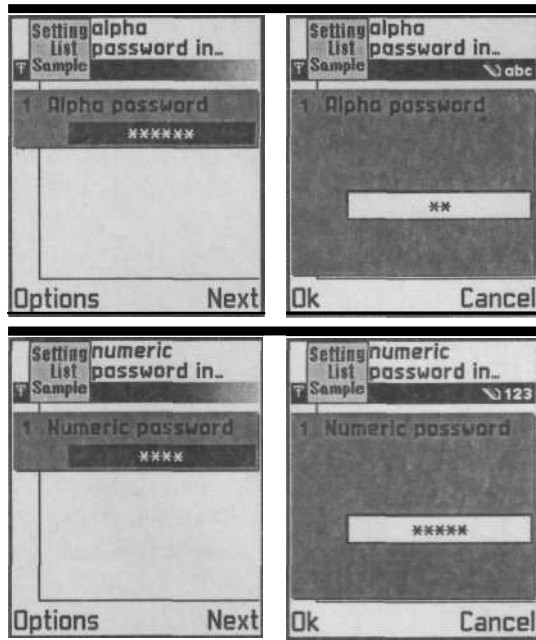


Рис. 8.19. Password editor

#### 8.7.4. Демонстрационный пример Setting List

Пример Setting List, находящийся в каталоге SDK серии 60 в папке \Example, как нельзя лучше демонстрирует все возможности списка Setting. В этом примере рассматривается работа со всеми типами списка Setting, показывается обработка событий, полученных от пользователя с клавиш телефона и представлен общий механизм смены экранов или перехода в приложении. В папке \Series60Ex директории SDK находится папка settinglist, рассмотрим структуру папок и кодов примера Setting List:

- \aif - папка с AIF ресурсами;
  - aknexsettinglistaif.rss - информационный файл AIF;
  - две иконки и две маски для рабочего стола телефона и для панели Status pane;
- \group - папка с проектными файлами;
  - aknexsettinglist.mmp - файл MMP;
  - aknexsettinglist.pkg - файл PKG;
  - aknexsettinglist.rss - файл ресурса;
  - aknexsettinglistcaption.rss - файл ресурса для заголовков программы;
  - bld.ini - файл для сборки и компиляции проекта;
- \inc - папка с заголовочными файлами проекта;
  - aknexsettinglist.hrh - заголовочный файл ресурсов программы;

- aknexsettinglist.loc - файл локализации;
- aknexsettinglistapp.h — заголовочный файл приложения;
- aknexsettinglistappui.h - заголовочный файл для AppUi;
- aknexsettinglistdocument.h - бланк документа;
- aknexsettinglistview.h - заголовочный файл представления вида;
- G aknexsettinglistcontainer.h — заголовочный файл класса AknExSettingListContainer;
- aknexsettinglistbox.h - заголовочный файл класса AknExSettingListBox;
- aknexsettinglistitemdata.h - заголовочный файл класса AknExSettingListItemData;
- \src - папка с исходными кодами проекта;
  - aknexsettinglistapp.cpp - файл реализации класса приложения;
  - aknexsettinglistappui.cpp - файл реализации AppUi;
  - aknexsettinglistdocument.cpp - файл реализации класса Document;
  - aknexsettinglistview.cpp - представление вида;
  - aknexsettinglistcontainer.cpp - реализация класса контейнера;
  - aknexsettinglistbox.cpp - файл реализации класса AknExSettingListBox;
  - aknexsettinglistitemdata.cpp - файл реализации класса AknExSettingListItemData.

### **Файл A/F**

Информационный файл AIF-ресурса содержит стандартное определение для заголовка программы на английском языке. В программе используется свой уникальный идентификатор приложения UID3.

*/\**

---

```
* AknExSettingListaif.rss
* AknExSettingList
* Copyright (c) 2003 Nokia. All rights reserved.
```

*\*/*

```
...
#include <aiftool.rh^
```

```
RESOURCE AIF DATA
```

```
{
    app_uid = 0x10005C2 9;
    caption_list =
    {
        CAPTION
        {
            code = ELangEnglish;
```

```
caption = "AKNEXSETTINGLIST";
```

```
num_icons = 2;  
embeddability=KAppNotEmbeddable;  
newfile=KAppDoesNotSupportNewFile; }
```

### **Файл ресурса для заголовка программы**

Файл AknExSettingList\_caption.rss содержит объявления для названий программы на рабочий стол телефона и панель Status pane. В качестве названий используются макросы, расшифровка которых указана в файле локализации приложения.

```
/*
```

---

---

```
* AknExSettingList_captions.rss  
* AknExSettingList  
* Copyright (c) 2003 Nokia. All rights reserved.
```

```
*/
```

```
◆include "AknExSettingList.loc"  
#include <apcaptionfile.rh>
```

```
RESOURCE CAPTION_DATA  
{  
    caption = qtn_apps_tasp_list;  
    shortcaption = qtn_apps_tasp_grid; }
```

### **Заголовочный файл ресурсов HRH**

В файле AknExSettingList.hrh создаются три перечисляемых типа: TАknEx-SettingListMenuCommands - для команд меню программы, TАknExSettingItems - для идентификации каждого типа списка Setting и TАknExSettingListCba - для команды Next, назначенной правой подэкранной клавише телефона.

```
/*
```

---

---

```
* AknExSettingList.hrh  
* AknExSettingList  
* Copyright (c) 2003 Nokia. All rights reserved.
```

```
*/
```

```

#ifndef AKNEXSETTINGLIST_HRH
#define AKNEXSETTINGLIST_HRH

enum TAknExSettingListMenuCommands {
    EAknExSettingListCmdEmptyOutline = 0x6000,

EAknExSettingListCmdSettingList,
EAknExSettingListCmdOutline01,
EAknExSettingListCmdOutline02,
EAknExSettingListCmdOutline03,
EAknExSettingListCmdOutline04,
EAknExSettingListCmdOutline05,
EAknExSettingListCmdOutline06,
EAknExSettingListCmdOutline07,
EAknExSettingListCmdOutline08,
EAknExSettingListCmdOutline09,
EAknExSettingListCmdOutline10 };

enum TAknExSettingItems
{
    EAknExSettingText,
    EAknExSettingVolume,
    EAknExSettingEnumText,
    EAknExSettingSlider,
    EAknExSettingDate,
    EAknExSettingTime,
    EAknExSettingBinary,
    EAknExSettingPassAlph,
    EAknExSettingPassNumber,
    EAknExSettingIpAddress
};

enum TAknExSetListCba {
    EAknExSetListCbaCmdNext = 0x6300
};

#endif // AKNEXSETTINGLIST_HRH

```

В перечисляемом типе TAknExSettingListMenuComraands происходит объявление набора целочисленных констант или списка перечислений для команд меню приложения. При запуске программы Setting List пользователь попа-

дает на титульную станицу программы. Для того чтобы перейти или открыть экран с необходимым типом списка Setting, а это может быть любой из десяти доступных списков, нужно выбрать одноименную команду меню, как показано на рис. 8.20. После этого пользователь попадает на экран с одним из типов списка Setting.

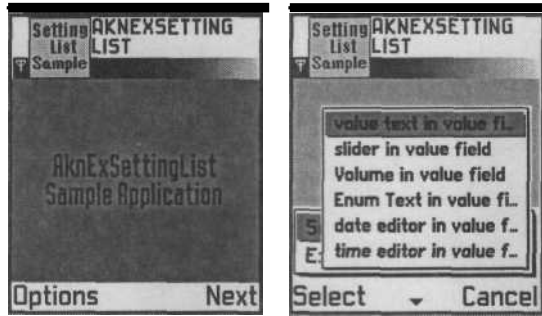


Рис. 8.20. Работа программы Setting List

Тип `TAknExSettingItems` содержит перечисления для идентификации всех десяти видов списка Setting. Об этом легко догадаться по названиям констант, чего не скажешь о придуманных названиях командам меню. Идентификаторы необходимы для создания списков в файле ресурса приложения и при рассмотрении файла `AknExSettingList.hrh` вы в этом убедитесь.

При попадании на титульный экран одного из типов списка, например, Password editor (рис. 8.19), правой подэкранной клавишей назначается команда **Next**. В перечисляемом типе `TAknExSettingListCba` происходит объявление команды **Next**. Присвоенное значение `0x6300` этой команде - это стандартное системное значение для команд типа **Next**, **Back**.

### **Ресурсы программы Setting List**

В Symbian OS традиционно файлы ресурсов находятся в папке `\group` и имеют расширение `*.rss`. Файл `AknExSettingList.rss` содержит объявления ресурсов для приложения Setting List. Этот файл очень большой по размеру, поэтому в книге приводятся по частям только его важные моменты. Для изучения полной спецификации файла `AknExSettingList.rss` откройте каталог SDK и перейдите в папку `\Series60Ex\settinglist\group`.

В начале исходного кода происходит подключение системных файлов, зада-СТСЛ ПДСНТИФИКАТор ресурса и декларируется большой набор констант.

/\*

\* AknExSettingList.rss

\* AknExSettingList

\* Copyright (c) 2003 Nokia. All rights reserved.

---

```
*/
// идентификатор ресурса
NAME EXPG

// подключаем системные библиотеки
◆include <eikon.rh>
◆include <eikon.rsg>
◆include <avkon.hrh>
◆include <avkon.rsg>
◆include <avkon.rh>
◆include <avkon.mbg>
◆include <avkon.loc>
◆include "aknexsettinglist.hrh"
◆include "aknexsettinglist.loc"

// Text Setting
◆define TEXT_SETTING_PAGE_NUMBER 1 ◆define
TEXT_SETTING_PAGE_WIDTH 9 ◆define
TEXT_SETTING_PAGE_LINES 5 ◆define
TEXT_SETTING_PAGE_MAXLENGTH10 10 ◆define
TEXT_SETTING_PAGE_MAXLENGTH20 20 ◆define
TEXT_SETTING_PAGE_MAXLENGTH30 30

// Slider Setting
◆define SLIDER_SETTING_PAGE_NUMBER 1
◆define SLIDER_SETTING_PAGE_MINVALUE 0
◆define SLIDER_SETTING_PAGE_MAXVALUE 100
◆define SLIDER_SETTING_PAGE_STEP 1

// Volume Setting
◆define VOLUME_SETTING_PAGE_NUMBER 1
◆define VOLUME_SETTING_PAGE_VALUE 8

// for Time Setting Page ◆define
TIME_SETTING_PAGE_NUMBER 1 ◆define
TIME_EDITOR_MIN_SECOND 0 ◆define
TIME_EDITOR_MIN_MINUTE 0 ◆define
TIME_EDITOR_MIN_nOUR 0 ◆define
TIME_EDITOR_MAX_SECOND 59 ◆define
TIME_EDITOR_MAX_MINUTE 59 ◆define
TIME_EDITOR_MAX_HOUR 23
```

```

// Duration Setting
#define DURATION_SETTING_PAGE_NUMBER 1
#define DURATION_EDITOR_MIN_SECOND 0
#define DURATION_EDITOR_MAX_SECOND 3000

// Alpha Password Setting
#define ALPHA_PASSWORD_SETTING_PAGE_NUMBER 1
#define ALPHA_PASSWORD_LENGTH_OF_STRING 8

// Numeric Password Setting
#define NUMERIC_PASSWORD_SETTING_PAGE_NUMBER 2
#define
NUMERIC_PASSWORD_OL09_SETTING_PAGE_NUMBER 1
#define NUMERIC_PASSWORD_LENGTH_OF_STRING 8

// IP Editor Setting
#define IP_EDITOR_MIN_FIELD_VALUE 20
#define IP_EDITOR_MAX_FIELD_VALUE 200

```

Все объявленные константы подключаются в программу с помощью директивы `#define`.

При создании типов списка `Setting` задействуются различные целочисленные значения, например, для типа `Volume` необходимо определить минимальное и максимальное значение звука. Это можно сделать непосредственно в месте создания и описания свойств `Volume control` с помощью цифр, но значительно удобнее создать константу с заданным значением и использовать ее по всему исходному коду. Из созданных констант видно, что предусмотрен широкий спектр всевозможных значений. При создании конкретных типов списка `Setting` мы обязательно вспомним обо всех константах, а сейчас нет смысла обсуждать каждую, потому что не известна пока их область применения.

Макрос ресурса `EIK_APP_INFO` создает в программе на левой подэкранной клавише `Options` меню. Одна из команд меню содержит вторичное меню (см. рис. 8.20).

---

```

// EIK_APP_INFO
// _____
RESOURCE EIK APP INFO
{
    menubar = r_aknexsettinglist_menubar;
    //cba = R AVKON SOFTKEYS OPTIONS BACK;
    cba = r_aknexsettinglist_cba_options_next;
}
// _____
// r_aknexsettinglist_menubar

```

```

// pecypc Menu Bar
//-----
RESOURCE MENU_BAR r_aknexsettinglist_menubar
{
    titles = {
        MENUJTITLE
            {
                menu pane = r_aknexsettinglist_menu;
                txt = qtn_aknexsettinglist_menubar;
            }
    }
}

//-----
// r_aknexsettinglist_menu
// меню Menu pane
//-----
RESOURCE MENU_PANE r_aknexsettinglist_menu
{
    items = {
        MENU_ITEM
            {
                command = EAknExSettingListCmdSettingList;
                cascade =
                    r_aknexsettinglist_settinglist_cascade; flags =
                    EEikMenuItemSeparatorAfter; txt =
                    qtn_aknexsettinglist_menu_setlist; }, MENU_ITEM //
                Menu "Exit"
            {
                command = EAknCmdExit;
                txt = qtn_aknexsettinglist_menu_exit;
            }
    }
}

//-----
// r_aknexsettinglist_settinglist_cascade //
//вторичное меню
//-----
RESOURCE MENU_PANE
r_aknexsettinglist_settinglist_cascade {
    items =

```



MENUITEM

```
command = EAknExSettingListCmdOutline01;  
txt = qtn_aknexsettinglist_outline01;
```

},

MENU\_ITEM

{

```
command = EAknExSettingListCmdOutline02;  
txt = qtn_aknexsettinglist_outline02;
```

┌

MENU ITEM

{

```
command = EAknExSettingListCmdOutline03;  
txt = qtn_aknexsettinglist_outline03;
```

MENUITEM

```
command = EAknExSettingListCmdOutline04;  
txt = qtn_aknexsettinglist_outline04;
```

MENUITEM

```
command = EAknExSettingListCmdOutline05;  
txt = qtn_aknexsettinglist_outline05;
```

MENUITEM

```
command = EAknExSettingListCmdOutline06;  
txt = qtn_aknexsettinglist_outline06;
```

MENUITEM

```
command <■ EAknExSettingListCmdOutline0?;  
txt = qtn_aknexsettinglist_outline07;
```

MENUITEM

```
command = EAknExSettingListCmdOutline08;  
txt = qtn_aknexsettinglist_outline08;
```

MENUITEM

```
command = EAknExSettingListCmdOutline0!?;  
txt = qtn_aknexsettinglist_outline09;
```

## MENUITEM

```
command = EAknExSettingListCmdOutline10;  
txt ■ qtn_aknexsettinglist_outline10;
```

В ресурсе MENU\_BAR создается меню r\_aknexsettinglist\_menu, расшифровка которого находится в ресурсе MENU\_PANE. Меню r\_aknexsetting-list\_menu в качестве команды (command) использует перечисление EAknExSettingListCmdSettingList, объявленное в файле AknSettingList.hrh. Затем идет спецификатор cascade, который указывает на создание вторичного меню r\_aknexsettinglist\_settinglist\_cascade. Обратите внимание на флаг (flags), используемый при создании меню. С помощью флажка EEikMenu-ItemSeparatorAfter происходит отключение активного в текущий момент пункта меню. Если пользователь выбрал один из пунктов меню, то он находится на экране, представляющем данную команду, и поэтому эта команда деактивируется и становится недоступной пользователю.

В спецификаторе txt должна находиться строка текста для этого меню. Для удобства локализации программы используется макрос, расшифровка которого находится в файле AknExSettingList.loc. Всем командам, используемым в программе Setting List, назначаются макросы для удобства локализации приложения. В меню программы также предусмотрена команда выхода **Exit**. Вторичное MeHior\_aknexsettinglist\_settinglist\_cascade содержит десять команд для каждого из типов списка Setting. Команды объявляются с помощью спецификатора MENU\_ITEM и содержат идентификатор команды (command) и строку текста (txt), которая будет показана во вторичном меню.

Далее в файле AknExSettingList.rss идут множественные объявления текстовых строк для использования в различных местах программы. Текстовым строкам задаются макросы, расшифровки которых находятся в файле локализации программы Setting List. По окончании работы с текстом происходит создание типов списка Setting, первый - *Text editor*.

```
, I-----  
// r_aknexsettinglist_text_setting_page  
// для Outline04  
// _____  
RESOURCE AVKON_SETTING_PAGE  
r_aknexsettinglist_text_setting_page {  
-  
    number = TEXT SETTING PAGE NUMBER; label  
    = qtn_aknexsettinglist_text_title;
```

```
type = EEikCtEdwin;
editor_resource_id = r_aknexsettinglist_edwin;
```

```
RESOURCE EDWIN r_aknexsettinglist_edwin
{
    width = TEXT_SETTING_PAGE_WIDTH; lines =
    TEXT_SETTING_PAGE_LINES;    maxlength =
    TEXT_SETTING_PAGE_MAXLENGTH20;
```

Список Text editor предназначен для команды меню Outline04, выполнив которую пользователь откроет этот тип списка. Объявление RESOURCE AVKON\_SETTING\_PAGE декларирует создание списка типа Setting, а от количества и значений задействованных параметров уже будет зависеть, какой из типов списка создается.

- number - это порядковое число будет отображаться для данного элемента списка в верхнем левом углу его титульной панели. Константе TEXT\_SETTING\_PAGE\_NUMBER в начале исходного кода было присвоено значение равное единице.
- label - строка текста для названия этого элемента списка. Используется ресурс типа RESOURCE TBuf256 для удобства локализации программы.
- type - это тип создаваемого списка Setting. С помощью типа EEikCtEdwin формируется тип Text editor списка Setting.
- editor\_resource\_id - идентификатор создаваемого ресурса. В Text editor задан r\_aknexsettinglist\_edwin, описав который, вы создадите список Text editor с необходимыми свойствами.

В ресурсе RESOURCE EDWIN r\_aknexsettinglist\_edwin указывается ширина текстового поля (width), равная девяти символам. Количество строк (lines) равно пяти строкам, максимальное количество введенных символов (maxlength) равно двадцати.

Дальше происходит создание типа *Slider control* списка Setting.

```
// _____
//   r_aknexsettinglist_slider_setting_page
//   для outline05
// _____
```

```
RESOURCE AVKON_SETTING_PAGE
r_aknexsettinglist_slider_setting_page
{
    number = SLIDER_SETTING_PAGE_NUMBER; label =
    qtn_aknexsettinglist_slider_title; type = EAknCtSlider;
```

```
editor_resource_id = r_aknexsettinglist_slider;
```

```
RESOURCE SLIDER r_aknexsettinglist_slider {  
    layout = EAknSettingsItemSliderLayout;  
    minvalue = SLIDER_SETTING_PAGE_MINVALUE;  
    maxvalue = SLIDER_SETTING_PAGE_MAXVALUE;  
    Step = SLIDER_SETTING_PAGE_STEP;  
    valuetype = EAknSliderValuePercentage;  
    minlabel = qtn_slider_minlabel;  
    maxlabel = qtn_slider_maxlabel;
```

Объявление ресурса для типа Slider control идентично объявлению Text editor, но в параметре type содержится значение EAknCtSlider, определяющее создание списка Slider control. Такой вид исходного кода закономерен для создания всех типов списка Setting. Идентификатор ресурса r\_aknexsettinglist\_slider описывает свойства создаваемого типа списка Slider control.

- layout - отображение заголовка в списке Slider control.
- minvalue - минимальное значение, от которого происходит отчет в программе. Это значение равно 0.
- maxvalue - максимальное значение, до которого можно передвинуть бегунок Slider. В программе оно равно 100.
- step - это шаг, то есть минимальное значение, на которое можно за один раз передвинуть бегунок. Равно единице.
- valuetype - задает, как именно установленное значение будет представлено на экране - в виде чисел, процентного соотношения, фракций или геометрической фигуры.
- minlabel, maxlabel - макрос локализации, определяющий минимальное и максимальное значения для устанавливаемых свойств.

Затем в файле AknExSettingList.rss создается тип списка *Volume control*.

---

```
// r_aknexsettinglist_volume_setting_page  
// For outline06  
//
```

---

```
RESOURCE AVKON_SETTING_PAGE  
r_aknexsettinglist_volume_setting_page
```

**Г**

```
number =  
VOLUME_SETTING_PAGE_NUMBER; label =  
qtn_aknexsettinglist_volume_title; type =  
EAknCtVolumeControl;
```

```
editor_resource_id = r_aknexsettinglist_volume;
```

```
RESOURCE VOLUME r_aknexsettinglist_volume {  
    flags ■ ESettingsVolumeControl;  
    value = VOLUME_SETTING_PAGE_VALUE;
```

Тип создаваемого списка Volume control задается перечислением EAknCtVolumeControl. В ресурсе RESOURCE VOLUME r\_aknexsettinglist\_volume описываются свойства создаваемого типа списка:

- flags - это флаг, указывающий на стиль создаваемого типа списка. В программе используется ESettingsVolumeControl, формирующий стиль контроля в виде прямоугольников, как показано на рис. 8.11.
- value - максимальное значение для Volume control, в программе равно целому значению 8.

Тип списка *Time editor* создается на основании EEikCtTimeEditor в ресурсе r\_aknexsettinglist\_time\_setting\_page.

---

```
// r_aknexsettinglist_time_setting_page  
// для outline07.  
// -----
```

```
RESOURCE AVKON_SETTING_PAGE  
r_aknexsettinglist_time_setting_page  
{  
    number = TIME_SETTING_PAGE_NUMBER;  
    label = qtn_aknexsettinglist_time_title;  
    type = EEikCtTimeEditor;  
    editor_resource_id = r_aknexsettinglist_time_editor;
```

```
RESOURCE TIME_EDITOR r_aknexsettinglist_time_editor {  
    minTime = TIME  
    {  
        second = TIME_EDITOR_MIN_SECOND;  
        minute = TIME_EDITOR_MIN_MINUTE;  
        hour = TIME_EDITOR_MIN_HOUR;  
    };  
    maxTime = TIME
```

```

second = TIME_EDITOR_MAX_SECOND;
minute = TIME_EDITOR_MAX_MINUTE;
hour = TIME_EDITOR_MAX_HOUR;

```

В ресурсе TIME\_EDITOR r\_aknexsettinglist\_time\_editor содержатся описания минимального и максимального допустимого значения времени. В minTime значится second, minute и hour и они равны 0, а в maxTime, second и minute равны 59, а hour равен значению 23. Все эти значения заданы с помощью констант в начале исходного кода файла AknExSettingList.rss.

Дальше в исходном коде происходит создание списков типа Alphabetic Password editor и Numeric Password editor.

```

// _____
//   r_aknexsettinglist_alpha_password_setting_page
//   для outline08.
// _____
//
RESOURCE AVKON_SETTING_PAGE
r_aknexsettinglist_alpha_password_setting_page
{
    number = ALPHA_PASSWORD_SETTING_PAGE_NUMBER;
    label = qtn_aknexsettinglist_alpha_title;
    type = EEikCtSecretEd;
    editor_resource_id = r_aknexsettinglist_alpha_password;
}

RESOURCE SECRETED r_aknexsettinglist_alpha_password {
    num_letters = ALPHA_PASSWORD_LENGTH_OF_STRING;
}

// _____
//   r_aknexsettinglist_ol09_numeric_password_setting_page
//   для outline09.
// _____
//
RESOURCE AVKON_SETTING_PAGE
r_aknexsettinglist_ol09_numeric_password_setting_page
{
    number =
    NUMERIC_PASSWORD_OLQ2_SETTING_PAGE_NUMBER;
    label = qtn_aknexsettinglist_numeric_title; type =
    EAknCtNumericSecretEditor;
}

```

```
editor_resource_id =
r_aknexsettinglist_numeric_password.Γ
```

```
RESOURCE NUMSECRETED r_aknexsettinglist_numeric_password
{
    num_code_chars = NUMERIC_PASSWORD_LENGTH_OF_STRING;
```

Конструкция кода, создающая эти два списка практически одинакова. Для *Alphabetic Password editor* используется тип `EEikCtSecretEd`, тогда как для *Numeric Password editor*- тип `EAKnCtNumericSecretEditor`. В списке типа *Alphabetic* при вводе символов на несколько секунд видны набираемые цифры или буквы. Длина вводимых символов ограничена значением 8 в типе `num_letters`. В типе *Numeric* изначально все вводимые символы засекречены и камуфлируются звездочками. Количество символов в программе *Setting List* для списка типа *Numeric Password editor* задано так же значением 8 в `num_code_chars`.

Тип *IP editor* предназначен для ввода и редакции IP-адреса. В создании этого типа списка используется `EAKnCtIpEditor`. Описание свойств создаваемого списка находится в ресурсе `RESOURCE IP_FIELD_EDITOR r_aknexsetting-list_ip_editor`.

```
// _____ . _____
// r_aknexsettinglist_ip_editor // For
outlineO.
```

```
RESOURCE IP_FIELD_EDITOR
r_aknexsettinglist_ip_editor
{
    min_field_values = IP_FIELD
    {
        first_field = IP_EDITOR_MIN_FIELD_VALUE;
        second field = IP EDITOR MIN FIELD VALUE;
        third field = IP EDITOR MIN FIELD VALUE;
        fourth_field = IP_EDITOR_MIN_FIELD_VALUE;
    };
    max_field_values = IP_FIELD {
        first_field = IP_EDITOR_MAX_FIELD_VALUE; second_field
        = IP_EDITOR1MAXJIELD_VALU"E; third_field =
        IP_EDITOR_MAX_FIELD_VALUE; fourth_field =
        IP_EDITOR_MAX_FIELD_VALUE;
```

```
flags = 0;
```

## RESOURCE AVKON\_SETTING\_PAGE

```
r_aknexsettinglist_ip_address_setting_page
```

```
{  
    label= qtn_aknexsettinglist_ip_editor_title;  
    type = EAknCtlpFieldEditor;  
    editor_resource_id = r_aknexsettinglist_ip_editor;
```

Для всех значений `min_field_values` задается целочисленное значение 20 в виде константы `IP_EDITOR_MIN_FIELD_VALUE`. Максимальное значение установлено константой `IP_EDITOR_MAX_FIELD_VALUE` и равно 200.

Затем в файле ресурса программы `Setting List` происходит создание массива строк текста для типа *Enumerated text*. В качестве строк текста используются константы и в файле локализации они расшифровываются некоторыми условными названиями.

На этот момент были созданы почти все вложенные страницы для каждого типа списка, в которые пользователь попадает после выбора элемента списка на титульной панели. Теперь надо создать все типы списков с титульными панелями и привязать уже созданные вложенные страницы к соответствующим типам списков. Схема создания титульной панели списка `Setting` строится по следующему принципу. Формируется ресурс `RESOURCE AVKON_SETTING_ITEM_LIST`, где описываются свойства создаваемого списка и в параметре `item` объявляется ссылка для связывания ранее созданных вторичных страниц с титульными панелями списка. Посмотрите, как это делается на примере списков типа `Text editor`, `Volume control` и `Slider control`.

```
// _____  
// r_aknexsettinglist_setting_list_setting_text  
// ListBox( Setting style )  
// для Outlined.  
// _____
```

```
RESOURCE AVKON_SETTING_ITEM_LIST  
r_aknexsettinglist_setting_list_setting_text  
{  
flags= EAknSettingItemNumberedStyle;  
title = qtn_aknexsettinglist_outline01;  
initial_number = 1;  
items =  
    {  
        AVKONSETTINGITEM
```



```
    identifier = EAknExSettingText;
    setting page resource =
        r aknexsettinglist text setting page;
    name = qtn_aknexsettinglist_text_title;
```

---

```
// r aknexsettinglist setting list setting slider
// ListBox( Setting style ) // для Outline02.
```

---

```
RESOURCE AVKON SETTING ITEM LIST
r aknexsettinglist setting list setting slider
{
flags= EAknSettingItemNumberedStyle;
title = qtn aknexsettinglist outline02;
initial number = 1; items = {
    AVKON_SETTING_ITEM
    {
        identifier = EAknExSettingSlider;
        setting page resource =
            r aknexsettinglist slider setting page;
        name = qtn_aknexsettinglist_slider_title;
```

---

```
// r aknexsettinglist setting list setting volume
// ListBox( Setting style ) // для Outline03
```

---

```
RESOURCE AVKON_SETTING_ITEM_LIST
r aknexsettinglist setting list setting volume
{
flags= EAknSettingItemNumberedStyle;
title = qtn_aknexsettinglist_outline03;
initial number ■ 1; items = {
    AVKON_SETTING_ITEM
```

```

{
  identifier = EAknExSettingVolume;
  setting page resource =
    r aknexsettinglist volume setting page;
  name = qtn_aknexsettinglist_volume_title;

```

Как видите, ничего сложного нет, и весь исходный код хорошо читается, рассмотрим эти ресурсы:

- `flags` - флаг, определяющий стиль создаваемого списка;
- `title` - строка текста (в файле локализации находится ее расшифровка);
- `initial_number` - число и порядковый номер списка будут отображаться на титульной панели в левом верхнем углу.

Затем происходит привязка ранее созданных страниц к титульной панели списков в ресурсе `AVKON_SETTING_ITEM` с тремя следующими параметрами:

- `identifier`- идентификатор типа списка находится в файле с расширением `*.hrh`;
- `setting_page_resource` - здесь указывается созданный ранее ресурс, например, для списка типа Volume control это ресурс `r_aknexsettinglist_setting_list_setting_volume`. (Все же лучше избегать длинных названий. Я понимаю, что этот код написан программистами Nokia, но согласитесь, ни читать, ни писать подобные названия неудобно, можно было придумать и попроще).

Вот таким образом происходит создание титульного листа и вложенной страницы для редакции или указания каких-то опций в программе. Аналогичный исходный код используется и для других типов списка Setting, он легко читается и, думается, что вы разберетесь с ним самостоятельно. Файл локализации содержит расшифровку для каждой команды и строки текста в программе Setting List. Приложение создавалось только для английского языка. Поэтому в программе файл с расширением `*.loc` всего один, и внутри него содержится описание текстовых констант. Это еще один способ локализации программы. На данный момент мы рассмотрели все файлы ресурсов и теперь можно переходить непосредственно к исходному коду программы.

Классы Application, Document и AppUi вполне узнаваемы и содержат программный код, изученный нами в *главе 7*. Рассмотрение этих классов мы пропустим и перейдем к оставшимся классам программы Setting List.

### ***Класс CAknExSettingListView***

Класс CAknExSettingListView наследует возможности системного класса CAknView. В примерах с классами программы Test механизм был несколько иной, и мы использовали для этих целей класс CCoeControl. В Symbian OS

можно использовать множество вариантов по созданию программ, один из них реализован в приложении Setting List, а возможности класса CCoEControl реализует в данном случае класс CAknExSettingListContainer. Заголовочный файл класса CAknExSettingListView.h выглядит следующим образом:

```
/*
=====
* CAknExSettingListView
* Copyright (c) 2003 Nokia. All rights reserved.
=====
*/
#ifndef AKNEXSETTINGLISTVIEW_H
#define AKNEXSETTINGLISTVIEW_H

◆include <aknview.h>

class CAknExSettingListContainer;
class CAknExSettingListListbox;
class CAknExSettingListListItemData;

class CAknExSettingListView : public
CAknView {
    public:
        CAknExSettingListView();
        virtual --CAknExSettingListView () ;

    public:
        void DisplayNextOutlineL();
        void DisplayPreviousOutlineL();
        void SetCurrentOutlineId(const Tint aOutlineId);

    public:
        TUid Id() const;
        void HandleCommandL(Tint aCommand);
        Tint CurrentOutlineIdO ;

    private:
        void IndicateTitlePaneTextL(const Tint aCommand);
        void SwapContainerL(TBool aActiveContainer); void
        CreateListBoxL(Tint aResourceId);

    private:
```

```

void DoActivateL(const TVwsViewIds aPrevViewId,
                 TUid aCustomMessageId, const TDesC8&
                 aCustomMessage);
void DoDeactivate();

private:

CAknExSettingListContainer* iContainer;
CAknExSettingListListBox* iListBox;
TBool iActiveContainer;
CAknExSettingListItemData* iData; Tint
iCurrentOutlineId;

#endif

```

Две функции, `DisplayNextOutlineL()` и `DisplayPreviousOutlineL()`, регулируют в приложении переход между экранами на основании идентификатора `Tint iCurrentOutlineId` для текущей позиции выбранной команды меню. Реализация двух этих функций в файле `CAknExSettingListView.cpp` выполнена на основании оператора `switch`. В тот момент, когда выбрана одна из команд меню в программе `Setting List`, открывающая один из типов списка, можно с помощью подэкранной клавиши **Next** или джойстика для команд **Left**, **Right** перейти в программе на следующий экран, отображающий уже другой тип списка `Setting`. При выборе команд **Next** и **Right** для клавиш телефона в работу включается функция `DisplayNextOutlineL()`, производя переход по командам меню на один шаг и открывая новый экран со следующим списком. С помощью функции `DisplayPreviousOutlineL()` и клавиши телефона **Left** пользователь возвращается на шаг назад.

Функция `TUid Id()` `const` возвращает текущий идентификатор для `View`, а функция `SetCurrentOutlineId(const Tint aOutlineId)` устанавливает идентификатор, на основе которого функции `DisplayNextOutlineL()` и `DisplayPreviousOutlineL()` регулируют переход с экрана на экран.

Функция `HandleCommandL(Tint aCommand)` обрабатывает команды в меню программы `Setting List`. В этом случае задействуются две функции `SwapContainerLO` и `CEikChoiceList::CreateListBoxL()`. Функция `SwapContainerL()` устанавливает логическое значение для текущего экрана, то есть если выбран этот экран с типом списка `Setting`, то `TRUE`, а если нет - `FALSE`. Системная функция `CEikChoiceList::CreateListBoxL()` создает непосредственно список для экрана на основании данных ресурса приложений, находящихся в файле `AknExSettingList.hrh`.

Функция `IndicateTitlePaneTextL(const Tint aCommand)` по идентификатору ресурсов формирует новую надпись на титульной панели для каждого из типов списка `Setting`.

Функции DoActivateL () и DoDeactivate () соответственно создают и уничтожают объект класса контейнер. В программе это класс CAknExSettingListContainer, который играет управляющую роль в приложении, регулируя какой из экранов и типов списка будет отображен на экране телефона. В программе может существовать несколько классов типа View, но в один отдельно взятый момент времени, возможно представление только одного из классов типа View и, как правило, класс типа Container регулирует это положение. Зачем нужно несколько классов типа View? Это значительно упрощает программу, например, когда вы создадите игру, у вас будут классы для представления (тип View) экрана с красочным интерактивным меню, экраном опций, отдельным экраном для заставки игры, основным циклом игры. Все это можно определить в одном классе, но гораздо проще создать несколько классов, распределив между ними рабочие функции. Для того чтобы активизировать одно из представлений вида вызывается функция DoActivateL (), а на экране может быть представлен только один класс тип View, поэтому для представления другого класса типа View одной программы, необходимо деактивировать текущий класс функцией DoDeactivate () и так далее для всех компонентов программы.

Класс CAknExSettingListContainer, исходный код которого находится в файлах CAknExSettingListContainer.h и CAknExSettingListContainer.cpp программы SettingList, производный от системного класса CCoEControl осуществляет контроль за представлением вида и событиями, полученными с клавиш телефона. В функции CAknExSettingListContainer::OfferKeyEventL() с помощью ключевых кодов для клавиш Left (EKeyLeftArrow) и Right (EKeyRightArrow) обрабатывается ввод данных, полученных от пользователя.

### **Класс CAknExSettingListListBox**

Класс CAknExSettingListListBox создает список вида Setting, посмотрим на спецификацию этого класса, находящуюся в файле AknExSettingList-ListBox.h.

```
/*  
^ _____ ^  
* AknExSettingListListBox.h  
* AknExSettingList  
* Copyright (c) 2003 Nokia. All rights reserved.  
*/  
  
#ifndef AKNEXSETTINGLISTLISTBOX_H  
#define AKNEXSETTINGLISTLISTBOX_H  
  
#include <AknSettingItemList.h>  
  
class CAknExSettingListItemData;
```

```

class CAknExSettingListView;

class CAknExSettingListListBox : public CAknSettingItemL
{
    public:
        CAknSettingItem* CreateSettingItemL(Tint
            identifier);
        void SetData(CAknExSettingListItemData* aData);
        void SetView(CAknExSettingListView* aView);

    private:
        TKeyResponse OfferKeyEventL(const TKeyEvent&
            aKeyEvent,
                                     TEventCode aType);
        void SizeChangedO ;

    private:
        CAknExSettingListItemData* iData;
        CAknExSettingListView* iView; }; #endif

```

Самая главная функция класса - это CreateSettingItemL (), создающая типы списков Setting. В функции используется управляющий оператор switch, где на основании полученного идентификатора создается тот или иной вид списка, как показано в листинге функции CreateSettingItemL ().

```

// -----
// CAknExSettingListListBox::CreateSettingItemL
// -----
CAknSettingItem*
CAknExSettingListListBox::CreateSettingItemL(Tint
aldentifier)
{
    CAknSQttingltQm* SQttingltQm = NULL;

    switch (aldentifier)
    {
        case EAknExSettingText:
settingItem = new (ELeave)
CAknTextSettingItem(aldentifier, iData->iTextBuf);
        break;
        case EAknExSettingVolume:

```

```

        settingItem = new (ELeave)CAknVolumeSettingItem(
            aIdentifier, iData->iVolume );
    break; case EAknExSettingEnumText:
        settingItem = new (ELeave)
CAknEnumeratedTextPopupSettingItem(
            aIdentifier, iData->iLanguageCode);
    break; case EAknExSettingSlider:
        settingItem = new (ELeave)CAknSliderSettingItem(
            aIdentifier, iData->iSliderValue);
    break; case EAknExSettingDate:
        settingItem = new (ELeave)
CAknTimeOrDateSettingItem(aIdentifier,
            CAknTimeOrDateSettingItem::EDate,
            iData->iDate );
    break; case EAknExSettingTime:
        settingItem = new (ELeave)
CAknTimeOrDateSettingItem(aIdentifier,
            CAknTimeOrDateSettingItem::ETime,
            iData->iTime);
    break; case EAknExSettingBinary:
        settingItem = new (ELeave)
CAknBinaryPopupSettingItem(aIdentifier, iData-
            >iBinary); break;
case EAknExSettingPassAlph:
        settingItem = new (ELeave)
CAknPasswordSettingItem(aIdentifier,
            CAknPasswordSettingItem::rEAlpha,
            iData->iPw);

    break;
case EAknExSettingPassNumber:
        settingItem = new (ELeave)
CAknPasswordSettingItem(aIdentifier,
            CAknPasswordSettingItem::ENumeric,
            iData->iPin);
    break; case EAknExSettingIpAddress:
        settingItem = new (ELeave)CAknIpFieldSettingItem(
            aIdentifier, iData-

```

```
>iIpAddress);  
    break;  
    default:  
    break;  
    }  
    return settingItem;  
}
```

Все достаточно просто: создается указатель на объект `settingItem` класса `CAknSettingItem`, и при выборе одной из констант перечисляемого типа `TaknExSettingItem`, объявленного в файле `CAknExSettingList.hrh`, происходит поочередное создание всех типов списка `Setting`.

Последний класс, `CAknExSettingListItemData`, содержит необходимые данные для каждого из типов списка. В файле `PKG` находятся выходные данные для создания установочного пакета программы `Setting List`.

В следующей главе будет рассматриваться работа с графикой.



## Глава 9. Программирование графики

Создавая программы с использованием графики можно сформировать красивое и неповторимое приложение. Гораздо приятней пользоваться программой с красочно оформленным интерфейсом, чем с программой, созданной в рамках классического оформления встроенных приложений Symbian OS. Операционная система Symbian имеет множество возможностей в работе с графикой, и в этой главе мы рассмотрим основные базовые компоненты.

Механизм работы с графикой в Symbian OS построен на основе взаимодействия приложения с Window Server (Сервер окна). На рис. 9.1 схематично показана общая модель взаимодействия Window Server и программ.



Рис. 9.1. Взаимодействие программ и Window Server

Для каждого приложения с помощью класса RWsSession создается сессия, связывающая программу с Window Server, через который на системном уровне происходит вывод графики на экран телефона и ее обновление, в случае если пользователь совершит действия по нажатию клавиш. Класс RWsSession низкоуровневый. Его задача состоит в обеспечении доступа приложения к Window Server. В свою очередь Window Server снабжает программу полноценным механизмом работы с графикой и обработкой событий, поступающих от пользователя и связанных с изменением состояния графического контекста.

В Symbian OS имеются графические библиотеки, содержащие разнообразный набор функциональных возможностей. Рассмотрим эти библиотеки:

- GDI - базовая библиотека для всех графических ресурсов системы. Обеспечивает работу со шрифтом, цветом и разного рода геометрическими операциями;

- WS32 представляет работу с Window Server, осуществляя доступ к графическому контексту;
- BITGDI - библиотека для работы с изображениями или точечными рисунками (BMP);
- FBSCLI представляет Font Server и Bitmap Server для работы со шрифтами и изображениями;
- MEDIACLINTIMAGE - мультимедийная библиотека для различных геометрических операций с изображениями.

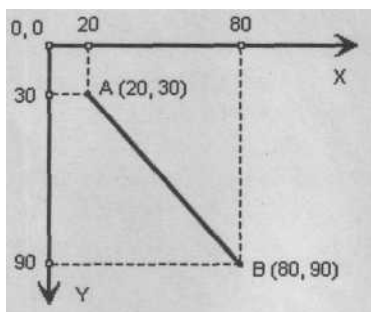


Рис. 9.2. Система координат в Symbian OS

В Symbian OS (в отличие от декартовой системы координат) применяется перевернутая система. Начальная точка (0, 0) системы координат находится в верхнем левом углу. Положительная ось X идет слева направо по верхней кромке экрана, а положительная ось Y - сверху вниз по левой стороне экрана (см. рис. 9.2).

Такая же система координат применяется в компьютерных системах и в мобильных телефонах при работе с двумерной графикой.

Теперь давайте перейдем к конкретным примерам, демонстрирующим практическую сторону программирования графики.

## 9.1. Рисование линий

Начнем с самого простейшего. Нарисуем на экране беспорядочно четыре линии, закрашенные разными цветами. Для этого нам надо будет выполнить определенный алгоритм действий. Прежде всего, устанавливаются две точки для начала и конца линии. Каждая точка должна содержать координаты по осям X и Y, как показано на рис. 9.2. Затем необходимо задать толщину линии. Далее выбирается цвет линии, стиль начертания, после чего можно рисовать ее на экране. Но прежде чем перейти к исходному коду программы LineTest, надо более тщательно ознакомиться с цветовой составляющей в Symbian OS.

Цвет в Symbian OS задан в виде констант с целочисленными значениями. Любой нарисованный графический элемент можно закрашивать каким-либо цветом, а для упрощения этого процесса в Symbian OS определен макрос `AKN_LAF_COLOR()`. Смысл работы этого макроса очень прост: в исходном коде создается константа, например, для зеленого цвета: `const Tint KGreen = 185;`

и с помощью макроса `AKN_LAF_COLOR()` происходит преобразование цветовой составляющей в RGB-значение. `TRgb colorGreen = AKN_LAF_COLOR(KGreen);`

Переменная `colorGreen` используется непосредственно в исходном коде для установки цвета графическому элементу. Для каждого из цветов в Symbian OS задается целочисленное значение, которое вы можете найти в табл. 9.1. Зеленый и серый цвета с номерами - это разные градации или оттенки этих цветов.

*Таблица 9.1. Цветовые значения*

<b>Цвет</b>	<b>Значение</b>
Белый	0
Желтый	5
Светло-желтый цвет	9
Светло-фиолетовый	13
Темно-желтый	17
Светло-красный	20
Оранжевый	23
Красный	35
Серый 3	43
Светло-коричневый	51
Светло-зеленый	76
Светло-сиреневый	84
Серый 6	86
Коричневый	95
Фиолетовый	105
Темно-красный	107
Светло-синий	120
Серый 9	129
Темно желтый	131
Темно-коричневый	137
Темно-фиолетовый	141
Светло-зеленый 2	146
Зеленый 2	159
Серый 12	172
Темно-сиреневый	176
Зеленый	185
Синий	210
Черный	215
Серый 1	216
Серый 2	217
Серый 4	218
Серый 5	219
Серый 7	220
Серый 8	221
Серый 10	222
Серый 11	223
Серый 13	224
Серый 14	225

Рассмотрим программу LineTest, в которой происходит прорисовка четырех линий. Команды меню для программы LineTest опущены, но в окончательном варианте программе, готовой для установки на телефон все команды для подэкранных клавиш должны быть обработаны соответствующим образом. На компакт-диске программа LineTest находится в папке \Code\LineTest\. Вся работа по рисованию графики происходит в функции Draw () файла исходного кода Test\_AppView.cpp, рассмотрим этот код.

```

// файл Test AppView.cpp
// реализация класса CTestAppView
// рисуем линии
//*****
// подключаем системные библиотеки
#include <coemain.h>
◆include <aknotewrappers.h>
◆include <eikenv.h>
// подключаем файл ресурса
◆include <Test.rsg>
// подключаем заголовочный файл
#include "Test_AppView.h"

// цветовые константы const
Tint KOrange = 23; const Tint
KDarkRed = 107; const Tint
KLightViolet = 84; const Tint
KBrown = 137;

// двухфазный конструктор
CTestAppView* CTestAppView::NewL(const TRect& aRect)
{
    CTestAppView* self = CTestAppView::NewLC(aRect);
    Cleanupstack::Pop(self) ;
    return self;
}
// двухфазный конструктор
CTestAppView* CTestAppView::NewLC(const TRect& aRect)
{
    CTestAppView* self = new (ELeave) CTestAppView;
    CleanupStack::PushL(self) ;
    self->ConstructL(aRect) ;
    return self;
}
// двухфазный конструктор
void CTestAppView::ConstructL(const TRectS aRect) {
    // создаем окно
    CreateWindowL();
    // определяем клиентскую область
    SetRect(aRect);
    // активизируем механизм прорисовки
    ActivateL();
}

```

```

}
// конструктор
CTestAppView::CTestAppView()
{
}
// деструктор
CTestAppView: ~CTestAppView ()
{
}
// рисуем на экране
void CTestAppView::Draw(const TRectS /*aRect*/) const
{
// получаем графический контекст окна
CWindowGcs gc = SystemGcO;
// очищаем клиентскую область окна
gc.Clear();

// оранжевый цвет
TRgb colorOrange = AKN_LAF_COLOR(KOrange) ;
// темно-красный цвет
TRgb colorDarkRed = AKN_LAF_COLOR(KDarkRed);
// сиреневый цвет
TRgb colorLightViolet = AKN_LAF_COLOR(KLightViolet);
// коричневый цвет
TRgb colorBrown = AKN_LAF_COLOR(KBrown);

// начальная точка для первой линии
TPoint pointBegin1(20, 10);
// конечная точка для первой линии
TPoint pointEnd1(20, 120);
// определяем толщину первой линии
TSize penSizePoint1(3, 3);
// задаем толщину первой линии
gc.SetPenSize(penSizePoint1);
// задаем цвет первой линии
gc.SetPenColor(colorOrange);
// стиль начертания первой линии
gc.SetPenStyle(CGraphicsContext::ESolidPen);
// рисуем первую линию
gc.DrawLine(pointBegin1, pointEnd1);

// вторая линия
TPoint pointBegin2(40, 10);
TPoint pointEnd2(40, 120);

```

```

TSize penSizePoint2(3, 3);
gc.SetPenSize(penSizePoint2);
gc.SetPenColor(colorDarkRed);
gc.SetPenStyle(CGraphicsContext::ESolidPen);
gc.DrawLine(pointBegin2, pointEnd2);

// третья линия
TPoint pointBegin3(60, 40);
TPoint pointEnd3(140, 40);
TSize penSizePoint3(3, 3);
gc.SetPenSize(penSizePoint3);
gc.SetPenColor(colorLightViolet);
gc.SetPenStyle(CGraphicsContext::ESolidPen);
gc.DrawLine(pointBegin3, pointEnd3);

// четвертая линия TPoint
pointBegin4(60, 40); TPoint
pointEnd4(140, 120); TSize
penSizePoint4(3, 3);
gc.SetPenSize(penSizePoint4);
gc.SetPenColor(colorBrown);
gc.SetPenStyle(CGraphicsContext::ESolidPen);
gc.DrawLine(pointBegin4, pointEnd4); }

```

Для работы с макросом ANN\_LAF\_COLOR () в начале исходного кода подключается заголовочный файл eikenv.h. Затем объявляются цветовые константы для оранжевого, темно-красного, светло-сиреневого и коричневого цветов с присвоением им целочисленных значений для каждого цвета.

```

// цветовые константы const
Tint KOrange = 23; const Tint
KDarkRed = 107; const Tint
KLightViolet = 84; const Tint
KBrown = 137;

```

Определив таким образом цветовые константы и подключив заголовочный системный файл eikenv.h, переходим в функцию Draw (), где происходит процесс отрисовки графики на экране. После получения графического контекста и очистки клиентской области экрана, производится конвертация цветовых значений с помощью макроса ANN\_LAF\_COLOR () для всех четырех цветов.

```

// оранжевый цвет

```

```
TRgb colorOrange = AKN_LAF_COLOR(KOrange);
// темно-красный цвет
TRgb colorDarkRed = AKN_LAF_COLOR(KDarkRed) ;
// сиреневый цвет
TRgb colorLightViolet = AKN_LAF_COLOR(KLightViolet);
// коричневый цвет
TRgb colorBrown = AKN_LAF_COLOR(KBrown);
```

В дальнейшем все четыре переменные для четырех разных цветов будут участвовать в закрашивании четырех линий.

Для того чтобы нарисовать линию, нам нужно определиться с начальной и конечной точкой, а отрезок между ними и будет нашей линией. Каждая точка состоит из пары значений по осям X и Y.

```
// начальная точка для первой линии TPoint
pointBegin1 (20, 10); // конечная точка для первой
линии TPoint pointEnd1(20, 120);
```

Объект pointBegin1 класса TPoint содержит координаты для начальной точки линии: по оси X это 20 пикселей, а по оси Y 10 пикселей.

В Symbian OS есть три специализированных класса: TPoint, TSize и TRect для облегчения программирования графики. При создании объектов pointBegin1 и pointEnd1 был задействован конструктор класса TPoint, содержащий позиции координат по осям X и Y. Класс TPoint используется специально для создания точки в пространстве. Если же вы хотите переназначить координаты точки, используя уже объявленный, созданный и инициализированный объект, воспользуйтесь функцией SetXY ().

После создания двух точек в пространстве надо определить толщину рисуемой линии.

```
// определяем толщину первой линии
TSize penSizePoint1(3, 3) ; // задаем
толщину первой линии
gc.SetPenSize (penSizePoint1) ;
```

В этом случае задействуется класс TSize, при создании объекта которого применяется конструктор с двумя целочисленными параметрами TSize (iwidth, iHeight), задающими ширину и высоту для определенной области. В нашем случае мы рисуем линию, поэтому заданный размер будет толщиной линии, а именно, в три пикселя.

И в конце задаем цвет, стиль начертания и рисуем линию с помощью функции DrawLine (), параметры которой - это начальная и конечная точки, между которыми рисуется линия.

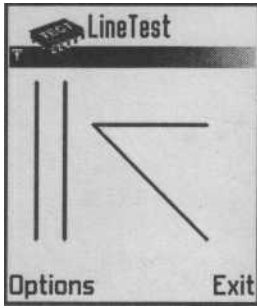


Рис. 9.3. Работа программы LineTest

Аналогично рисуются оставшиеся три линии, но уже со своими координатами и цветом. На рис. 9.3 изображена работа программы LineTest.

## 9.2. Рисуем прямоугольник

Нарисовать прямоугольник на экране можно несколькими способами. Класс TRect, рисующий фигуры прямоугольной формы имеет три различных конструктора. Создавая объект класса TRect, и применяя один из конструкторов можно добиться различной реализации одной и той же задачи. Иногда в исходном коде удобнее использовать один вид конструктора, иногда другой.

Рассмотрим имеющиеся конструкторы класса TRect.

Первый конструктор имеет четыре параметра для создания прямоугольника. TRect (Tint aAx, Tint aAy, Tint aBx, Tint aBy) Параметры конструктора класса TRect:

- aAx - целочисленное значение и точка по оси X;
- aAy - целочисленное значение первая точка по оси Y;
- aBx — целочисленное значение вторая точка по оси X;
- aBy — целочисленное значение вторая точка по оси Y.

Между двух этих координат в пространстве происходит прорисовка прямоугольной фигуры.

Во втором конструкторе определены два параметра, но в качестве параметров используется класс TPoint, с которым вы знакомы по предыдущему разделу. С помощью класса TPoint можно создать точку в пространстве, применяя двойку целочисленных значений по осям X и Y для определения координат точки. TRect (const TPointS aPointA, const TPointS aPointB)

Параметры конструктора TRect:

- aPointA - левый верхний угол прямоугольника;
- aPointB — правый нижний угол прямоугольника.

Относительно этих двух точек в пространстве формируется остальное построение фигуры. Третий конструктор класса TRect имеет еще более интересную конструкцию. Имеются два параметра, первый - это точка в пространстве (левый верхний угол), а второй - это высота и ширина прямоугольника. На основании этих данных и рисуется прямоугольник. TRect (const TPointS aPoint, const TSizeS aSize)

Параметры конструктора TRect:

- aPoint - левый верхний угол в пространстве;
- aSize - ширина по оси X и высота по оси Y для рисуемого прямоугольника.

Класс TSize - это еще один основной класс для упрощения работы с графикой в Symbian OS.

Конструктор TSize выглядит следующим образом:  
TSize (Tint aWidth, Tint aHeight)

>



Два целочисленных параметра определяют размер некой фигуры на основании ширины `aWidth` и высоты `aHeight`. Все значения задаются в пикселях, как впрочем, и для других параметров классов `TRect` и `TPoint`.

Теперь перейдем к демонстрационному примеру, который находится на компакт-диске в папке `\Code\RectTest`. Основные действия по представлению графики на экране в проекте `RectTest` происходят в файле `Test_AppView.cpp`, рассмотрим его код.

```
/y*****
// файл Test AppView.cpp
// реализация класса CTestAppView
// рисуем четыре прямоугольника

// подключаем системные библиотеки
◆include <coemain.h>
#include <aknnotewrappers.h>
#include <eikenv.h>
// подключаем файл ресурса
◆include <Test.rsg>
// подключаем заголовочный файл
◆include "Test_AppView.h"

// цветовые константы const
Tint KBlue = 210; const Tint
KDarkPurple = 141; const Tint
KGrey10 = 219; const Tint
KBlack = 215; const Tint
KDarkYellow = 17;

// двухфазный конструктор
CTestAppView* CTestAppView::NewL(const TRectS aRect)
{
    CTestAppView* self = CTestAppView::NewLC(aRect) ;
    CleanupStack::Pop(self);
    return self;
}
// двухфазный конструктор
CTestAppView* CTestAppView::NewLC(const TRects aRect) {
    CTestAppView* self = new (ELeave) CTestAppView;
    CleanupStack::PushL(self);
    self->ConstructL(aRect) ;
    return self;
}
```

```

// двухфазный конструктор
void CTestAppView::ConstructL(const TRectS aRect)
{
// создаем окно
CreateWindowL();
// определяем клиентскую область
SetRect(aRect);
// активизируем механизм прорисовки
ActivateL();
}
// конструктор
CTestAppView::CTestAppView()
{
}
// деструктор
CTestAppView::~CTestAppView()
{
}
// рисуем на экране
void CTestAppView::Draw(const TRect& /*aRect*/) const
{
// получаем графический контекст окна
CWindowGc& gc = SystemGcO;
// очищаем клиентскую область окна
gc.Clear();
//=====^=====
// рисуем первый прямоугольник, используя конструктор //
TRect(TInt aAx, TInt aAy, TInt aBx, TInt aBy)


---


// синий цвет
TRgb colorBlue = AKN LAF COLOR(KBlue); // цвет первого
прямоугольника gc.SetPenColor(colorBlue); // задаем
размер первого прямоугольника TRect drawRect1(10, 20,
50, 70); // рисуем первый прямоугольник
gc.DrawRect(drawRect1);
//=====^=====
// рисуем второй прямоугольник, используя конструктор
// TRect(const TPoints aPointA, const TPointS aPointB);


---


// темно-лиловый цвет
TRgb colorDarkPurple = AKN LAF COLOR(KDarkPurple);
// точка для верхнего левого угла прямоугольника

```

```

TPoint pointBegin1(30, 40);
// точка для правого нижнего угла прямоугольника
TPoint pointEnd1(60, 80);
// толщина бордюра
TSize penSizeRect2(6, 6);
// устанавливаем толщину
gc.SetPenSize(penSizeRect2);
// цвет второго прямоугольника
gc.SetPenColor(colorDarkPurple);
// задаем размер второму прямоугольнику
TRect drawRect2(pointBegin1, pointEnd1);
// рисуем второй прямоугольник
gc.DrawRect(drawRect2);
// сбрасываем установленные параметры
gc.Reset();

```

---

```

// рисуем третий прямоугольник, используя конструктор //
TRect(const TPointS aPoint, const TSizeS aSize)

```

---

```

// координаты верхнего левого угла
TPoint pointTopLeft3(90, 20);
// размер прямоугольника
TSize sizeRect3(*x*/50, /*y*/100);
// светло-серый цвет
TRgb colorGrey10 = AKN LAF COLOR(KGrey10);
// цвет кисти
gc.SetBrushColor(colorGrey10);
// закрашиваем кистью прямоугольник
gc.SetBrushStyle(CGraphicsContext::ESolidBrush);
// задаем размер третьему прямоугольнику
TRect drawRect3(pointTopLeft3, sizeRect3);
// рисуем третий прямоугольник
gc.DrawRect(drawRect3);

```

---

```

// рисуем четвертый прямоугольник, используя конструктор
// TRect(const TPointS aPoint, const TSizeS aSize)
// _____ , _____
// ----- ~
// координаты верхнего левого угла TPoint
pointTopLeft4(20, 100); // размер
прямоугольника TSize sizeRect4(50, 30);
// черный цвет для бордюра TRgb
colorBlack = AKN LAF COLOR(KBlack); //
определяем толщину бордюра ■

```

```

TSize penSizeRect4(8, 8);
// задаем толщину бордюра
gc.SetPenSize(penSizeRect4);
// устанавливаем цвет бордюру
gc.SetPenColor(colorBlack);
// темно-желтый цвет для кисти
TRgb colorKDarkYellow = AKN LAF COLOR(KDarkYellow);
// устанавливаем цвет кисти
gc.SetBrushColor(colorKDarkYellow);
// закрашиваем кистью прямоугольник
gc . SetBrushStyle (CGraphicsContext: .-ESolidBrush) ;
// задаем размер четвертому прямоугольнику
TRect drawRect4(pointTopLeft4, sizeRect4);
// рисуем четвертый прямоугольник
gc.DrawRect (drawRect4);
} //*****

```

В начале исходного кода создаются пять констант для работы с цветом: синим, темно-лиловым, одной из градаций серого, черным и темно-желтым. Далее в функции Draw () рисуются четыре разных прямоугольника.

Первый прямоугольник при создании объекта drawRect1 класса TRect использует конструктор из четырех целочисленных значений (по осям X и Y), на пересечении которых создаются точки, и рисуется прямоугольник. В качестве цвета линий задан синий цвет. Толщина линий не задана, а значит по умолчанию это один пиксель. Если так же не указать цвет для линий, то по умолчанию это черный цвет.

При прорисовке второго прямоугольника используется конструктор класса TRect с двумя параметрами на основе класса TPoint. Создается точка для левого верхнего угла с координатами (30,40) и для нижнего правого угла с координатами (60, 80) пикселей. Для прямоугольника устанавливается толщина линии в (6, 6) пикселей и закрашивается этот бордюр темно-лиловым цветом. В конце исходного кода для создания второго прямоугольника используется функция Reset (): gc.Reset ();

Эта функция сбрасывает все настройки, в данном случае для толщины и цвета бордюра. Если не вызывать эту функцию, то установленные значения будут действовать и для следующего исходного кода, описывающего создания той или иной фигуры, пока не будет вызвана функция Reset ().

Третий прямоугольник рисуется с помощью конструктора класса TRect. TRect (const TPoints aPoint, const TSize& aSize)

Создается точка для верхнего левого угла и размер прямоугольника по ширине и высоте, на основании этих данных и происходит построение третьего прямоугольника. Чтобы закрасить внутреннее пространство прямоугольника в серый цвет, применяется *кисть* (Brush).

```
gc.SetBrushStyle(CGraphicsContext::ESolidBrush);
```

Но при этом все равно автоматически прорисовывается бордюр прямоугольника шириной в один пиксель черного цвета. Если вы захотите закрасить прямоугольник, круг, эллипс или любой другой графический элемент, то перед закрасиванием фигуры кистью напишите следующую строку кода:

```
gc.SetPenStyle(CGraphicsContext::ENullPen);
```

Тем самым вы сбросите прорисовку контура фигуры.

Последний прямоугольник так же рисуется с помощью класса `TSize`, но с толстым бордюром в (8, 8) пикселей черного цвета. А внутреннее пространство прямоугольника закрашивается в темно-желтый цвет. На рис. 9.4 изображена работа программы `RectTest`.

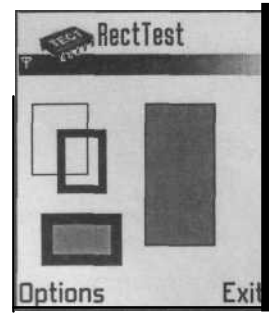


Рис. 9.4. Работа программы `RectTest`

### 9.3. Рисуем эллипс

Рисование эллипса происходит классом `TRect`. Первоначально происходит определение формы эллипса с помощью прямоугольника. Создавая прямоугольник с заданным размером и координатами, вы задаете площадь на экране для последующей отрисовки эллипса. Центр эллипса будет располагаться в центре прямоугольника, а его контур будет касаться сторон прямоугольника, учитывая особенность его формы. Сам прямоугольник не рисуется на экране, а лишь определяет площадь для формирования эллипса. Вместо эллипса можно нарисовать, например, окружность, если все стороны прямоугольника сделать равными.

Перейдем к примеру `EllipsTest`, где рисуются три разных по размеру и цвету эллипса. Пример находится на компакт-диске в папке `\Code\EllipsTest`, рассмотрим его содержимое:

```
// файл Test AppView.cpp
// реализация класса CTestAppView
// рисуем три эллипса на экране

// подключаем системные библиотеки
#include <coemain.h>
#include <aknnotewrappers.h>
#include <eikenv.h>
// подключаем файл ресурса
#include <Test.rsg>
// подключаем заголовочный файл
#include "Test AppView.h"

// цветовые константы
```

```

const Tint KOrange = 23;
const Tint KGreen2 = 159;

// двухфазный конструктор
CTestAppView* CTestAppView::NewL(const TRect& aRect)
{
    CTestAppView* self = CTestAppView::NewLC(aRect);
    CleanupStack::Pop(self);
    return self;
}
// двухфазный конструктор
CTestAppView* CTestAppView::NewLC(const TRectS aRect)
{
    CTestAppView* self = new (ELeave) CTestAppView;
    CleanupStack::PushL(self);
    self->ConstructL(aRect);
    return self;
}
// двухфазный конструктор
void CTestAppView::ConstructL(const TRectS aRect)
{
    // создаем окно
    CreateWindowL();
    // определяем клиентскую область
    SetRect(aRect);
    // активизируем механизм прорисовки
    ActivateL(); }
// конструктор
CTestAppView::CTestAppView()
{ }
// деструктор
CTestAppView::~CTestAppView() {
}
// рисуем на экране
void CTestAppView::Draw(const TRectS /*aRect*/) const
{
    // получаем графический контекст окна
    CWindowGcS gc = SystemGcO;
    // очищаем клиентскую область окна
    gc.Clear();
}

```

```

// _____
// рисуем первый эллипс

// левый верхний угол прямоугольника
TPoint pointBegin1 (10, 10);
// правый нижний угол прямоугольника
TPoint pointEnd1(40, 60);
// создаем прямоугольник
TRect drawRect1(pointBegin1, pointEnd1);
// рисуем в прямоугольнике эллипс
gc.DrawEllipse(drawRect1);

// _____
// _____
// рисуем второй эллипс

// левый верхний угол прямоугольника
TPoint pointBegin2 (50, 20);
// правый нижний угол прямоугольника
TPoint pointEnd2(80, 80);
// оранжевый цвет для эллипса
TRgb colorOrange = AKN LAF COLOR(KOrange) ;
// толщина бордюра у эллипса
TSize penSizeRect2 (4, 8);
// устанавливаем толщину
gc.SetPenSize(penSizeRect2) ;
// задаем цвет бордюру эллипса
gc.SetPenColor(colorOrange);
// создаем второй прямоугольник
TRect drawRect2(pointBegin2, pointEnd2);
// рисуем в прямоугольнике эллипс
gc.DrawEllipse(drawRect2);
// сбрасываем установленные параметры
gc.Reset();

// _____
// рисуем третий эллипс

// _____
// координаты верхнего левого угла
TPoint pointTopLeft3(90, 20);
// размер прямоугольника для эллипса
TSize sizeRect3(50, 100);
// светло зеленый цвет
TRgb colorGreen2 = AKN LAF COLOR(KGreen2);
// цвет кисти
gc.SetBrushColor (colorGreen2);
// закрашиваем кистью эллипс

```

```

gc.SetBrushStyle(CGraphicsContext::ESolidBrush);
// задаем размер третьему прямоугольнику TRect
drawRect3(pointTopLeft3, sizeRect3); // рисуем
эллипс gc.DrawEllipse(drawRect3);
} y/*****

```

В качестве цвета для эллипсов используется оранжевый и зеленый цвета. В функции Draw () происходит создание и отрисовка трех эллипсов.

Первый эллипс формируется при помощи конструктора класса TRect с двумя параметрами класса TPoint. После того, как создан прямоугольник с заданными размерами, происходит вызов функции DrawEllipse ():  
gc.DrawEllipse(drawRect1);

В качестве параметра, функция принимает созданный прямоугольник, который представлен объектом drawRect1 класса TRect. Все остальные действия по прорисовке эллипса происходят автоматически на основании размеров прямоугольника. Цвет и контурная линия заданы не были, а значит, толщина линии будет в один пиксель черного цвета.

Второй эллипс рисуется большего размера и имеет бордюр толщиной в (4, 8) пикселей. Два значения - 4 и 8 - это ширина рисуемого бордюра. Контур получается достаточно толстый, поэтому мы его закрасим в ярко-оранжевый цвет. В конце исходного кода по созданию второго эллипса вызывается функция Reset () для сброса установленных параметров для цвета и толщины линии.

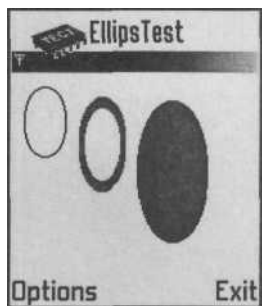


Рис. 9.5. Работа программы EllipsTest

Последний эллипс создан на основе конструктора класса TRect с параметрами, использующими классы TPoint и TSize, в которых задается левый верхний угол прямоугольника и его размер по ширине и высоте. Третий эллипс, вписанный в прямоугольник размером 50 на 100 пикселей, и закрашивается кистью в зеленый цвет:  
gc.SetBrushColor (colorGreen2);  
gc.SetBrushStyle(CGraphicsContext::ESolidBrush);

После этого он рисуется на экране при помощи функции gc.DrawEllipse(drawRect3);

На рис. 9.5 изображена работа программы EllipsTest.

## 9.4. Рисуем часть круга

В английском языке есть слово Pie (пирог) и именно это слово имеет в своем названии функция DrawPie (), отвечающая за отрисовку фигуры, похожей на пирог с вырезанным треугольным кусочком. Применительно к программированию слово «пирог» звучит весьма необычно, поэтому дадим этой фигуре название *часть круга*.



Чтобы нарисовать часть круга необходимо, так же как и в случае с эллипсом, создать прямоугольную площадь, внутри которой и рисуется часть круга. Функция DrawPie () содержит три параметра:

- KPieRect — прямоугольная площадь, внутри которой рисуется часть круга на основе класса TRect;
- pieStartPoint - двойка целочисленных значений, определяемых с помощью класса TPoint, задающих начальную точку для части круга;
- pieEndPoint - целочисленное значение на основе класса TPoint, задающее конечную точку для части круга.

С помощью первого параметра создается невидимый прямоугольник, внутри которого будет рисоваться часть круга. Стартовая точка pieStartPoint задает точку в пространстве площади прямоугольника по контуру рисуемого круга. Вторая точка pieEndPoint задает конец окружности и между первой и второй точками против часовой стрелки прорисовывается часть круга.

Перейдем к демонстрационной программе PieTest, иллюстрирующей принципы создания части круга. На компакт-диске программа находится в папке \Code\PieTest. В файле Test\_AppView.cpp располагается основной код приложения, рассмотрим его.

```
// файл Test_AppView.cpp
// реализация класса CTestAppView
// рисуем часть круга
y/*****
// подключаем системные библиотеки
#include <coemain.h>
#include <aknotewrappers.h>
#include <eikenv.h>
// подключаем файл ресурса
#include <Test.rsg>
// подключаем заголовочный файл
#include "Test_AppView.h"

// цветовые константы const
Tint KOrange = 23; const
Tint KGreen2 = 159;

// двухфазный конструктор
CTestAppView* CTestAppView::NewL(const TRectS aRect)
{
    CTestAppView* self = CTestAppView::NewLC(aRect);
    CleanupStack::Pop(self);
    return self;
}
```

```

// двухфазный конструктор
CTestAppView* CTestAppView::NewLC(const TRects aRect)
{
    CTestAppView* self = new (ELeave) CTestAppView;
    CleanupStack::PushL(self);
    self->ConstructL(aRect);
    return self;
}
// двухфазный конструктор
void CTestAppView::ConstructL(const TRectS aRect) {
// создаем окно
CreateWindowL();
// определяем клиентскую область
SetRect(aRect);
// активизируем механизм прорисовки
ActivateL(); }
// конструктор
CTestAppView::CTestAppView()
{ }
// деструктор
CTestAppView::~CTestAppView() {
}
// рисуем на экране void CTestAppView::Draw(const
TRect& /*aRect*/) const
{
    // получаем графический контекст окна
    CWindowGcS gc = SystemGcO; , // очищаем
    клиентскую область окна gc.Clear ();
//=====;=====;=====;=====;=====;=====;=====;=====;=====;=====
// рисуем часть круга

    // координаты верхнего левого угла
    TPoint pointTopLeft1 (10, 20);
    // размер прямоугольника
    TSize sizeRect1(80, 80);
    // светло-зеленный цвет
    TRgb colorGreen2 = AKN LAF COLOR(KGreen2);
    // толщина линии
    TSize penSizeRect1(4, 4);

```

```

    // устанавливаем толщину
    gc.SetPenSize (penSizeRect1);
    // устанавливаем цвет
    gc.SetPenColor (colorGreen2);
    // рисуем прямоугольник
    TRect drawRect1 (pointTopLeft1, sizeRect1);
    // начальная точка
    TPoint pointBegin1 (10, 20);
    // конечная точка
    TPoint pointEnd1 (80, 20);
    // рисуем кусок круга
    gc.DrawPie (drawRect1, pointBegin1, pointEnd1);
    // сброс настроек
    gc.Reset ();

// рисуем вторую часть круга
//
    // координаты верхнего левого угла
    TPoint pointTopLeft2 (100, 40);
    // размер прямоугольника
    TSize sizeRect2 (60, 60);
    // оранжевый цвет
    TRgb colorOrange = AKN_LAF_COLOR (KOrange);
    // устанавливаем цвет для кисти
    gc.SetBrushColor (colorOrange);
    // назначаем цвет кисти пирогу
    gc.SetBrushStyle (CGraphicsContext::ESolidBrush);
    // рисуем прямоугольник
    TRect drawRect2 (pointTopLeft2, sizeRect2);
    // начальная точка
    TPoint pointBegin2 (10, 30);
    // конечная точка
    TPoint pointEnd2 (50, 0);
    // рисуем часть круга
    gc.DrawPie (drawRect2, pointBegin2, pointEnd2);
}
/*****

```

Для первой части круга прямоугольная площадь задается размером 80 x 80 пикселей, создается бордюр (4, 4) и закрашивается в светло-зеленый цвет. Прямоугольник для части круга формируется на основе класса TRect с конструктором из двух параметров - TPoint и TSize, но выбор конструктора особого значения не имеет. Можно использовать любой удобный для вас конструктор при создании объекта класса TRect.

Затем с помощью строки кода:  
`gc.DrawPie(drawRectI, pointBeginI, pointEndI);`  
рисуется первая часть круга и происходит сброс настроек цвета и бордюра функцией `Reset ()`.

Вторая часть круга рисуется аналогичным образом, но с меньшим размером - 60 x 60 пикселей, пространство внутри части круга закрашивается в оранжевый цвет. На рис. 9.6 показан результат работы приложения `PieTest`.

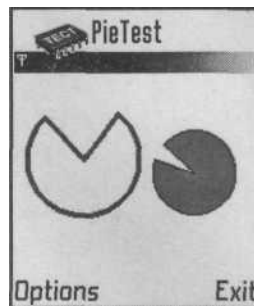


Рис. 9.6. Рисуем часть круга

## 9.5. Текст и шрифт

Очень часто требуется выводить любые виды текстовых сообщений на экран для информирования пользователя, например, о количестве набранных очков в игре, оставшихся патронах или других информационных сообщений.

Работа с текстом в Symbian OS построена на использовании системных классов и отличается особой простотой. Давайте перейдем к примеру `TextTest`, который находится на компакт-диске в папке `\Code\TextTest`.

Так же как и в случае с рисованием графики, основные действия происходят в файле `Test_AppView.cpp` в функции `Draw ()`. Рассмотрим исходный код файла `Test_AppView.cpp` проекта `TextTest`, иллюстрирующий простейший способ вывода текста на экран телефона.

```
!y*****
// файл Test AppView.cpp
// реализация класса CTestAppView
// текст
! y*****

// подключаем системные библиотеки
#include <coemain.h>
#include <aknnotewrappers.h>
#include <eikenv.h>
#include <stringloader.h>
// подключаем файл ресурса
#include <Test.rsg>
// подключаем заголовочный файл
#include "Test_AppView.h"

// двухфазный конструктор
CTestAppView* CTestAppView::NewL(const TRectS aRect)
{
    CTestAppView* self = CTestAppView::NewLC(aRect);
    CleanupStack::Pop(self);
}
```

```

        return self; }
// двухфазный конструктор
CTestAppView* CTestAppView::NewLC(const TRectS aRect) {
    CTestAppView* self = new (ELeave) CTestAppView;
    CleanupStack::PushL(self);
    self->ConstructL(aRect);
    return self;
}
// двухфазный конструктор
void CTestAppView::ConstructL(const TRect& aRect)
{
    // создаем окно
    CreateWindowL();
    // определяем клиентскую область
    SetRect(aRect);
    // активизируем механизм прорисовки
    ActivateL(); }
// конструктор
CTestAppView::CTestAppView()
{ }
// деструктор
CTestAppView::~CTestAppView() {
}
// рисуем на экране
void CTestAppView::Draw(const TRectS /*aRect*/) const
{
    // получаем графический контекст окна
    CWindowGc& gc = SystemGcO;
    // очищаем клиентскую область окна
    gc.Clear();
    // позиция для текста
    TPoint textPosition(20, 30);
    // строка текста
    TBuf<20> mytext;
    // загрузка текста
    StringLoader::Load(mytext, R TEXT FONT);
    // создаем шрифт
    const CFont* normalMyFont = iEikonEnv->NormalFont();
    // устанавливаем созданный шрифт

```

```

gc.UseFont(normalMyFont);
// рисуем текст
gc.DrawText(mytext, textPointPosition);
// очищаем память
gc.DiscardFont();
} /Z*****^*****

```

Изначально в исходном коде происходит подключение заголовочного файла `stringloader.h` для возможности работы с текстом.

Перейдем в функцию `Draw()`. С помощью объекта `textPointPosition` класса `TPoint` создается точка в пространстве с координатами  $x=20$  пикселей и  $y=30$  пикселей. Этой точкой определяется верхний левый угол невидимой прямоугольной области, внутри которой будет выводиться текст на экране. Затем создается объект `mytext` класса `TBuf`, с возможностью хранения двадцати символов в буфере.

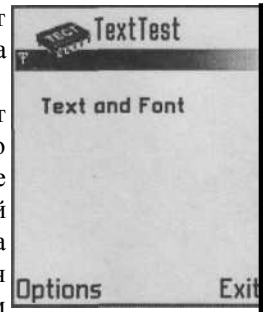
Далее происходит вызов функции `Load ()` с двумя параметрами. Первый параметр - это объект `mytext`, который будет содержать строку текста не более двадцати символов. Второй параметр как раз и определяет содержимое строки текста. В этом примере используется ресурс `R_TEXT_FONT` для гибкого подхода в локализации программы. В файле `Text.rss` в папке `\Code\TextTest\group` в ресурсах программы добавим расшифровку для `R_TEXT_FONT`. `buf = "Text and Font"`;

Теперь объект `mytext` содержит эту строку текста, и все готово для вывода ее на экран, но прежде надо задать шрифт для текста. Шрифт задается при помощи класса `CFont`: `const CFont* normalMyFont = iEikonEnv->NormalFont ();`

Этой строкой кода происходит назначение одного из видов системного шрифта для сформированного ранее текста. Функция `NormalFont ()` производит назначение простого шрифта под названием `Normal` (Нормальный). Результат этой операции сохраняется в объекте `normalMyFont`. В строке кода `gc.UseFont(normalMyFont);`

простой шрифт назначается для всех последующих строк текста, и уже в функции `DrawText ()` строка текста "Text and Font" выводится на экран. Всего существует три функции `CGraphicsContext: : DrawText ()` с разными наборами параметров, регулирующими начертание текста на экране. Функция `DrawText ()` в нашем примере имеет два параметра. Первый параметр - это строка текста, представленная объектом `mytext`, а второй параметр определяет позицию для вывода текста. А точнее, определяется позиция на экране для левого верхнего угла невидимой прямоугольной области, внутри которой рисуется текст. Размер прямоугольной области определяется автоматически функцией `DrawText ()` на основании вида задействованного шрифта, а это размер и длина строки. В нашем случае, это двадцать символов, включая пробелы.

Последняя функция `DiscardFont ()` подстраховывает неправильное создание шрифта и освобождает память. На рис. 9.7 показана работа программы `Text Test`.



Прямоугольная область, в которой выводится текст приложения, напрямую связана со стилем используемого шрифта и его размером. Каждый текст может иметь свое написание, заданное *метрикой*, с помощью которой определяется дистанция между буквами, наклон, высота букв и так далее. На основе этих данных формируется написание текста с заданным шрифтом. Помните, в первом классе были тетрадки в линейку для прописи, где приходилось писать буквы в строго заданных линейчках с заданным наклоном и расстоянием между буквами?

Вот это и есть метрика. Посмотрите на рис. 9.8, где схематично показана метрика, применяемая в `Symbian OS`.

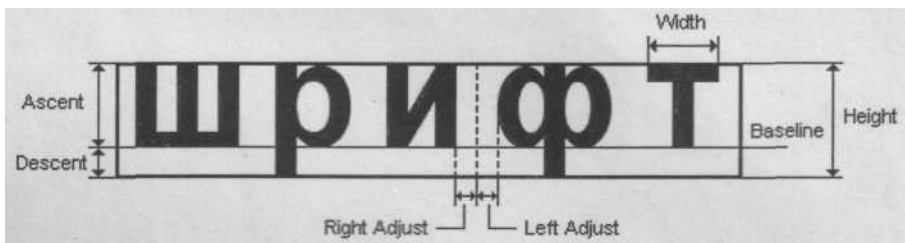


Рис. 9.8. Метрика

- Baseline - это горизонтальная линия, на которой расположен текст;
- Width - ширина буквы в тексте;
- Height - высота буквы в тексте;
- Ascent - дистанция от Baseline до верха буквы;
- Descent - расстояние от Baseline до низа буквы;
- Right Adjust — пространство после буквы;
- Left Adjust — пространство перед буквой;
- Move - это сумма значений в Width, Left Adjust и Right Adjust.

Все вышеперечисленные метричные данные можно использовать при создании текста, вызвав функцию `CGraphicsContext::Draw ()` с нужным набором параметров. Сейчас давайте перейдем к примеру `FontText`, где показывается применение различных системных шрифтов. Проект находится в папке `\Code\FontText`. Нас интересует файл `Test_AppView.cpp` в папке `\src` в каталоге проекта, рассмотрим этот исходный код.

```
// файл Test_AppView.cpp
// реализация класса CTestAppView
```

```

// различные виды шрифта

// подключаем системные библиотеки
#include <coemain.h>
#include <aknnotewrappers.h>
#include <eikenv.h>
#include <stringloader.h>
// подключаем файл ресурса
#include <Test.rsg>
// подключаем заголовочный файл
#include "Test_AppView.h"

// цветовые константы const
Tint KBlue = 210; const Tint
KLightRed = 20; const Tint
KOrange = 23; const Tint KBrown
= 95; const Tint KDarkPurple =
141;

// двухфазный конструктор
CTestAppView* CTestAppView::NewL(const TRectS aRect)
{
    CTestAppView* self =
        CTestAppView::NewLC(aRect);
    CleanupStack::Pop(self);
    return self;
}
// двухфазный конструктор
CTestAppView* CTestAppView::NewLC(const TRectS aRect)
{
    CTestAppView* self = new (ELeave) CTestAppView;
    CleanupStack::PushL(self);
    self->ConstructL(aRect);
    return self;
}
// двухфазный конструктор
void CTestAppView::ConstructL(const TRectS aRect)
{
    // создаем окно
    CreateWindowL();
    // определяем клиентскую область
    SetRect(aRect);
    // активизируем механизм прорисовки
    ActivateL(); }

```





```

const CFont* titleMyFont = iEikonEnv->TitleFont();
// устанавливаем созданный шрифт
gc.UseFont(titleMyFont);
// рисуем текст
gc.DrawText(mytext, textPointPosition2);
// очищаем память
gc.DiscardFont ();

// шрифт Annotation
// -----
// светло-красный цвет
TRgb colorLightRed = AKN LAF COLOR(KLightRed);
// устанавливаем цвет
gc.SetPenColor(colorLightRed);
// позиция для текста
TPoint textPointPosition3(15, 55);
// создаем шрифт
const CFont* annotationMyFont = iEikonEnv-
    >AnnotationFont(); //
устанавливаем созданный шрифт
gc.UseFont(annotationMyFont); //
рисуем текст
gc.DrawText(mytext, textPointPosition3);
// очищаем память gc.DiscardFont(); //
сбрасываем настройки gc.Reset();
// шрифт Legend
// -----
// позиция для текста
TPoint textPointPosition4(40, 70);
// создаем шрифт
const CFont* legendMyFont = iEikonEnv->LegendFont();
// устанавливаем созданный шрифт
gc.UseFont(legendMyFont);
// рисуем текст
gc.DrawText(mytext, textPointPosition4);
// очищаем память
gc.DiscardFont ();
I / -----
// шрифт Symbol
// -----
// оранжевый цвет

```

```

TRgb colorOrange = AKN LAF COLOR(KOrange);
// устанавливаем цвет
gc.SetPenColor(colorOrange);
// позиция для текста
TPoint textPointPosition5(5, 85);
// создаем шрифт
const CFont* symbolMyFont = iEikonEnv->SymbolFont();
// устанавливаем созданный шрифт
gc.UseFont(symbolMyFont);
// рисуем текст
gc.DrawText(mytext, textPointPosition5);
// очищаем память
gc.DiscardFont();
// сбрасываем настройки
gc.Reset();

```

---

```

// шрифт Dense

// позиция для текста
TPoint textPointPosition6(20, 100);
// создаем шрифт
const CFont* denseMyFont = iEikonEnv->DenseFont();
// устанавливаем созданный шрифт
gc.UseFont(denseMyFont);
// рисуем текст
gc.DrawText(mytext, textPointPosition6);
// очищаем память
gc.DiscardFont();
//=====
// рисуем текст вертикально
//=====:=====
// позиция для текста
TPoint textPointPosition7(140, 20);
// создаем шрифт
const CFont* normalMyFont1 = iEikonEnv->NormalFont();
// устанавливаем созданный шрифт
gc.UseFont(normalMyFont1);
// рисуем текст
gc.DrawTextVertical(mytext, textPointPosition7, EFalse);
// очищаем память
gc.DiscardFont();

```

---

```

// рисуем подчеркнутый текст

```

---

```

// коричневый цвет
TRgb colorBrown = AKN_LAF_COLOR(KBrown);
// устанавливаем цвет
gc.SetPenColor(colorBrown);
// позиция для текста
TPoint textPointPosition8(5, 120);
// создаем шрифт
const CFont* legendMyFont1 = iEikonEnv->LegendFont();
// устанавливаем созданный шрифт
gc.UseFont(legendMyFont1);
// устанавливаем тексту подчеркивание
gc.SetUnderlineStyle(EUnderlineOn);
// рисуем текст
gc.DrawText(mytext, textPointPosition8);
// очищаем память
gc.DiscardFont();
// сбрасываем подчеркивание
gc.SetUnderlineStyle(EUnderlineOff);

// рисуем зачеркнутый текст
// _____
//
// темно-лиловый цвет
TRgb colorDarkPurple = AKN_LAF_COLOR(KDarkPurple); //
устанавливаем цвет gc.SetPenColor(colorDarkPurple); // позиция для
текста TPoint textPointPosition9 (50, 135); // создаем шрифт
const CFont* symbolMyFont1 = iEikonEnv->SymbolFont(); // устанавливаем
созданный шрифт gc.UseFont(symbolMyFont1); // устанавливаем
зачеркивание текста gc.SetStrikethroughStyle(EStrikethroughOn); // рисуем
текст
gc.DrawText(mytext, textPointPosition9); // очищаем память
gc.DiscardFont(); }

```

В примере FontTest на экран последовательно выводится одна и та же строка текста, но при этом для каждой строки используется свой шрифт. Мы задействовали шесть шрифтов: Normal, Title, Annotation, Legend, Symbol и Dense. Из англоязычных названий нетрудно догадаться о предназначении каждого. В Symbian OS есть и другие шрифты, информацию о них можно найти в справочной системе.

Позиции для строк текста задавались целочисленными значениями на основании класса TPoint, цель которого состоит в создании точки в пространстве. Для вывода шрифта на экран можно еще воспользоваться функциями Height () и width () для получения значений высоты и ширины экрана. Например, для нахождения точки в центре экрана можно написать такой код: TPoint centerPoint(aRect.Width(), aRect.Height());

По умолчанию цвет текста черный, а для изменения цвета необходимо задействовать функцию SetPenColor () с цветовой константой, как это было сделано в предыдущих разделах. Исходный код, рисующий строку текста на экране, закомментирован. Надеюсь, вам не составит труда разобраться с ним самостоятельно. Лишь две конструкции кода с созданием подчеркнутого и зачеркнутого текста тре-

буют к себе внимания. В этом случае шрифт, текст, цвет и вывод строки текста происходит как обычно, но еще задействуется функция SetUnderlineStyle () для установки подчеркивания текста и функция SetStrikethroughStyle () для зачеркивания текста. В обоих случаях в качестве параметров используются константы с окончанием On для включения задействованных установок (EUnderlineOn и EStrikethroughOn). Чтобы отключить установленные свойства для текста, воспользуйтесь аналогичными константами, но с концовкой Off, как это сделано в подчеркнутом тесте. Вызов функции Reset () так же сбросит все установленные настройки. На рис. 9.9 показана работа программы FontTest.

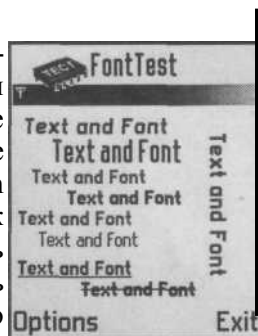


Рис. 9.9. Работа программы FontTest

## 9.6. Работа с изображениями

Графические изображения очень часто применяются в программах, в основном в играх или при создании различных логотипов. В Symbian OS формат для графических изображений MBM (Multi-bitmap), создаваемый на основе конвертации точечного рисунка BMP. Можно так же загружать изображения в форматах GIF и JPEG, но они все равно будут конвертированы в формат MBM и процесс конвертации этих форматов гораздо сложнее, чем BMP, поэтому на начальном этапе намного проще работать с точечными рисунками.

Первое, что нужно сделать в проекте, где вы собираетесь загружать BMP файл(ы) - это открыть проектный файл MMP и задекларировать команды для загрузки изображения. Формат декларации строго определен и происходит в следующем порядке:

- START BITMAP - команда, объявляющая о начале блока команд для изображения;
- HEADER - эта команда показывает, что сгенерированный файл MBM компилятор должен искать в каталоге \SDK\Epoc32;
- TARGETPATH - это путь к файлу MBM в системе;

- SOURCEPATH - это путь к файлу в вашем проекте. Предварительно в проект создается папка с любым удобным названием, например, bitmaps, image, и в ней размещается загружаемое в программу изображение;
- SOURCE - название файла BMP;
- END - команда, завершающая блок команд для точечного рисунка.

Теперь давайте перейдем к демонстрационному примеру ImageTest, где происходит загрузка BMP-файла размером 174 x 144 пикселя, задействуя тем самым всю клиентскую область приложения. Проект ImageTest на компакт-диске находится в папке \Code\ImageTest. Сразу перейдем к файлу Test.mmp в папке \group проекта и посмотрим, каким образом была объявлена загрузка файла Bmp.bmp из папки \image проекта ImageTest.

```
START BITMAP      Test.mbm
HEADER
TARGETPATH       \system\apps\test
SOURCEPATH       ..\image
SOURCE           cl2  Bmp.bmp
END
```

После этого можно импортировать проект ImageTest в любую среду программирования и разбираться с кодом программы.

Для работы с BMP в заголовочном файле Test\_AppView.cpp проекта ImageTest объявляется объект iBitmap класса CFbsBitmap. Класс CFbsBitmap - это основной класс для работы с точечными рисунками, но еще существует класс CWSBitmap, значительно расширяющий возможность работы с Window Server. Теперь давайте перейдем к исходному коду файла Test\_AppView.cpp, который находится в папке \Code\ImageTest\src\Test\_AppView.cpp, и где происходят основные действия по загрузке изображения в приложение.

```
y!*****
// файл Test AppView.cpp
// реализация класса CTestAppView
// загружаем точечный рисунок
//A*****

// подключаем системные библиотеки
#include <coemain.h>
◆include <aknotewrappers.h>
#include <aknutils.h>
#include <fbs.h>
// подключаем файл ресурса
#include <Test.rsg>          v
#include <Test.mbg>
// подключаем заголовочный файл
#include "Test_AppView.h"
```

```

// двухфазный конструктор
CTestAppView* CTestAppView::NewL(const TRectS aRect)
{
    CTestAppView* self = CTestAppView::NewLC(aRect);
    CleanupStack::Pop(self);
    return self;
}
// двухфазный конструктор
CTestAppView* CTestAppView::NewLC(const TRectS aRect)
{
    CTestAppView* self = new (ELeave) CTestAppView;
    CleanupStack::PushL(self);
    self->ConstructL(aRect);
    return self;
}
// двухфазный конструктор
void CTestAppView::ConstructL(const TRectS aRect)
{
    // путь к файлу Test.mbm
    _LIT(KPath, "WsystemWappsWtestWTest.mbm");
    // bitmap представляет изображение
    TFileName bitmap (KPath);
    // загружаем изображение
    iBitmap = CEikonEnv::Static()->CreateBitmapL(bitmap, 0);
    // создаем окно
    CreateWindowL ();
    // определяем клиентскую область
    SetRect(aRect);
    // активизируем механизм прорисовки
    Activate!. ();
}
// конструктор
CTestAppView::CTestAppView()
{ }
// деструктор
CTestAppView::~CTestAppView() {
    delete iBitmap;
}
// рисуем на экране
void CTestAppView::Draw(const TRectS /*aRect*/) const
{
    // получаем графический контекст окна

```

```

CWindowGcS gc = SystemGcO;
// очищаем клиентскую область окна
gc.Clear();
// выводим изображение на экране
gc.DrawBitmap(TPoint(0, 0), iBitmap);
}

```

Для загрузки точечного рисунка нужно подключить две системные библиотеки `aknutils.h` и `fbs.h`. В функции `ConstructL()`, играющей роль конструктора, сначала происходит инициализация переменной `KPath` и впоследствии объекта `bitmap` класса `TFileName`, с названием и местонахождением сгенерированного файла MBM. По идее, вы должны загружать в проект BMP-файл, который находится в папке проекта, но изначально в проекте происходит компиляция всех файлов ресурсов и изображений. В связи с этим наше изображение `Vmp.bmp` проекта `ImageTest` конвертируется в `Testmbm`, так, как было указано в файле `MMP`, и сохраняется в системной папке SDK в `\system\apps\test\Test.bmp`. С этого места и происходит его загрузка в программу уже с помощью функции `CreateBitmapL()`. Функция имеет два параметра. Первый - это объект `bitmap` класса `TFileName`, представляющий данные о загружаемом файле изображения. Второй параметр - это идентификатор изображения. У нас картинка одна, поэтому можно просто поставить значение `0`. В функции `Draw()` происходит отображение загруженного файла с помощью функции `DrawBitmap()`. Первый параметр этой функции - точка в пространстве клиентской области экрана, определяющая положение левого верхнего угла изображения. У нас это значение равно `(0, 0)` — это начало координат или левый верхний угол клиентской области экрана. Второй параметр —



Рис. 9.10. Работа программы ImageTest

объект `iBitmap` класса `CFbsBitmap`, представляющий конвертированный точечный рисунок формата MBM. И последнее, что нужно сделать, - это обратиться к файлу `Test.pkg` проекта `ImageTest`, где надо указать путь к изображению, для того чтобы во время упаковки программы изображение так же было заархивировано в SIS-файл. Делается это следующим образом:

```

"..\\.\\.epoc32\data\z\system\apps\Test\Test.mbm"-
"!:\system\apps\Test\Test.mbm"

```

На рис. 9.10 показана загрузка изображения в программу `ImageTest`.

В следующей главе мы поговорим о Java 2 ME-программировании в Symbian OS.



# Глава 10. Программирование Java приложений

Операционная система Symbian в полной мере поддерживает работу Java-приложений. Среда исполнения Java-программ встроена в Symbian OS независимым модулем, тем самым, обеспечивая поддержку платформы Java 2 Micro Edition в смартфонах и коммуникаторах разных производителей.

В этой книге мы только коснемся общих концепций построения Java 2 ME-программ на платформе Symbian OS, а так же рассмотрим некоторые моменты, связанные с графическим интерфейсом пользователя. Создание Java-программ для Symbian OS ничем не отличается от создания аналогичных программ для мобильных телефонов с поддержкой Java. Издательством ДМК недавно была выпущена книга под названием «Программирование мобильных телефонов на Java 2 Micro Edition», в которой доступно объясняется полный этап создания Java 2 ME-программ под мобильные телефоны различных марок.

## 10.1. Платформа Java 2 ME

Технология Java, созданная компанией Sun Microsystems Inc., состоит из трех основных элементов:

- система времени исполнения;
- язык Java;
- библиотека Java API.

Совокупность этих элементов и образует технологию Java, которая, в свою очередь, делится на три независимых друг от друга платформы:

- Java 2 Enterprise Edition;
- Java 2 Standard Edition;
- Java 2 Micro Edition.

Платформы Java 2 EE и Java 2 SE предназначены для работы на серверных и персональных компьютерных системах, а платформа Java 2 ME подготовлена специально для работы на мобильных устройствах с ограниченными системными ресурсами. Мобильные телефоны, в отличие от мощных компьютерных систем, обладают куда меньшими системными характеристиками, а Java 2 ME прекрасно адаптирована для работы в мобильных устройствах. Надежность, компактность и архитектурная независимость платформы Java 2 ME послужили ее широкому распространению.

Платформа Java 2 ME, как уже было упомянуто, состоит из системы времени выполнения, или *виртуальной Java машины*, языка программирования Java и на-

бора библиотечных функций, необходимых при создании Java-программ. Работа Java-приложений построена на основе интерпретации, то есть при компиляции исходного кода программы создается виртуальный код, который при работе программы налету читается виртуальной Java-машиной, образуя тем самым среду времени исполнения Java-программ. Очевидно, что среда времени исполнения или виртуальная Java-машина должна быть встроена в телефон. Таким образом осуществляется поддержка платформы Java 2 ME производителями телефонов. Мобильные устройства, работающие на основании прошивки, на сегодняшний день в своем большинстве имеют поддержку Java 2 ME (об этом заботятся производители). Тогда как смартфоны и коммуникаторы, работающие под управлением операционной системы Symbian, имеют встроенную поддержку платформы Java 2 ME, осуществляемую самой Symbian OS, о чем позаботились уже системные программисты компании Symbian Ltd.

Для разработчиков платформа Java 2 ME построена в виде блочной архитектуры настраиваемых друг над другом модулей. Два основных модуля в Java 2 ME - это конфигурация CLDC (Connected Limited Device Configuration - конфигурация подключаемых устройств с ограничениями) и профиль MIDP (Mobile Information Device Profile - информационный профиль мобильных устройств). Профиль MIDP настраивается над конфигурацией CLDC и именно в такой связке осуществляется разработка мобильных приложений для Symbian OS.

Профиль MIDP и конфигурация CLDC - это жестко заданные спецификации, изменение ядра которых программисту недоступно. Конфигурация CLDC и профиль MIDP имеют по две версии:

- MIDP 1.0, MIDP 2.0;
- CLDC 1.0, CLDC 1.1.

### **10.1.1. Конфигурация CLDC**

Две существующие версии *CLDC* отличаются не многим. Первая версия, CLDC 1.0, была доступна с самого начала создания платформы Java 2 ME, и на основе этой конфигурации выполнено 95% телефонов. Разница в двух конфигурациях минимальна, а очевидный единственный плюс новой версии, CLDC 1.1, - это поддержка чисел с плавающей точкой (дробные числа). Но пока конфигурация CLDC 1.1 насчитывает пару-тройку очень дорогих телефонов, поэтому пока рано ориентироваться на эту конфигурацию. Операционная система Symbian версии 6.1 поддерживает только конфигурацию CLDC 1.0, тогда как Symbian OS 7.0 поддерживает возможность работы с CLDC 1.1.

Теперь о том, из чего состоит CLDC. Конфигурация CLDC создана крупной экспертной группой, в которую входят большинство производителей мобильных устройств. CLDC предъявляет ряд технических требований к телефонам, а именно:

- 16- или 32-разрядный процессор;
- минимум 160 Кб выделенной памяти под платформу Java 2ME;

- питание от аккумулятора;
- беспроводное сетевое соединение.

Это минимальные системные требования для телефонов, поддерживающих технологию Java.

В состав конфигурации CLDC входит виртуальная Java-машина, носящая название KVM (Kilobait Virtual Machine) из-за своего маленького размера и ориентированности на мобильные устройства, а так же ряд пакетов из Java 2 SE, в усеченном виде:

- javalang;
- javautil;
- javaio.

Так же предъявляются некоторые требования к свойствам языка, которые должны соответствовать самой спецификации языка Java.

### ***10.1.2. Профили MIDP***

Над конфигурацией надстраиваются *профили*, которые имеют две версии: MIDP 1.0 и MIDP 2.0. Два этих профиля построены на одной базе, но в профиле MIDP 2.0 содержится ряд дополнительных пакетов с набором новых интерфейсов, классов и констант, значительно улучшающих разработку Java-программ. Платформа Symbian OS версии 6.1 работает только с профилем MIDP 1.0, а поздняя версия Symbian OS 7.0 уже с версией MIDP 2.0.

Профиль MIDP так же разработан экспертной группой, в составе которой множество производителей мобильных устройств. В конфигурации MIDP предъявляет свои минимальные технические требования к аппаратной части устройства:

- разрешение экрана от 96 x 54 пикселей;
- глубина экрана от 1 бита;
- 32 Кб динамической памяти;
- 128 Кб для Java компонентов;
- 8 Кб для хранения постоянных данных.

Профиль MIDP 1.0 включает в себя следующие пакеты:

- javax.microedition.lcdi;
- javax.microedition.io;
- javax.microedition.pki;
- javax.microedition.midlet;
- javax.microedition.rms.

В состав профиля MIDP 2.0 входят все вышеперечисленные пакеты из профиля MIDP 1.0 и еще три дополнительных пакета:

- javax.microedition.lcdi.game;
- javax.microedition.media;
- javax.microedition.mediacontrol.

То есть профиль MIDP 2.0 является расширенной версией профиля MIDP 1.0.

Вам как программисту наличие разных версий конфигураций и профилей добавляет лишнюю головную боль. Ряд телефонных аппаратов работают только с CLDC 1.0/MIDP 1.0, ряд с CLDC 1.0/MIDP 2.0, другие телефоны поддерживают CLDC 1.1/MIDP 2.0, что соответственно только усложняет разработку программ. При создании Java 2 ME-приложений надо следить за тем, какую из связок профиля и конфигурации планируется использовать. Иначе программа, написанная, например, под CLDC 1.1/MIDP 2.0, не сможет работать на устройстве, поддерживающем CLDC 1.0/MIDP 1.0. Поэтому при написании Java-программ под Symbian OS надо быть особенно внимательным и тщательно планировать разработку приложений. Выбор той или иной связки профиля и конфигурации происходит на этапе создания проекта в одной из интегрированных сред программирования приложений. И уже на основе выбранной связки конфигурации и профиля CLDC/MIDP будут доступны соответствующие API. Можно разграничить версии профилей и конфигураций следующим образом:

- Symbian OS 6.1 - CLDC 1.0/MIDP 1.0;
- Symbian OS 7.0 - CLDC 1.0/MIDP 2.0.

В *приложении 2* перечислены все имеющиеся мобильные телефоны, работающие под управлением Symbian OS с подробным описанием технических характеристик, и вы можете проследить поддержку того или иного профиля и конфигурации.

Такая запутанная ситуация с CLDC/MIDP не добавляет оптимизма программисту, но технология Java 2 ME еще только развивается и это необходимо иметь в виду. Поэтому при изучении программирования телефонов на базе Symbian OS лучше ориентироваться на версии CLDC 1.0/MIDP 2.0 - это ближайшее будущее, которое наступит уже завтра.

Полный перечень функциональных возможностей пакетов Java 2 ME можно найти на сайте компании Sun Microsystems по адресу: <http://javasun.com> или в справочнике книги «Программирование мобильных телефонов на Java 2 Micro Edition» издательства ДМК.

## 10.2. Мидлет

Приложение, созданное под платформу Java 2 ME, позиционируется как мидлет. *Мидлет* - это программа, написанная для мобильного телефона с использованием платформы Java 2 Micro Edition. Структура программ на Java 2 ME делится на основной класс мидлета, наследуемый от класса MIDlet пакета `javax.microedition.midlet.*`, и вспомогательные классы, реализующие, как правило, системную логику программы. *Основной класс мидлета* - это сердце и мотор всего приложения, с него начинается работа всей программы. Конечно, при желании можно интегрировать создаваемые дополнительные классы и в исходный код основного класса мидлета, но язык программирования Java тем и хорош, что он объектно-ориентированный и все его лучшие качества именно в этом. Для каких-то демонстрационных примеров допустимо встраивание классов в основной класс мидлета, но в реальном приложении надо воздерживаться от такой модели построения программ.

Основной класс мидлета, наследуемый от класса MIDlet, имеет три предопределенных метода, которые должны быть реализованы программистом:

- void startAppO - с помощью этого метода начинается работа мидлета, выражаясь абстрактно — это как зажигание в машине или точка отсчета программы;
- void pauseAppO - иногда требуется приостановить приложение на заданный промежуток времени, например, выход из программы на некоторое время, без закрытия приложения и возврат, а реализация этого метода дает такую возможность;
- void destroyApp() - если метод startAppO «зажигание», то этот метод «тормозная педаль» программы, то есть, реализовав этот метод, вы предопределяете логическое завершение работы всего приложения.

После загрузки Java-приложения в телефон и его активизации происходит инициализация всех глобальных переменных программы, исполнение кода конструктора основного класса мидлета и поиск метода startAppO. Метод startApp () - отправная точка приложения, и с его работы начинается активизация Java-программы. Обычно метод startApp () содержит экземпляры сопутствующих классов приложения, осуществляя тем самым связь между классами одной программы.

Метод destroyApp () завершает работу приложения. Вызов метода destroyApp () почти всегда происходит после того, как пользователь решит выйти из программы, воспользовавшись командой **Выход** из меню приложения. Команда **Выход** в Java-программах так же может располагаться и на подэкранной клавише, нажав на которую пользователь покидает программу. Работа метода destroyApp () сводится к обнулению всех ссылок в программе и выгрузке мидлета из памяти телефона.

### **10.2.1. Структура работы мидлета**

Системные ресурсы телефонов ограничены, поэтому работа в Java-программах строится по принципу поэкранного отображения информации. В один промежуток времени прорисовывается только один экран, за который отвечает класс Display из пакета javax.microedition.lcdui.\*. Чтобы произвести смену информации на дисплее, необходимо стереть один экран и нарисовать или отобразить на дисплее другой. Давайте рассмотрим небольшой исходный код примера. В этом простом примере создается простейший класс MyMidlet, отображающий текст на экране и имеющий команду выхода из программы.

```
/**
 * проект Java
 * Класс MyMidlet
 */

// подключаем библиотеки импорта
```

```

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

// класс MyMidlet
public class MyMidlet extends MIDlet implements
CommandListener

    // команда выхода из приложения
    private Command exitCommand;
    // объект d представляет экран телефона
    private Display d;
    // конструктор класса MyMidlet
    public MyMidlet()

        // получаем ссылку на дисплей
        d = Display.getDisplay(this) ;
        // команда выхода
        exitCommand = new Command("Выход", Command.EXIT, 1);

// старт всей программы
public void startApp()

    // создаем текстовый контейнер
    TextBox t = new TextBox("MnnjieT", "Symbian OS ", 256,
0);
    // добавляем команду выхода
    t.addCommand(exitCommand);
    // устанавливаем обработчик событий
    t.setCommandListener(this) ;
    // отражаем текущий дисплей телефона
    d.setCurrent(t) ;

public void pauseApp() { }
public void destroyApp(boolean unconditional) { }

public void commandAction(Command c, Displayable s)
{
    if (c == exitCommand)
    {
        destroyApp(false);
        notifyDestroyed();
    }
}

```

В первых двух строках исходного кода программы Java происходит подключение или импорт двух библиотек Java API в проект. В этих библиотеках содержатся интерфейсы, классы, функции, константы для работы с пользовательским интерфейсом, с командами выхода, перехода в программе и так далее. За библиотеками импорта следует исходный код, объявляющий создание класса `MyMidlet`, наследуемого от класса `MIDlet` и задействуется интерфейс `CommandListener`. `import javax.microedition.midlet.*;`

Интерфейс `CommandListener` нужен для обработки событий, связанных с нажатием клавиш телефона, отвечающих за команды выхода или перехода в программе - это простой обработчик событий, реализованный в исходном коде примера Java с помощью метода `commandAction ()`.

Создаваемый класс `MyMidlet` наследует все возможности суперкласса `MIDlet` из пакета `javax.microedition.midlet.*`, ему становятся доступными методы `startApp ()`, `pauseApp ()` и `destroyApp ()`, описывая которые вы сможете создать рабочее приложение, `private Command exitCommand; private Display d;`

Следующие две строки исходного кода примера Java содержат два глобальных объявления, создающих объект `exitCommand` класса `Command` и объект `d` класса `Display`. В Java-программировании создание объекта происходит только при выделении ему динамической памяти, то есть при использовании ключевого слова `new`, а до этого момента объект `exitCommand` является всего лишь переменной. А вот при создании объекта `d` класса `Display` ситуация совсем иная. Давайте перейдем в конструктор класса `MyMidlet` и познакомимся с этой особенностью подробнее.

```
public MyMidlet() {  
    d = Display.getDisplay(this) ;  
    exitCommand = new Command("Выход", Command.EXIT, 1);
```

При создании объекта класса `Display` использовать оператор `new` не нужно - в Java 2 ME это делается автоматически. Вы просто создаете объект класса `Display`, в нашем случае это `d`, и объект `d` представляет физический дисплей телефона. При вызове метода `getDisplay ()` происходит получение ссылки на класс `Display` и ее сохранение в объекте `d`. Можно сказать, что выделение динамической памяти для объекта класса `Display` с помощью оператора `new` заменено конструкцией кода с применением метода `getDisplay ()`.

Впоследствии в исходном коде для отображения текущего экрана вызывается метод `setCurrent ()`, например: `d.setCurrent ()`;

В конструкторе класса `MyMidlet` также создается объект `exitCommand`, где

используется конструктор класса `Command` с тремя параметрами. Первый параметр - это текстовая надпись, отображающаяся на экране телефона над подэк-ранной клавишей, при нажатии которой будут происходить предусмотренные для этой клавиши команды действия. Вторым параметром конструктора `Command` как раз и определяется смысл команды назначенной для подэкранной клавиши. В примере Java была использована константа `EXIT` или выход. И последний параметр в конструкторе `Command` - это целочисленное значение, определяющее приоритет выполнения данной команды. Поскольку команд в Java-программе может быть много, то необходимо определять приоритет их выполнения. Число меньшее по значению означает высший приоритет для команды.

За конструктором класса `MyMidlet` в исходном коде примера Java следует вызов метода `startApp ()`, являющегося входной точкой всего приложения.

```
public void startApp() {
    TextBox t = new TextBox("Мидлет", "Symbian OS", 256, 0);
    t.addCommand(exitCommand);
    t.setCommandListener(this);
    d.setCurrent (t);
}
```

В первой строке тела метода `startApp ()` происходит создание объекта `t` класса `TextBox`. Класс `TextBox` - это класс пользовательского интерфейса из состава Java 2ME API (о классах пользовательского интерфейса мы поговорим позже в этой главе). При создании объекта `t` использовался конструктор класса `TextBox` с четырьмя параметрами. Первый параметр конструктора класса `TextBox` - это информационная метка или лейбл, появляющийся в верхней части дисплея телефона и служащий логическим названием для текущего экрана телефона. Вторым параметром - это строка текста, доступная для редактирования после запуска работы приложения на телефоне. Третьим параметром задается целочисленным значением, указывающим на максимальное количество доступных символов для отображения на экране телефона. И последний параметр конструктора класса `TextBox` устанавливает некоторые специфические ограничения, обычно величина этого параметра задается нулем. Создав объект класса `TextBox`, вы тем самым создали текстовый контейнер, в котором можно набирать текст с помощью клавиш телефона или редактировать имеющийся. Далее в методе `startApp ()` происходит добавление команды Выход из программы с помощью двух строк кода:

```
t.addCommand(exitCommand) ;:
t.setCommandListener(this);
```

В первой строке происходит присоединение команды Выход к объекту класса `t`, а во второй строке кода происходит вызов метода `setCommandListener ()` с добавлением обработчика событий к объекту `t` класса `TextBox`. И уже в строке программного кода



```
d.setCurrent (t);
```

происходит отображение текущего дисплея телефона. Вызов метода `setCurrent ()` производит прорисовку или обновление текущей информации на дисплее телефона. Системные ресурсы телефонов малы (в основном это относится к частоте работы процессора, нежели к системной памяти телефонов на платформе Symbian OS) и принцип работы Java-приложений сводится к поэкранному виду отображения информации. При каждом изменении состояния дисплея телефона должен следовать вызов метода `setCurrent ()` для обновления содержимого экрана.

И в конце программного кода из примера Java в строках:

```
public void commandAction(Command c, Displayable s)
{
    if (c == exitCommand)
    {
        destroyApp(false);
        notifyDestroyed();
    }
}
```

устанавливается обработчик событий для команд программы, с помощью которых пользователь, уже физически выбрав определенную клавишу телефона, может воспользоваться командами приложения. В примере ListingIO\_1 в методе `CommandAction ()` применяется конструкция операторов `if/else` для определения событий для команды `exitCommand`, а это завершение программы и выгрузка мидлета из памяти телефона.

Таким образом, структура работы мидлета и его жизненный цикл строится в основном классе мидлета, осуществляющем контроль над работой программы.

## **10.2.2. Экранная навигация**

Навигация в Java-программах очень важная система и может быть реализована как с помощью классов пользовательского интерфейса содержащихся в пакете `javax.microedition.lcdui.*`, так и командами, располагающимися над подэкранными клавишами телефона. Обычно на клавиатуре телефонов имеются две клавиши, расположенные непосредственно под дисплеем. Они и служат для обработки команд, находящихся в нижней части экрана телефона над этими клавишами. При создании Java-программы вы можете предусмотреть значительно большее количество команд, и чтобы все эти перечисленные команды были доступны, применяются классы пользовательского интерфейса или подэкранные клавиши. Когда в приложении вы создаете две команды, то при отображении этих команд на дисплее, они размещаются над подэкранными клавишами слева и справа. Если же команд программы больше чем две, то сервис телефона автоматически создает

меню на одной из клавиш телефона, которое становится доступным при нажатии этой клавиши. Создавая меню телефона таким способом, необходимо тщательно продумать модель используемой в приложениях навигации.

Классы пользовательского интерфейса предоставляют программисту значительно больше возможностей - это и списки, флаги, текстовые поля, кнопки, переключатели и многое другое. Классы пользовательского интерфейса платформы Java 2 ME теоретически разделены на две группы: высокоуровневые и низкоуровневые классы пользовательского интерфейса. Посмотрите на рис. 10.1, где показана иерархия классов пользовательского интерфейса.

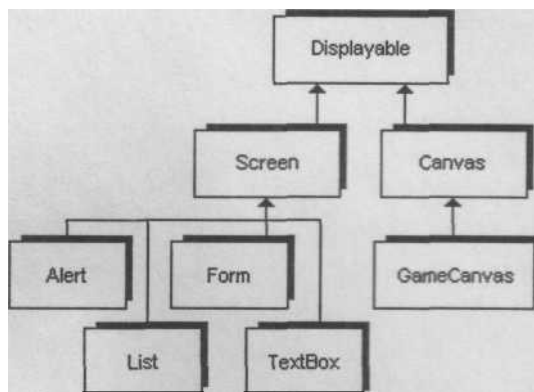


Рис. 10.1. Иерархия классов пользовательского интерфейса

Часть классов, располагающихся в иерархии абстрактного класса Screen, принадлежит к высокоуровневым классам, а часть классов в иерархии Canvas к низкоуровневым.

*Высокоуровневые классы пользовательского интерфейса* - это шаблонные классы с набором жестко заданного функционала, необходимого для создания меню, списков, групп выбираемых элементов, флагов переключателей, бегущих строк, кнопок, текстовых полей и много другого. Создавая объекты с применением методов этих классов, вы сможете пользоваться всеми имеющимися возможностями этих классов.

*Низкоуровневые классы пользовательского интерфейса* - это классы, дающие возможность программисту работать с графикой и обрабатывать низкоуровневые команды, поступающие с клавиш телефона.

В своих приложениях вы можете использовать любое количество классов из обоих интерфейсов, комбинируя и komponуя классы в одну программу. На рис. 10.1 изображена иерархия классов для CLDC 1.0/MIDP2.0, на основании этих версий конфигурации и профиля будет происходить построение дальнейшей части главы.

## 10.3. Высокоуровневый пользовательский интерфейс

Классы *высокоуровневого пользовательского интерфейса* из пакета `javax.mic-roedition.lcdui.*` обладают жестко заданным интерфейсом, и если вы создаете приложения под Symbian OS для серии 60 или UIQ, то нет особой необходимости отслеживать совместимость с разрешением экрана. Классы высокоуровневого интерфейса дают программисту некий шаблонный типаж функциональных возможностей, которые автоматически встраиваются в интерфейс серии 60 и UIQ. Единственное, над чем необходимо осуществлять контроль - это над версией профиля. Для Symbian OS 6.1 доступен только профиль MIDP 1.0 и при создании программ для Symbian OS 6.1 нельзя допускать использование компонентов из MIDP 2.0. Некоторые классы пользовательского интерфейса доступны только в профиле MIDP 2.0, но более того, в некоторых классах для MIDP 1.0 могут содержаться методы, которые работают только с профилем MIDP 2.0! Поэтому будьте внимательны при планировании структурной модели работы приложения. Тем не менее, общая картина с MIDP/CLDC для платформы вполне очевидна: под Symbian OS 6.1 работает связка CLDC 1.0/MIDP 1.0, а для Symbian OS 7.0 и выше связка CLDC 1.0/MIDP 2.0.

### 10.3.1. Класс *TextBox*

Класс `TextBox` необходим для создания редактируемого текстового контейнера. Создавая объект класса `TextBox` и присоединяя его к дисплею телефона (то есть, отражая или прорисовывая графический контекст данного класса на экране), вы создаете редактируемую текстовую область для набора букв, цифр и знаков.

Класс `TextBox` имеет один конструктор класса с четырьмя параметрами, а так же набор методов для работы с экземплярами класса `TextBox`. На рис. 10.2 изображен дисплей телефона серии 60 в работе с классом `TextBox`.

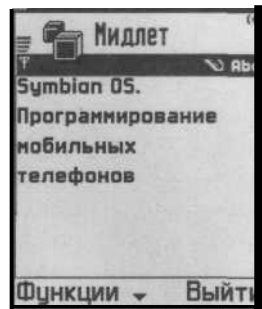


Рис. 10.2. Работа класса `TextBox` на телефоне серии 60

### 10.3.2. Класс *List*

Работа с классом `List` в Java-программах построена на основе выбираемых списков, что собственно и говорит о названии этого класса. Класс `List` реализует возможности интерфейса `Choice` при выборе типа создаваемого списка. Можно создавать эксклюзивный, множественный и одиночный тип списка. Класс `List` содержит два различных по параметрам конструктора служащих для создания объектов этого класса. При создании объектов класса `List` можно загрузить различные кнопки для списков приложения. На рис. 10.3 представлен класс `List` в работе на телефоне серии 60.

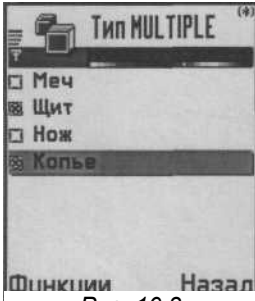


Рис. 10.3.

Работа класса List

на телефоне серии 60

### 10.3.3. Класс Alert

С помощью класса Alert в программах создаются различные уведомления информационного характера. Как правило, это небольшое по размеру диалоговое окно (для UIQ) или полностью задействованный экран (для серии 60), но все зависит от конкретной модели телефона.

Класс Alert может быть использован для предупреждения пользователя о появившейся ошибке или, например, с вопросом о дальнейшей установке программы, то есть класс Alert — это все то, что связано с короткими информационными сообщениями, появляющимися в отдельных и различных по размеру окнах.

При создании объекта класса Alert можно воспользоваться двумя видами конструкторов: с простым текстовым уведомлением и уведомлением с графическим изображением.

### 10.3.4. Класс Form

Класс Form, пожалуй, самый мощный класс из-за своих специфических свойств. Класс Form - это пустая форма, в которую можно встраивать классы из иерархии суперкласса Item. При создании объекта класса Form и его дальнейшем отображении на дисплее телефона, появится экран с белым фоном, то есть пустая форма. Класс Form имеет два конструктора, и предоставляется большой набор методов для работы с объектами класса Form. А, встраивая подклассы суперкласса Item, уже можно создавать программы с красивым интерактивным интерфейсом. При этом, компоная программы, можно встраивать любой необходимый набор подклассов Item, отображая их одновременно на одном экране телефона.

Абстрактный класс Item имеет в своей иерархии восемь подклассов для работы с пользовательским интерфейсом, которые встраиваются в пустую форму (в класс Form). На рис. 10.4 показана иерархия класса Item.

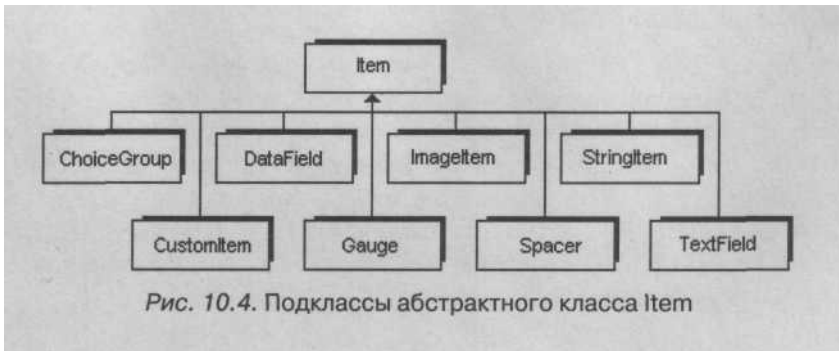


Рис. 10.4. Подклассы абстрактного класса Item

### **10.3.5. Класс *ChoiceGroup***

Класс *ChoiceGroup* служит для интеграции в форму, группу связанных элементов управления, для последующего выбора необходимых действий. Класс *ChoiceGroup* работает на основе класса *Choice* и дает возможность создавать элементы управления типа:

- переключатели;
- флаги;
- списки.

Класс *ChoiceGroup* имеет два конструктора с двумя и четырьмя параметрами для создания объектов класса. При использовании конструктора с четырьмя параметрами, вы сможете загружать иконки для элементов управления, что сделает программу, безусловно, интересней. Так же класс *ChoiceGroup* содержит множество методов для работы с элементами управления.

### **10.3.6. Класс *StringItem***

Экран телефона должен быть представлен классом *Form*, куда встраиваются объекты класса *StringItem*. С помощью класса *StringItem* на экран телефона выводится не редактируемая строка текста.

С помощью методов и констант класса *StringItem* можно оформлять и форматировать текст в виде кнопок или гиперссылок, задавая конкретное место для элементов на экране.

### **10.3.7. Класс *TextField***

Применяя класс *TextField* в своих программах, вы создаете редактируемые текстовые поля с ограничениями. Как правило, класс *TextField* используется для создания адресных книг или полей для ввода текста или цифр. При создании полей с помощью класса *TextField* есть возможность строгого определения атрибута этого поля. Например, можно сделать поле для ввода только дробного числа или пароля, адреса электронной почты, номера телефона, текста, Интернет-адреса, а так же комбинаций текста и цифр. Объекты класса *TextField* интегрируются только в форму созданную на основе класса *Form*.

### **10.3.8. Класс *DateField***

Класс *DateField* используется для работы с датой и временем. Сам класс *DateField* очень простой, а все его функциональные возможности реализованы программно. Класс *DateField* интегрируется в форму на основе класса *Form*.

Создавая объекты класса *DateField* можно применять два конструктора класса *DateField* с двумя и тремя параметрами. При помощи конструктора, состоящего из трех параметров, устанавливается точное время по выбранному часовому поясу.

### **10.3.9. Класс Spacer**

Дословный перевод названия класса Spacer означает «пространство». Не-что подобное класс Spacer и делает, а именно: организует квадрат или прямоугольник пустого пространства, отодвигая элементы формы на заданные значения. Конструктор класса Spacer, необходимый для создания объектов этого класса, включает в себя два параметра: ширину и высоту создаваемого пространства в пикселях.

### **10.3.10. Класс *1m ad el tern***

Для загрузки изображений в программу необходимо воспользоваться клас-сом Image Item. В Java 2 ME-программах при работе с изображениями применяется формат PNG (Portable Network Graphics - портативный формат сетевой графики).

Объекты класса Image Item так же интегрируются в форму, представленную классом Form. Загрузка различных изображений, заставок, фона, фотографий значительно «оживляют» внешний вид приложения.

### **10.3.11. Класс Gauge**

Класс Gauge служит для создания измерителей различных процессов на основе интерактивных графических элементов. Обычно класс Gauge применяется при загрузке программ, игр, файлов. В тот момент, когда пользователю необходимо выждать процесс удаления или загрузки появляется измеритель, показывающий графически данный процесс.

Графически класс Gauge воссоздается на экране телефона как широкая линия, постепенно заполняющаяся определенным цветом по мере прохождения процесса. Но в зависимости от производителей телефонов графический контекст класса Gauge может изменяться по-разному.

### **10.3.12. Класс Custom Item**

Класс Custom Item из иерархии суперкласса Item относительно новый класс и добавлен в профиль MIDP 2.0. На основе класса CustomItem программист может построить в форму нестандартный графический контекст. Это могут быть таблицы построенные, например, по типу Windows программы Excel. Более того, класс CustomItem обладает универсальной возможностью работать с событиями, происходящими через нажатия клавиш телефона.

Платформа Java 2 ME имеет еще огромное количество классов, не входящих в иерархию классов пользовательского интерфейса, но предназначенных именно для упрощения работы по созданию пользовательского интерфейса. Рассмотрим некоторые из них.

### 10.3.13. Класс Font

Класс Font из пакета javax.microedition.lcdui.\* предлагает программисту набор системных шрифтов для использования в приложениях. Java 2 ME имеет следующие способы отображения шрифтов на экране телефона.

Размерность шрифта:

- большой шрифт;
- средний шрифт;
- маленький шрифт.

Стиль шрифта:

- жирный шрифт;
- курсив;
- обычный шрифт;
- подчеркнутый шрифт.

Начертание шрифта:

- стандартный шрифт;
- пропорциональный шрифт;
- шрифт с небольшим интервалом.

Вполне достойный набор возможностей для отображения шрифта на экране в Java 2 ME-программах.

### 10.3.14. Класс Ticker

Не всегда востребованный, но очень интересный класс Ticker создает в приложении бегущую строку по верхней кромке экрана. Бегущая строка появляется на экране справа и движется налево циклично с одинаковой скоростью.

## 10.4. Низкоуровневый пользовательский интерфейс

В отличие от высокоуровневых классов, *низкоуровневые классы пользовательского интерфейса* предназначены для работы с графикой и для обработки событий, поступающих с клавиш телефона. Шаблонные классы высокоуровневого интерфейса содержат средства для создания стандартных списков, диалогов, элементов управления и так далее. При работе с этими классами программисту не нужно заботиться о размере, например, диалогового окна или списка, Java 2 ME выполнит эту задачу сама. А вот при работе с низкоуровневыми классами ситуация кардинально меняется и вам придется следить за разрешением дисплея телефона. Каждый производитель телефона придерживается своей конструкции по созданию и разработке внешнего вида и аппаратной части телефона. Но если в отношении клавиш телефона уже имеются сложившиеся стандарты, например, в играх используются клавиши под цифрами 2, 4, 8, 6 и 5 либо джойстик, то с разрешением экрана ситуация противоположная.

Одно из главных человеческих качеств - это восприятие окружающего мира при помощи глаз. Мы можем воспринимать окружающий нас мир таким, какой он есть, то есть объемным (трехмерным) и цветным с огромным, пока не доступным для телефонов и мониторов разрешением «экрана» (если можно так выразиться). И человеку с его способностями хочется такого же большого и качественного экрана в телефоне. Поэтому каждый производитель на свой лад пытается улучшить дисплей телефона, увеличивая разрешение и битность экрана. В связи с этим размеры дисплея изменяются от модели к модели даже одного производителя. При программировании графики средствами Java 2 ME необходимо следить за размерами дисплея телефона. Например, возьмем Sony Ericsson 900 с разрешением 208 x 320 пикселей и Nokia 6600 с разрешением 176 x 208 пикселей - два разных размера экрана и соответственно различные возможности отрисовки графических элементов. Поэтому при программировании графики в Java 2 ME под Symbian OS необходимо очень внимательно планировать работу с низкоуровневыми классами, не забывая так же и об используемых версиях профиля и конфигурации.

### **10.4.1. Класс Canvas**

Абстрактный класс Canvas представляет собой обобщенный графический контекст экрана телефона. Для отрисовки графики необходимо создавать свои классы, наследуемые от класса Canvas. В состав этого класса входит большое количество методов для обработки событий, полученных с клавиш телефона. Обработка событий с клавиш телефона происходит на основе «ключевых кодов» или констант, жестко закрепленных за каждой из клавиш. Построение графики происходит при помощи класса Graphics.

### **10.4.2. Класс Graphics**

С применением класса Graphics происходит прорисовка или представление на экране телефона графики. Построение графики в Java 2 ME основывается на двухмерной системе координат, изображенной на рис. 9.2 в главе 9.

В класс Graphics включены разнообразные методы для работы с графикой. Применяя методы этого класса можно рисовать линии, прямоугольники, шрифты, сферы, задавать цвет.

Класс Graphics может использоваться и в профиле MIDP 1.0, и в профиле MIDP 2.0, но в первой версии MIDP работа с классами Graphics и Canvas несколько отличается от MIDP 2.0. В частности, при создании игр в MIDP 1.0 применяется свой игровой цикл, разбитый на несколько потоков и усложняющий работу с графикой. Посмотрите на исходный код игрового цикла для профиля MIDP 1.0.

```
public class DemoGraphics extends Canvas implements Runnable
{
public void run()
```



```

while (true)
{
// обновление графических элементов
repaint ();
// задержка цикла на 20 миллисекунд Thread.sleep(20);

public void paint( Graphics g
) {
// код, создающий графические элементы }
public void keyPressed( int keyCode )
{
// обработка событий с клавиш телефона

```

В приведенном программном коде игровой цикл разбит на три потока в методах run (), paint () и keyPressed (). Значительно проще реализован тот же самый цикл, но уже для профиля MIDP 2.0. Но, к сожалению, MIDP 2.0 недоступен для Symbian OS 6.1.

### **10.4.3. Класс GameCanvas**

Абстрактный класс GameCanvas, как видно из названия, ориентирован на создание мобильных игр. Этот класс доступен только в профиле MIDP 2.0. В иерархии этого класса содержатся еще четыре класса:

- Layer,
- Layer Manager,
- TiledLayer,
- Sprite.

Схема игрового цикла класса GameCanvas значительно усовершенствована и выглядит следующим образом:

```

public void run() {
Graphics g = getGraphics();
while(true)
{
// метод, обрабатывающий нажатия клавиш с телефона
inputKey();
// метод, прорисовывающий графику
GameGraphics();
}
}

```

```
// копирует графические элементы на экран из внеэкранного буфера  
flushGraphics();
```

Конструкция игрового цикла класса `GameCanvas` не использует входные системные потоки и содержит один цикл, в теле которого происходит прорисовка графики и обработка событий с клавиш телефона. Применяется также методика двойной буферизации, построенная на работе с внеэкранным буфером, который улучшает качество рисуемой графики.

#### **10.4.4. Класс `Layer`**

Класс `Layer` инкапсулирует основные свойства имеющихся уровней игры. С помощью методов класса `Layer` происходит прорисовка уровней. Класс `Layer` имеет два подкласса `TiledLayer` и `LayerManager` для работы с уровнями.

#### **10.4.5. Класс `TiledLayer`**

Класс `TiledLayer` необходим для создания фоновых изображений в игре. Можно загружать готовые изображения, а можно использовать работу с ячейками для создания фона игры. Игры, написанные на Java 2 ME, строятся по принципу уровней: на одном уровне находится фоновое изображение, на другом препятствия, артефакты и главный герой, при этом количество уровней не ограничено.

#### **10.4.6. Класс `LayerManager`**

А вот класс `LayerManager` играет роль менеджера уровней, отслеживая имеющиеся уровни в игре и накладывая их один на другой, формируя тем самым полноценную игровую сцену. Несколько методов и конструктор класса `LayerManager` дают такую возможность.

#### **10.4.7. Класс `Sprite`**

Класс `Sprite` представляет в игре двухмерное изображение (спрайт). Как правило, это активно используемые анимированные изображения. В качестве объектов класса `Sprite` могут быть прорисованы машины, самолеты, бомбы, люди, звери, главные герои, различные бонусы и так далее.

Работа с классом `Sprite` основана на использовании анимационной последовательности *фреймов* для рисования интерактивной графики. Класс `Sprite` содержит три разных конструктора для создания анимированных и неанимированных спрайтов. С помощью методов класса `Sprite` можно производить различные виды трансформации и переопределение опорной позиции спрайта, а также устанавливать столкновение с другими спрайтами или элементами уровней.

## 10.5. Звуковое сопровождение

Воспроизведение звуковой дорожки значительно улучшает атмосферу игры, да и представить игру без звука невозможно. В Java-приложениях возможна работа с тональными звуками и воспроизведение звуковых файлов.

Воспроизведение тональных звуков построено на генерации определенного тона, совпадающего по тональности с музыкальными нотами. Собирая тональные звуки в последовательную цепочку, можно создать незатейливую мелодию, напоминающую звук в играх для консольных приставок Dendy или Sega

Наиболее интересным вариантом работы со звуком является возможность воспроизведения Java-программами в Symbian OS полноценных звуковых файлов. В таблице 10.1 перечислены все возможные звуковые файлы, поддерживаемые в Java 2 ME и их доступность для каждой платформы.

**Таблица 10.1. Формат поддерживаемых звуковых файлов для Symbian OS**

Аудио/Видео формат	Symbian OS 7.0	Серия 60 1.2	Серия 60 2.1
	UIQMIDP2.0	MIDP1.0	MIDP2.0
Тональные звуки	да	да	да
AU audio (*.au)	да	нет	да
Wave audio (*.wav)	да	да	да
MP3 (*.mp3)	да	нет	нет
AMR audio (*.amr)	нет	да	да
AMR wideband audio (*.awb)	нет	да	нет
Nokia ring tone (*.mg)	нет	да	да
MIDI(*.mid)	нет	да	да
Polyphonic MIDI (*.mid)	нет	нет	да
RAW audio (*.raw)	нет	да	да
3 GPP video (*.3gp)	нет	да	нет
NIM video (*.nim)	нет	нет	да
MP4 video (*.mp4)	нет	нет	да
Real Media video (*.rm)	нет	нет	да

Дополнительно телефон Sony Ericsson поддерживает еще форматы RMF (\*.rmf), iMelody (\*.imy) и MIDI (\*.mid).

## 10.6. Распространение Java 2 ME программ

При написании Java-программ для телефонов, работающих на основе прошивки, очень важной составляющей является размер создаваемой программы. Например, в телефон Nokia 3100 можно загрузить приложения с максимальным размером 44 Кб, а в Sony Ericsson 610-60 Кб. Но для Symbian OS этот фактор не столь критичен.

Программы, написанные для телефонов на Java 2 ME, распространяются в заархивированном виде на двух файлах с расширениями \*.jar и \*.jad.

*JAR-файл* - это скомпонованная, откомпилированная и заархивированная Java-программа. *JAD-файл* - это дескриптор приложения, описывающий основ-

ные атрибуты программы. Создавая Java 2 ME-программу в одном из специализированных инструментариях, необходимо явно упаковать программу для ее дальнейшего распространения. Что касается инструментариев, то существует множество как платных, так и бесплатных средств программирования.

## 10.7. Среда программирования J2ME Wireless Toolkit 2.2

Среда программирования J2ME Wireless Toolkit 2.2, изображенная на рис. 10.6, одна из простейших, а главное бесплатных сред, созданных компанией SUN Microsystems.

Загрузить среду J2ME Wireless Toolkit 2.2 можно с сайта компании SUN Microsystems по Интернет адресу <http://java.sun.com>.

После установки J2ME Wireless Toolkit 2.2 для *создания проекта* воспользуйтесь командами меню **File => New Project**. В появившемся диалоговом окне **New Project** задайте имя проекту и основному классу мидлета. Далее в рабочем каталоге J2 ME Wireless Toolkit будут сформированы следующие папки:

- \src - папка для исходных кодов программы;
- \res - файлы ресурсов программы;
- \lib — подгружаемые библиотечные файлы;
- \classes - папка для проверенных откомпилированных классов;
- \tmpclasses - непроверенные откомпилированные файлы;
- \tmlib — временная папка для библиотек;
- \Ып - папка для файлов JAD, JAR и файла манифеста.

После создания проекта необходимо создать в любом текстовом редакторе исходный код программы, поскольку в J2ME Wireless Toolkit отсутствует встроенный редактор, и разместить коды в папке src.

Для *компиляции проекта* выберите команду в меню **Project => Build**, после этого в паке bin появятся откомпилированные файлы программы. Чтобы *протестировать* созданное приложение на эмуляторе телефона, выберите из меню команду **Project => Run**. Теперь для переноса программы на телефон ее необходимо упаковать. Выберите в меню **Project => Package**, произойдет упаковка имеющейся программы, исходные файлы которой (QAD и JAR) будут находиться в папке bin рабочего каталога J2ME Wireless Toolkit.

## 10.8. Другие средства программирования Java 2 ME приложений

На сайте компании SUN Microsystems можно загрузить последнюю, шестую, версию Java SUN Studio 6 Mobility 2004Q - замечательной среды программирования, распространяемую абсолютно бесплатно, но на основе регистрации. Это полноценная многофункциональная среда программирования, включающая в себя текстовый редактор, компилятор, отладчик и эмуляторы телефонов.

Компания Borland тоже славится своими интегрированными средствами программирования приложений. Специализированная среда для создания Java-программ носит название Borland JBuilderX Enterprise 10 for Windows 2000/XP. Хотя данный продукт не распространяется бесплатно, его ограниченную по времени trial-версию можно загрузить с сайта компании по адресу в Интернете: <http://www.borland.com>.

Среда программирования Borland JBuilderX может работать как с J2ME, так и с J2SE программами, что делает это программное обеспечение универсальным продуктом для решения разносторонних задач.

Компания Metrowerks имеет отличную среду программирования для Java-программ под названием Metrowerks CodeWarrior Wireless Studio 7. Trial-версию этой среды программирования можно найти на сайте компании: <http://www.metrowerks.com>. Среда имеет мощный текстовый редактор, компилятор, отладчик и эмулятор телефона на базе модели Sony Eriksson P800.

Для создания программ на Java 2 ME кроме всех перечисленных средств вам понадобится набор инструментальных средств разработчика Java 2 SDK SE, последние версии которых всегда можно бесплатно загрузить с сайта компании SUN Microsystems. Средства разработчика SDK от различных производителей телефонов находятся на сайте этой компании.

Большинство перечисленных в этой главе средств программирования, а также подробнейший разбор процесса по созданию программ на Java 2 ME, вы можете найти в книге издательства ДМК «Программирование мобильных телефонов на Java 2 Micro Edition».

## Заключение

Вашему вниманию было предложено огромное количество материала, который, конечно, можно было бы разделить на несколько отдельных книг. Но хотелось сделать одну хорошую книгу, где рассматривается цикл по созданию программ для Symbian OS от начала и до конца, включая все имеющиеся SDK и средства программирования. Надеюсь, мне удалось реализовать эту идею в полной мере, рассмотрев все тонкости создания программ для операционной системы Symbian.

По ситуации на рынке телефонов, Symbian OS занимает лидирующее положение в мире. Думается, что ее позиции будут только укрепляться. Как вы убедились, система действительно компактна, стабильна и хорошо реализована. Минимальные системные ресурсы, необходимые для работы Symbian OS, значительно уменьшают стоимость конечного продукта. А продуманная система лицензирования и отделение графического контекста от ядра системы заинтересовывает все новых и новых производителей мобильных телефонов.

# Приложение 1. Справочник

Системная библиотека Symbian OS - это одна из важнейших частей всей платформы. Применяя системные классы, можно реализовать приложение практически любой сложности. Более того, компания Symbian Ltd настоятельно рекомендует использовать системную библиотеку вместо собственных классов.

Этот справочник создан на основе оригинальной документации к Symbian OS 7.0s. Документация доступна в SDK и на сайте компании Symbian Ltd по адресу в Интернет: <http://www.symbian.com>.

Библиотека разделена на разделы, или подсистемы, где собраны компоненты для решения определенной задачи. Насчитывается более ста подсистем, которые имеют в своем составе более тысячи различных классов. Каждый класс содержит свой набор функций, макросов, констант и так далее. Весь этот огромный функционал рассмотреть невозможно, для этого потребовалась отдельная книга примерно в тысячу страниц. Поэтому в справочнике дается краткая характеристика основным классам, интерфейсам и перечисляемым типам для каждой подсистемы. На основе этой информации вам будет легко разобраться непосредственно с составляющими того или иного класса, интерфейса или типа.

Кроме этого, как вы уже знаете, Symbian OS имеет систему предупреждения сбой (System Panic), которая так же отражена в оригинальной справочной системе Symbian OS. Поскольку система предупреждений очень проста, вы сможете разобраться с этим вопросом самостоятельно.

## 1.1. Graphics 2D Hardware Acceleration

`CGraphicsAccelerator` - абстрактный класс для работы с двухмерной графикой. Его использование значительно увеличит скорость работы с графикой. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `graphicsaccelerator.h`.

`CHardwareGraphicsAccelerator` - осуществляет графическое ускорение на аппаратном уровне. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `graphicsaccelerator.h` и библиотечный файл `scdv.lib`.

`CSoftwareGraphicsAccelerator` - осуществляет графическое ускорение программными средствами. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `graphicsaccelerator.h` и библиотечный файл `bitgdi.lib`.

`RHardwareBitmap` - класс для работы с точечными рисунками на аппаратном уровне. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `graphicsaccelerator.h` и библиотечный файл `scdv.lib`.

TAcceleratedBitmapInfo - структура данных, считывающая информацию о точечном рисунке. Доступна в Symbian OS 7.0. Для работы необходимо подключить заголовочный файл graphicsaccelerator.h.

TAcceleratedBitmapSpec - класс утилит, обеспечивающий доступ к спецификации точечного рисунка. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл graphicsaccelerator.h и библиотечный файл bitgdi.lib.

TBitmapLockCount - производит подсчет количества блокировок, сделанных с помощью объектов класса TAcceleratedBitmapSpec. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл graphicsaccelerator.h.

TGopBitBltAlphaBitmap - класс, ускоряющий процесс копирования точечного рисунка в заданный регион экрана, можно использовать альфа-канал. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл graphicsaccelerator.h.

TGopBitBltAlphaChannel - класс, ускоряющий процесс копирования точечного рисунка в заданный регион экрана с последующим смешиванием цвета по альфа-каналу (alpha-blending). Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл graphicsaccelerator.h.

TGopBitBlt - класс, ускоряющий процесс копирования точечного рисунка в заданный регион экрана. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл graphicsaccelerator.h.

TGopBitBltMasked - класс, ускоряющий процесс копирования точечного рисунка в заданный регион экрана с использованием операции маскирования. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл graphicsaccelerator.h.

TGopBitBltTransparent - класс, ускоряющий процесс копирования точечного рисунка в заданный регион экрана с возможностью изменения пикселей исходного изображения. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл graphicsaccelerator.h.

TGopFadeParams - определяет постепенное специфическое изменение цветов в прямоугольной области. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл graphicsaccelerator.h.

TGopFadeRect - ускоренное изменение цветов в заданной прямоугольной области. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл graphicsaccelerator.h.

TGopFilledPolygon - графическое ускорение операции по заполнению полигонов с цветом. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл graphicsaccelerator.h и библиотечный файл bitgdi.lib.

TGopFilledPolygonWithPattern - класс для ускоренной операции по заполнению полигонов модели. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл graphicsaccelerator.h.



TGopFilledRect - класс для ускоренной операции по заполнению прямоугольной области цветом. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл graphicsaccelerator.h.

TGopFilledRectUsingDrawMode - класс для ускоренной операции по заполнению прямоугольной области цветом с использованием пароразрядных операторов. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл graphicsaccelerator.h.

TGopFilledRectWithPattern - ускоренная графическая операция по заполнению прямоугольной области точечными рисунками. Доступна от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл graphicsaccelerator.h.

TGopFillPattern - представляет точечный рисунок для использования графическим ускорителем. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл graphicsaccelerator.h.

TGopInvertRect - класс, инвертирующий цвет пикселей в прямоугольной области. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл graphicsaccelerator.h.

TGopScaledBitBltAlphaBitmap - класс, копирующий прямоугольную область точечного рисунка в другой рисунок с применением alpha-blending. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл graphicsaccelerator.h.

TGopScaledBitBltAlphaChannel - класс, копирующий прямоугольную область точечного рисунка в заданную прямоугольную область с применением alpha-blending. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл graphicsaccelerator.h.

TGopScaledBitBlt - класс, ускоряющий копирование прямоугольной области точечного рисунка в другую область. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл graphicsaccelerator.h.

TGopScaledBitBltMasked - класс, ускоряющий копирование прямоугольной области точечного рисунка в другую область, используя третий точечный рисунок для маскирования. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл graphicsaccelerator.h.

TGopScaledBitBltTransparent - ускоряет копирование прямоугольной области точечного рисунка в другую область, с прозрачными пикселями исходного рисунка. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл graphicsaccelerator.h.

TGopTransparency - класс, определяет прозрачность заданных пикселей. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл graphicsaccelerator.h.

TGraphicsAcceleratorCaps - сохраняет возможности графического ускорителя в памяти. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл graphicsaccelerator.h.

TGraphicsOperation - абстрактный базовый класс для работы с графическими операциями. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл graphicsaccelerator.h.

TTransparencyType - перечисляемый тип для работы с прозрачностью цветов. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл graphicsaccelerator.h.

## 1.2. Agenda Entry and Instance

CAgnAnniv — класс, позволяет занести в список событие, которое должно произойти в течение года. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmentry.h и библиотечный файл agnmodel.lib.

CAgnAppt - класс, создает тип записи, которая имеет начало и окончание срока действия согласно системному времени. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmentry.h и библиотечный файл agnmodel.lib.

CAgnAttendee - класс, предоставляет возможность создания групповой записи. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmxentr.h и библиотечный файл agnmodel.lib.

CAgnBasicEntry — абстрактный базовый класс для создания типовых задач. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmbasic.h и библиотечный файл agnmodel.lib.

CAgnCategory - класс, создает запись задачи определенной категории. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл agmxentr.h и библиотечный файл agnmodel.lib.

CAgnEntry - абстрактный базовый класс для записи различного рода информационных сообщений. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmentry.h и библиотечный файл agnmodel.lib.

CAgnEvent - класс, создает запись события, выполнение которого должно произойти в назначенную дату. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmentry.h и библиотечный файл agnmodel.lib.

CAgnRptDef - класс, создает повторяющиеся события для определенной записи. Доступен от версии Symbian OS 5.1. Для работы необходимо подключить заголовочный файл agmrptd.h и библиотечный файл agnmodel.lib.

TAgnBasicAppt - использует класс CAgnAppt для хранения деталей записи. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmbasic.h и библиотечный файл agnmodel.lib.

TAgnBasicEvent — использует класс CAgnAppt для хранения деталей записи определенного события. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл gmbasic.h и библиотечный файл agnmodel.lib.

TAgnDailyRpt - ежедневное повторение событий. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmrptd.h и библиотечный файл agnmodel.lib.

TAgnEntryId - класс для идентификации записи в файле. Доступен от версии Symbian OS 5.1. Для работы необходимо подключить заголовочный файл agmids.h и библиотечный файл agnmodel.lib.

TAgnException - класс, сохраняет данные, исключаяющие повторение записей. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmexcpt.h и библиотечный файл agnmodel.lib.

TAgnId - базовый класс для создания идентификационного (ID) списка. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmids.h и библиотечный файл agnmodel.lib.

TAgnInstanceDateTimeld - класс, устанавливающий начало и окончание события по экземпляру ID. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmids.h и библиотечный файл agnmodel.lib.

TAgnInstanceld - класс, содержащий идентификатор. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmids.h и библиотечный файл agnmodel.lib.

TAgnLiteEntry - класс, устанавливающий запрос к функциям, дающим возможность чтения занесенных в список деталей. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmcomon.h и библиотечный файл agnmodel.lib.

TAgnMonthlyByDatesRpt - класс для хранения даты, которая должна повторяться в течение месяца. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmrptd.h и библиотечный файл agnmodel.lib.

TAgnMonthlyByDaysRpt - класс для хранения дня, который должен повторяться в течение месяца. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmrptd.h и библиотечный файл agnmodel.lib.

TAgnMonthlyRpt - класс, определяющий общее поведение при совместной работе двух типов TAgnMonthlyByDaysRpt и TAgnMonthlyByDatesRpt. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmrptd.h и библиотечный файл agnmodel.lib.

TAgnReplicationData - класс для синхронизации данных. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmrepli.h и библиотечный файл agnmodel.lib.

TAgnRpt - абстрактный базовый класс для типов, работающих с повторениями событий. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmrptd.h и библиотечный файл agnmodel.lib.

TAgnUniqueld - уникальный идентификатор для работы с agenda файлами. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmids.h и библиотечный файл agnmodel.lib.

TAgnWeeklyRpt - класс, обеспечивающий еженедельное повторение событий. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmrptd.h и библиотечный файл agnmodel.lib.

TAgnYearlyByDateRpt - класс для повторяющихся каждый год событий по установленной дате. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmrptd.h и библиотечный файл agnmodel.lib.

TAgnYearlyByDayRpt - класс для повторяющихся каждый год событий по установленному дню. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmrptd.h и библиотечный файл agnmodel.lib.

### 1.3. Agenda File

CAgnAlarm - класс для работы с оповещениями. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmalarm.h и библиотечный файл agnmodel.lib.

CAgnDayDateTimeInstanceList — список AgnInstanceDateTimeld для одного дня. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmlists.h и библиотечный файл agnmodel.lib.

CAgnDayList - список элементов для одного дня. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmlists.h и библиотечный файл agnmodel.lib.

CAgnEntryModel - базовый класс для CAgnIndexedModel и CAgnModel, необходимый для обращения к сохраненным в файле данным. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agnmodel.h и библиотечный файл agnmodel.lib.

CAgnIndexedModel - класс для индексации данных. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agnmodel.h и библиотечный файл agnmodel.lib.

CAgnList - класс для работы со списком. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmlists.h и библиотечный файл agnmodel.lib.

CAgnModel - класс для работы с agenda model. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agnmodel.h и библиотечный файл agnmodel.lib.

CAgnMonthInstanceList - список заданий на один месяц. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmlists.h и библиотечный файл agnmodel.lib.

CAgnObserver - абстрактный базовый класс для наблюдения за agenda model. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmobsrv.h и библиотечный файл agnmodel.lib.

CAgnSymbolList - список дней в месяце, содержащий два символа. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmlists.h и библиотечный файл agnmodel.lib.

CAgnSyncIter - класс для синхронизации с agenda model по итераторам. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmsiter.h и библиотечный файл agnmodel.lib.

MAgnActiveStep - базовый класс для CAgnEntryModel. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmactiv.h.

MAgnModelStateCallBack - класс для повторного вызова agenda model. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmcallb.h и библиотечный файл agnmodel.lib.

MAgnProgressCallBack - представляет информацию о продвижении и окончании процесса. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmcallb.h и библиотечный файл agnmodel.lib.

MAgnVersit - предусматривает статическую зависимость CAgnEntryModel. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agnmodel.h.

TAgnDayFilter - класс для фильтрации событий. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmfilts.h и библиотечный файл agnmodel.lib.

TAgnEntrylter - класс для итерации входных значений в agenda file. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmmiter.h и библиотечный файл agnmodel.lib.

TAgn Filter- идентификационный класс. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmfilts.h и библиотечный файл agnmodel.lib.

TAgnsrvFindFilter - класс для фильтрации agenda model по входным одиночным символам. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmfilts.h и библиотечный файл agnmodel.lib.

TAgnsrvTidyFilter - фильтр неотмеченных данных. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmfilts.h и библиотечный файл agnmodel.lib.

TAgnSymbolFilter - фильтр для дней, дополнительно сохраняющих символ. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmfilts.h и библиотечный файл agnmodel.lib.

TAgnWhichInstances - указывает на образцы после их модификации. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmpact.h.

#### **1.4. Agenda Server Client Side**

RAgendaServ - клиентский интерфейс для сервера списков основных операторов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agclient.h и библиотечный файл agnmodel.lib.

#### **1.5. Agenda Model Utilities**

AgnDateTime - класс для обеспечения функции по преобразованию даты и времени между форматами. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmdate.h и библиотечный файл agnmodel.lib.

TAgnDateTime - класс контейнер для дат и времени. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmdate.h и библиотечный файл agnmodel.lib.

TAgnDate - класс для работы с датой. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmdate.h и библиотечный файл agnmodel.lib.

TAgnTime - класс для работы со временем. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmdate.h и библиотечный файл agnmodel.lib.

TAgnVersion - класс для определения номера версии agenda model. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmver.h и библиотечный файл agnmodel.lib.

## **1.6. Alarm Server**

AlarmClientUtils - этот класс содержит основные утилиты. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл ASCLiClientUtils.h и библиотечный файл AlarmClient.lib.

ASCLiDef initions - класс для восстановления номера версии сервера оповещений. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл ASCLiDefinitions.h и библиотечный файл AlarmClient.lib.

ConsoleAlarmAlertServer - внутренний класс, не предназначен для общего использования.

RAlarmServer - определяет клиентское API для создания оповещений. Доступен в Symbian OS с 5.0 по 6.1 и в 7.0s. Для работы необходимо подключить заголовочный файл t32alm.h и библиотечный файл ealwl.lib.

RASCLiSession - клиентский интерфейс оповещений. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл ASCLiSession.h и библиотечный файл AlarmClient.lib.

TAlarmInf o - сохраняет информацию оповещений. Доступен в Symbian OS с 5.0 по 6.1 и в 7.0s. Для работы необходимо подключить заголовочный файл t32wld.h.

TAlmSoundPlay - класс для воспроизведения сигнала. Доступен в Symbian OS с 5.0 по 6.1 и в 7.0s. Для работы необходимо подключить заголовочный файл t32alm.h.

TASCLiSoundPlayDef inition - класс, определяющий, когда и насколько будет запущен звуковой сигнал оповещения. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл ASCLiSoundPlay.h и библиотечный файл AlarmClient.lib.

TASShdAlarm - создает объект, содержащий информацию о предстоящем оповещении. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл ASShdAlarm.h и библиотечный файл AlarmSha-red.lib.

TBitFlagsT - класс-шаблон для определения клиентских установок. Для работы необходимо подключить заголовочный файл ASShdBitFields.h и библиотечный файл AlarmClient.lib.

TAlarmCategory - перечисляемый тип для задания уникального идентификатора. Для работы необходимо подключить заголовочный файл ASShdDefs.h и библиотечный файл AlarmClient.lib.

TAlarmCharacteristicsFlags - перечисляемый тип для задания флагов, определяющих вид оповещения. Для работы необходимо подключить заголовочный файл ASShdDefs.h и библиотечный файл AlarmClient.lib.

TAlarmId - перечисляемый тип, уникальный идентификатор оповещения. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл ASShdDefs.h и библиотечный файл AlarmClient.lib.

TAlarmSoundName - перечисляемый тип, сохраняющий название запущенного звукового файла оповещения. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл ASShdDefs.h и библиотечный файл AlarmClient.lib. В версии Symbian OS 7.0 для работы подключается заголовочный файл t32alm.h.

TAlarmMessage - перечисляемый тип, сохраняющий уведомления о текстовом сообщении в буфере. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл ASShdDefs.h и библиотечный файл AlarmClient.lib. В версии Symbian OS 7.0 для работы подключается заголовочный файл t32alm.h.

TASAltClientServerStateFlags - внутренний перечисляемый тип, не предназначен для общего использования.

TBitFlags16 - перечисляемый тип, представляющий 16-битный флаг. Для работы необходимо подключить заголовочный файл ASShdBitFields.h и библиотечный файл AlarmClient.lib.

TBitFlags32 - перечисляемый тип, представляющий 32-битный флаг. Для работы необходимо подключить заголовочный файл ASShdBitFields.h и библиотечный файл AlarmClient.lib.

TBitFlags8 - перечисляемый тип, представляющий 8-битный флаг. Для работы необходимо подключить заголовочный файл ASShdBitFields.h и библиотечный файл AlarmClient.lib.

TBitFlags - перечисляемый тип, представляющий простой флаг. Для работы необходимо подключить заголовочный файл ASShdBitFields.h и библиотечный файл AlarmClient.lib.

TAlarmChangeEvent - перечисляемый тип для определения событий. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл ASShdDefs.h и библиотечный файл AlarmClient.lib.

TAlarmCharacteristics - перечисляемый тип, определяющий сигнальные характеристики. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл ASShdDefs.h и библиотечный файл AlarmClient.lib.

TAlarmClockRepeat - перечисляемый тип для определения частоты повторения сигнала. Доступен в Symbian OS 5.0 - 6.1 и в 7.0s. Для работы необходимо подключить заголовочный файл t32alm.h.

TAlarmDayOrTimed - перечисляемый тип для задания сигнала оповещения. Для работы необходимо подключить заголовочный файл ASShdDefs.h и библиотечный файл AlarmClient.lib.

TAlarmGlobalSoundState — перечисляемый тип для определения состояния звукового сигнала. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл ASShdDefs.h и библиотечный файл AlarmClient.lib.

TAlarmRepeatDefinition - перечисляемый тип, определяющий повторение оповещений. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл ASShdDefs.h и библиотечный файл AlarmClient.lib.

TAlarmServerInitiatedClientPanic - перечисляемый тип для идентификации паники при клиентском сеансе работы. Для работы необходимо подключить заголовочный файл ASShdDefs.h и библиотечный файл AlarmClient.lib.

TAlarmSetState - перечисляемый тип, указывающий на имеющиеся оповещения. Доступен в Symbian OS 5.0 - 6.1 и в 7.0s. Для работы необходимо подключить заголовочный файл t32alm.h.

TAlarmSoundState — перечисляемый тип, представляющий установленные состояния звукового оповещения. Доступен в Symbian OS 5.0 - 6.1 и в 7.0s. Для работы необходимо подключить заголовочный файл t32alm.h.

TAlarmState - перечисляемый тип, представляющий состояние оповещения. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл ASShdDefs.h.

TAlarmStatus - перечисляемый тип, определяющий доступность имеющегося уведомления. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл ASShdDefs.h.

TAlarmType - перечисляемый тип для определения типов тревоги. Доступен в Symbian OS 5.0 - 6.1 и в 7.0s. Для работы необходимо подключить заголовочный файл t32alm.h.

TASAlertServerResponse - внутренний перечисляемый тип, не предназначен для общего использования.

TASAlertOpCode - внутренний перечисляемый тип, не предназначен для общего использования.

TASAlertStateFlags - внутренний перечисляемый тип, не предназначен для общего использования.

## **1.7. Animation**

CAnim - базовый класс для работы с анимацией. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл w32adll.h.

CAnimDll - класс, содержащий одну виртуальную функцию для создания объектов анимации. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл w32adll.h.

CAnimFunctions - класс, содержащий функции-утилиты для работы с анимацией. Доступен от версии Symbian OS 5.0 (в версии 5.1 не доступен). Для работы необходимо подключить заголовочный файл w32adll.h.



CAnimGc - класс, представляющий графический контекст. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл w32adll.h.

CFreeTimerWindowAnim - класс-таймер. Доступен от версии Symbian OS 5.1. Для работы необходимо подключить заголовочный файл w32adll.h.

CreateCAnimDll () - функция, необходима для определения Window Server анимации в DLL. Доступна от версии Symbian OS 5.0. Место расположения в wsanimu.def.

CSpriteAnim — класс, представляющий спрайт в анимации. Доступен от версии Symbian OS 5.1. Для работы необходимо подключить заголовочный файл w32adll.h.

CWindowAnim - класс для создания анимации. Доступен от версии Symbian OS 5.1. Для работы необходимо подключить заголовочный файл w32adll.h.

MAnimFreeTimerWindowFunctions - класс утилит для работы с таймером. Доступен от версии Symbian OS 5.1. Для работы необходимо подключить заголовочный файл w32adll.h.

MAnimGeneralFunctions - общий класс утилит для работы с анимацией. Доступен от версии Symbian OS 5.1. Для работы необходимо подключить заголовочный файл w32adll.h.

MAnimSpriteFunctions - класс утилит для работы с анимацией на основе спрайтов. Доступен от версии Symbian OS 5.1. Для работы необходимо подключить заголовочный файл w32adll.h.

MAnimWindowFunctions - класс, управляющий окном, где происходит непосредственная работа с анимацией. Доступен от версии Symbian OS 5.1. Для работы необходимо подключить заголовочный файл w32adll.h.

MEventHandler - класс для обработки событий. Доступен от версии Symbian OS 5.1. Для работы необходимо подключить заголовочный файл w32adll.h.

TWindowInfo - класс для сохранения положения окна и режим визуального отображения анимации. Доступен от версии Symbian OS 5.1. Для работы необходимо подключить заголовочный файл w32adll.h.

## **1.8. Application architecture framework**

CAraApplication - класс, определяющий основное поведение приложения. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл arappc.h и библиотечный файл arappc.lib.

CAraCommandLine - класс, содержащий некоторые информационные данные, необходимые для запуска приложения. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл aracmdln.h и библиотечный файл arappc.lib.

CAraDocument - базовый класс для всех документов, определяет вид и поведение документа. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл arappc.h и библиотечный файл arappc.lib.

CAraDoorBase - базовый класс для представления внедряемого в программу документа. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл aradbase.h и библиотечный файл arapagc.HB.

CAraModelDoor - класс для управления внедренными в приложение данными. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл aramdr.h и библиотечный файл arapagc.HB.

CAraModelHeader - класс-«обертка» для модели приложения. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл aramdr.h.

CAraProcess - класс для работы с ресурсами документа. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл arapagc.h и библиотечный файл arapagc.HB.

CAraRecentFile - класс с набором функций для управления списком доступных документов. Доступен от версии Symbian OS 5.0, в версии Symbian OS 7.0s не поддерживается. Для работы необходимо подключить заголовочный файл arapagc.h и библиотечный файл arapagc.HB.

KUIdApp - тип для определения универсального идентификатора. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл aradef.h.

KUIdPictureTypeDoor - тип для определения универсального идентификатора изображения. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл aradef.h.

KAppUidValue - тип, представляющий универсальный идентификатор для определения DLL, являющийся App Ш. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл aradef.h.

MAraEmbeddedDocObserver - интерфейс класса для обработки внедренных данных документа. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл arapagc.h.

MAraModelHeaderFactory - интерфейс класса для создания в приложении модели простых объектов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл aramdr.h.

TAraAppCapability - класс, определяющий основные возможности приложения. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл araid.h и библиотечный файл arapagc.HB.

TAraAppCapabilityBuf - тип, необходимый для упаковки класса TAraAppCapability. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл araid.h.

CAraAppCaptionFileReader - внутренний класс, не предназначен общего для использования.

TAraAppCaption - класс для определения, модифицированного дескриптора содержащегося в буфере. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл aradef.h.

TAraAppGroupName - тип для разбиения приложений на группы. Доступен от версии Symbian OS 7.0s. Для работы необходимо подключить заголовочный файл APADEF.h.

TAppEntry - класс, обеспечивающий вход в приложение. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `appid.h` и библиотечный файл `apparc.lib`.

TAppIdentifier — класс для идентификации приложения. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `appid.h` и библиотечный файл `apparc.lib`.

TAppInfo - класс, содержащий информацию о приложении. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `appid.h` и библиотечный файл `apparc.lib`.

TAppCommand - перечисляемый тип для определения командных кодов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `apgtask.h`.

TAppLastUsedEntry — класс для содержания списка файла документа. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `apparc.h` и библиотечный файл `apparc.lib`.

TAppModelDoorFactory - класс, использующий входную модель восстановления приложений. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `apamdr.h` и библиотечный файл `apparc.lib`.

## 1.9. Application architecture services

CAAppInfoFileReader - класс для чтения AIF-файлов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `argicnfl.h` и библиотечный файл `arggfx.lib`.

CAAppDoor - класс-«обертка» для внедренных в приложение документов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `argdoor.h` и библиотечный файл `arggfx.lib`.

CAAppMaskedBitmap — класс, отождествляющий иконку приложения. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `argicnfl.h` и библиотечный файл `arggfx.lib`.

CAAppViewArray - тип для определения массива объектов. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `appid.h`.

CAAppSystemControl - расширенный класс для контроля DLL. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `argctl.h` и библиотечный файл `arggfx.lib`.

CAAppWindowGroupName — класс, предоставляющий доступ к названию групп. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `argwgnam.h` и библиотечный файл `arggfx.lib`.

CAAppSession - класс, обеспечивающий сеанс с сервером. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `argcli.h` и библиотечный файл `arggfx.lib`.

TAppViewInfo - класс, содержащий базовую информацию об application view. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл aroid.h и библиотечный файл apparc.lib.

TAppSystemEvent - класс для определения системных событий. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл apgtask.h.

TAppTask - определенная задача для приложения. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл apgtask.h и библиотечный файл arggfx.lib.

TAppTaskList - класс для обращения к задачам приложения. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл apgtask.h и библиотечный файл arggfx.lib.

## 1.10. Application Utilities

Baf lutils — класс, содержащий набор утилит для работы с файловой системой. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл bautils.h и библиотечный файл baf1.lib.

## 1.11. Array keys

TKey — абстрактный класс для определения характеристик используемых клавиш. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

TKeyArrayFix - класс для определения характеристик ключевых кодов, используемых с элементами фиксированной длины. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32base.h и библиотечный файл euser.lib.

TKeyArrayVar - класс для определения характеристик ключевых кодов, используемых с элементами переменной длины. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32base.h и библиотечный файл euser.lib.

TKeyArrayPak - класс для определения характеристик ключевых кодов, используемых с элементами, упакованными в массив. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32base.h и библиотечный файл euser.lib.

TKeyCmpText - перечисляемый тип, управляющий сравнением между выбором ключевых дескрипторов или текстовых клавиш. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

TKeyCmpNumeric - перечисляемый тип, управляющий сравнением между выбором числовых клавиш. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

TSwap - абстрактный класс для определения поведения элементов массива. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32base.h и библиотечный файл euser.lib.

## 1.12. Asynchronous Services

CActive - системный базовый класс, задающий уровень абстракции для асинхронных событий. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32base.h и библиотечный файл euser.lib.

CActiveScheduler - класс, управляющий асинхронными запросами активных объектов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32base.h и библиотечный файл euser.lib;

CActiveSchedulerWait - класс для упрощения обработки вложенных циклов. Доступен от версии Symbian OS 7.0s. Для работы необходимо подключить заголовочный файл e32base.h и библиотечный файл euser.lib.

CAsyncOneShot - класс, предназначен для контроля над низкоприоритетными процессами программы и гарантирующий их выполнение. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32base.h и библиотечный файл euser.lib.

CAsyncCallBack - класс, управляющий асинхронными запросами, связанными с функциями возврата. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32base.h и библиотечный файл euser.lib.

CBaActiveScheduler — внутренний класс, не предназначен для общего использования.

CIdle - этот класс предназначен для запуска низкоприоритетных процессов программы, в тот момент, когда высокоприоритетные процессы не работают. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32base.h и библиотечный файл euser.lib.

TCallBack - класс, инкапсулирующий работу с функциями возврата. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

TRequestStatus - класс, индикатор состояния запроса. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

## 1.13. B-Trees

MBtreeKey - класс, интерфейс для сортирования и создания ключей для входа в B-tree. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл s32btree.h и библиотечный файл estor.lib.

RFilePagePool — класс для записи страниц через файл для выполнения интерфейса страничного пула. Для работы необходимо подключить заголовочный файл sb32file.h и библиотечный файл estor.lib.

RSecureStorePagePool — класс, использование зашифрованной памяти для выполнения интерфейса страничного пула. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл s32crypt.h и библиотечный файл estor.lib.

RStorePagePool - класс, использование памяти для выполнения интерфейса страничного пула. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл s32stor.h и библиотечный файл estor.lib.

TBtree - класс для упорядочивания входов с помощью ключевого значения в B-tree структуре. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл s32btree.h и библиотечный файл estor.lib.

TBtreeFixBase - класс, обеспечивающий B-tree для входов фиксированных размеров. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл sb32tree.h и библиотечный файл estor.lib.

### **1.14. Backup Server Client**

CBaBackupSessionWrapper - класс, разрешает создание резервной копии или восстановление файлов. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл babackup.h и библиотечный файл bafflib.

MBackupObserver - класс, интерфейс для резервного обозревателя сервера. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл babackup.h.

MBackupOperationObserver - класс, интерфейс для резервной операции сервера. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл babackup.h.

TBackupOperationAttributes - класс, атрибуты для резервной операции. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл babackup.h.

### **1.15. Basic Types**

Abs - возвращает абсолютное значение. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

CBase - базовый класс для всех классов Symbian OS, работающих с динамической памятью. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32base.h и библиотечный файл euser.lib.

Max — возвращает из двух значений большее. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

Min - возвращает из двух значений меньшее. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

Rng - дает возможность определения значения, которое находится в установленном диапазоне. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

TAllowDuplicates - перечисляемый тип, с помощью которого можно найти дублирующие значения. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

TAny - указатель для любого типа. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32def.h.

TBool - тип для булевых значений. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32def.h.

TChar - класс для работы с символьными данными. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

TCharF - класс для преобразования символов к размеру созданной формы. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

TCharLC - класс для преобразования символов к строчным буквам. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

TCharUC — класс для преобразования символов к верхнему регистру. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

TTrue - перечисляемый тип для определения истины (true). Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

TFalse - перечисляемый тип для определения FALSE. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

TInt16 — тип для определения 16-разрядного целого числа со знаком. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32def.h.

TInt32 - тип для определения 32-разрядного целого числа со знаком. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32def.h.

TInt64 - тип для определения 64-разрядного целого числа со знаком. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

TInt8 - тип для определения 8-разрядного целого числа со знаком. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32def.h.

Tint - тип для определения не менее 32-разрядного целого числа со знаком. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32def.h.

TReal32 - тип для определения 32-разрядного числа с плавающей точкой со знаком. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32def.h.

TReal64 — тип для определения 64-разрядного числа с плавающей точкой со знаком. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32def.h.

TReal - тип для определения 64-разрядного числа с плавающей точкой (идентичен типу TReal 64). Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32def.h.

TRef ByValue — класс-шаблон, инкапсулирующий ссылку на объект. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

TText16 - тип, определяющий 16-разрядный символ без знака. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32def.h.

TText8 - тип, определяющий 8-разрядный символ без знака. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32def.h.

TText - тип, необходим для формирования текстового символа. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32def.h.

TUint16 - тип для определения 16-разрядного целого числа без знака. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32def.h.

TUint32 - тип для определения 32-разрядного целого числа без знака. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32def.h.

TUint8 - тип для определения 8-разрядного целого числа без знака. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32def.h.

TUint - тип для определения не менее 32-разрядного целого числа без знака. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32def.h.

VA\_LIST - тип, определяющий C-подобный массив указателей на TUint8. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32def.h.

## **1.16. Bitmap Animation**

RBitmapAnim - класс для упаковки данных анимации и передачи их серверу окна (Window Server) для последующего представления на экране телефона. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл bmpdecli.h и библиотечный файл bmpanim.lib.

CBitmapAnimClientData - класс для работы с анимацией. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл bmpdecli.h и библиотечный файл bmpanim.lib.

CBitmapFrameData - класс для работы с фреймами анимационной последовательности. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл bmpdecli.h и библиотечный файл bmpanim.lib.

## **1.17. Bitmaps**

CBitmapContext - абстрактный класс, представляющий графический контекст устройства. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл gdi.h.

CBitmapDevice - абстрактный класс, определяющий основные атрибуты для графического контекста устройства. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл gdi.h.



CFbsBitGc - этот класс обеспечивает реализацию графического контекста для классов CGraphicsContext и CBitmapContext. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл bitstd.h и библиотечный файл bmpanim.lib.

CFbsBitmap - класс, менеджер для работы со шрифтом точечного рисунка. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл fbs.h и библиотечный файл fbscli.lib.

CFbsBitmapDevice - специализированный класс, обеспечивающий связь графического устройства с точечным рисунком, находящимся в оперативной памяти. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл bitdev.h и библиотечный файл bitgdi.lib.

CFbsDevice - абстрактный базовый класс для графического устройства, с помощью которого происходит прорисовка шрифта и точечного рисунка. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл bitdev.h и библиотечный файл bitgdi.lib.

CFbsScreenDevice - класс, дающий прямой доступ к экрану телефона без задействования сервера окна (Window Server). Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл bitdev.h и библиотечный файл bitgdi.lib.

SEpocBitmapHeader - класс для содержания информации о точечном рисунке. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл bitmap.h.

TBitmapf ileCompression - тип для проведения компрессии заданного файла. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл bitmap.h.

TBitmapUtil - класс, обеспечивающий наиболее быстрый доступ к точечному рисунку. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл fbs.h и библиотечный файл fbscli.lib.

## **1.18. Bitmap Transform**

CBitmapRotator — класс для вращения рисунка на экране телефона. Доступен от версии Symbian OS 7.0s. Для работы необходимо подключить заголовочный файл BitmapTransforms.h и библиотечный файл BitmapTransforms.lib.

CBitmapScaler - класс для масштабирования точечного рисунка на экране телефона. Доступен от версии Symbian OS 7.0s. Для работы необходимо подключить заголовочный файл BitmapTransforms.h и библиотечный файл BitmapTransforms.lib.

Panic () - внутренняя функция, не предназначена для общего использования.

TBitmapTrans forms Panic - перечисляемый тип для определения ошибок, возникающих при трансформации рисунка. Доступен от версии Symbian OS 7.0s. Для работы необходимо подключить заголовочный файл BitmapTransformsMa-in.h и библиотечный файл BitmapTransformsMain.lib.

## 1.19. Bluetooth HCI Extension

CHciExtensionConduit - активный объект, управляющий определенными командами. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл hciproxy.h и библиотечный файл hciproxy.lib.

MVendorSpecif icHciConduit - класс, обеспечивает интерфейс для получения завершений команды. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл hciproxy.h и библиотечный файл hciproxy.lib.

## 1.20. Bluetooth Security Manager

RBTMan - класс сеанса для доступа к службе безопасности. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл btmanclient.h и библиотечный файл BTManClient.lib.

RBTManSubSession — класс подсеанса для доступа к службе безопасности. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл btmanclient.h и библиотечный файл bluetooth.lib.

RBTSecuritySettings — класс подсеанса службы безопасности для регистрации услуг службы безопасности. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл btmanclient.h и библиотечный файл BTManClient.lib.

TBTServiceSecurity - класс для настроек защиты Bluetooth, включает: универсальный идентификатор, идентификатор протокола, идентификатор канала, требования доступа. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл btmanclient.h и библиотечный файл btmanclient.lib.

## 1.21. Bluetooth Service Discovery Agent

CElementParser - класс для синтаксического анализа входного буфера. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл btsdp.h и библиотечный файл bluetooth.lib.

CSdpAgent - класс обслуживания Bluetooth для запросов на ближнее устройство. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл btsdp.h и библиотечный файл bluetooth.lib.

CSdpAttrIdMatchList - класс для чтения идентификаторов атрибутов ближнего устройства. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл btsdp.h и библиотечный файл bluetooth.lib.

CSdpSearchPattern - класс, содержащий список уникальных универсальных идентификаторов. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл btsdp.h и библиотечный файл bluetooth.lib.

MSdpAgentNotifier - класс для обработки ответов на запросы протокола обслуживания Bluetooth. Доступен от версии Symbian OS 6.1. Для работы не-

обходимо подключить заголовочный файл `btsdp.h` и библиотечный файл `bluetooth.lib`.

`TAttrRange` - структура для задания диапазона значения идентификатора. Доступна от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `btsdp.h` и библиотечный файл `bluetooth.lib`.

## **1.22. Bluetooth Service Discovery Database**

`CSdpAttrValue` - базовый класс для создания атрибутов. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `btsdp.h` и библиотечный файл `bluetooth.lib`.

`CSdpAttrValueBoolean` - класс, содержащий значение булевого атрибута. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `btsdp.h` и библиотечный файл `bluetooth.lib`.

`CSdpAttrValueDEA` - класс, содержащий атрибут, который состоит из последовательности данных. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `btsdp.h` и библиотечный файл `bluetooth.lib`.

`CSdpAttrValueDES` — класс элементов данных атрибута. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `btsdp.h` и библиотечный файл `bluetooth.lib`.

`CSdpAttrValueInt` - класс, содержащий целое число со знаком атрибута размером до 128 битов. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `btsdp.h` и библиотечный файл `bluetooth.lib`.

`CSdpAttrValueList` - класс, содержащий списки элементов данных атрибутов. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `btsdp.h` и библиотечный файл `bluetooth.lib`.

`CSdpAttrValueNil` - класс, нулевой атрибут данных. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `btsdp.h` и библиотечный файл `bluetooth.lib`.

`CSdpAttrValueString` - класс, содержащий значение Text String для кодирования строк. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `btsdp.h` и библиотечный файл `bluetooth.lib`.

`CSdpAttrValueUint` - класс, содержащий целое число без знака атрибута размером до 128 битов. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `btsdp.h` и библиотечный файл `bluetooth.lib`.

`CSdpAttrValueURL` - класс, значение URL-атрибута. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `btsdp.h` и библиотечный файл `bluetooth.lib`.

`CSdpAttrValueUUID` - класс, значение UUID-атрибута размером до 128 битов. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `btsdp.h` и библиотечный файл `bluetooth.lib`.

`MSdpAttributeValueVisitor` - класс для перечисления значений в списке. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `btsdp.h` и библиотечный файл `bluetooth.lib`.

MSdpElementBuilder - класс для создания атрибута со значением множественных элементов. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `btsdp.h` и библиотечный файл `bluetooth.lib`.

RSdp - класс, обеспечивающий сеанс доступа к базе данных. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `btsdp.h` и библиотечный файл `sdpdatabase.lib`.

RSdpDatabase - класс, обеспечивающий сеанс для работы с атрибутами. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `btsdp.h` и библиотечный файл `sdpdatabase.lib`.

RSdpSubSession — внутренний класс, не предназначен для общего использования.

TSdpAttributelD - тип, идентификатор атрибута. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `btsdp.h` и библиотечный файл `bluetooth.lib`.

TSdpElementType - перечисляемый тип элементов данных атрибута. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `btsdp.h` и библиотечный файл `bluetooth.lib`.

TSdpServRecordHandle - тип, дескриптор к сервисной записи. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `btsdp.h` и библиотечный файл `bluetooth.lib`.

TUUIID - класс, содержащий универсальный уникальный идентификатор Bluetooth размером 128 бит. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `btsdp.h` и библиотечный файл `bluetooth.lib`.

### **1.23. Bluetooth Sockets**

TBasebandPageTimePolicy — перечисляемый тип для параметров синхронизации с другим устройством. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `bt_sock.h`.

TBTDevAddr - класс, представляющий 48-битный Bluetooth-адрес. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `bttypes.h` и библиотечный файл `bluetooth.lib`.

TBTL2CAPOptions - перечисляемый тип, являющийся опцией для сокетов. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `bt_sock.h` и библиотечный файл `bluetooth.lib`.

TBTLMIOctls - перечисляемый тип, определяет менеджер ссылок. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `bt_sock.h`.

TBTLMOptions - перечисляемый тип, определяет менеджер ссылок. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `bt_sock.h` и библиотечный файл `bluetooth.lib`.

TBTSockAddr - класс для Bluetooth-адреса сокета. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `bt_sock.h` и библиотечный файл `bluetooth.lib`.

THCIAddSCOConnectionBuf - тип, упаковывает для передачи объект THCIAddSCOConnectionIoctl. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `bt_sock.h` и библиотечный файл `bluetooth.lib`.

THCIAddSCOConnectionIoctl - структура с типами параметров данных. Доступна от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `bt_sock.h` и библиотечный файл `bluetooth.lib`.

THCIConnHandle - тип, дескриптор подключения. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `btypes.h` и библиотечный файл `bluetooth.lib`.

THCIDeviceClassBuf - тип, упаковывающий объект THCIDeviceClassIoctl. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `bt_sock.h` и библиотечный файл `bluetooth.lib`.

THCIDeviceClassIoctl - структура, формирует параметр `Class_o ^Devi ce`. Доступна от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `bt_sock.h` и библиотечный файл `bluetooth.lib`.

THCILocalVersionBuf - тип, упаковывающий объект THCILocalVersionIoctl для передачи. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `bt_sock.h` и библиотечный файл `bluetooth.lib`.

THCILocalVersionIoctl - структура для формирования версии Bluetooth-устройства. Доступна от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `bt_sock.h` и библиотечный файл `bluetooth.lib`.

THCIScanEnableBuf - тип, упаковывающий объект THCIScanEnableIoctl для передачи. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `bt_sock.h` и библиотечный файл `bluetooth.lib`.

THCIScanEnableIoctl - перечисляемый тип, формирует параметр `Scan_Enable`. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `bt_sock.h` и библиотечный файл `bluetooth.lib`.

THCISetEncryptionBuf - этот тип упаковывает объект THCISetEncryptionIoctl для передачи. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `bt_sock.h` и библиотечный файл `bluetooth.lib`.

THCISetEncryptionIoctl - структура для кодирования уровня ссылки, формирует параметр `Encryption_Enable`. Доступна от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `bt_sock.h` и библиотечный файл `bluetooth.lib`.

TInquirySockAddr - класс адреса для сокета, применяемый для отдаленных запросов. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `bt_sock.h` и библиотечный файл `bluetooth.lib`.

TLMDisconnectACLBuf - тип, упаковывающий для передачи объект TLMDisconnectACLIoctl. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `bt_sock.h`.

TLMDisconnectACLIoctl - структура, содержащая адрес удаленного устройства Bluetooth, подлежащего отсоединению. Для работы необходимо подключить заголовочный файл `bt_sock.h`.

Trf commRemotePortParams - класс с параметрами Bluetooth RFCOMM-порта. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `bt_sock.h` и библиотечный файл `bluetooth.lib`.

Trf commRPNTransaction - класс для определения одного или более параметров для Bluetooth RFCOMM-порта. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `bt_sock.h` и библиотечный файл `bluetooth.lib`.

TRPNFlowCtrlMask - перечисляемый тип с флажками для управления потоком данных для Bluetooth-порта. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `bt_sock.h` и библиотечный файл `bluetooth.lib`.

TRPNParameterMask - перечисляемый тип с bitmask-значениями для определения параметров Bluetooth-порта. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `bt_sock.h`.

TRPNValidityMask — перечисляемый тип, содержащий флажки, указывающие на установленные параметры порта. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `bt_sock.h` и библиотечный файл `bluetooth.lib`.

TSetBasebandPolicy - структура, определяет политику защиты для использования с указанным адресом устройства. Доступна от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `bt_sock.h`.

TSetBasebandPolicyBuf - тип, упаковывающий объект TSetBasebandPolicy. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `bt_sock.h`.

## 1.24. Bluetooth UI

TBTBasebandNotificationParams - класс, передает информации о случае подключения уведомителю. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `btextnotifiers.h` и библиотечный файл `btextnotifiers.lib`.

TBTDevAddrPckg - тип, упаковывает TBTDevAddr для перемещения. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `bttypes.h` и библиотечный файл `bluetooth.lib`.

TBTDeviceClass - класс, обслуживающий Bluetooth и определяющий информацию об этом устройстве. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `btdevice.h` и библиотечный файл `btdevice.lib`.

TBTDeviceName - тип, содержит имя буфера Bluetooth. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `btdevice.h` и библиотечный файл `bluetooth.lib`.

TBTDeviceSelectionParams - класс, позволяющий посылать параметры со списком устройств диалогу выбора. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `btextnotifiers.h` и библиотечный файл `btextnotifiers.lib`.

TBTDeviceSelectionParamsPckg - тип, упаковывает для передачи TBTDeviceSelectionParams. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `btextnotifiers.h` и библиотечный файл `btextnotifiers.lib`.

TBTDeviceResponseParams — класс с информацией о выбранном устройстве из диалога выбора. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `btextnotifiers.h` и библиотечный файл `btextnotifiers.lib`.

TBTDeviceResponseParamsPckg - тип, упаковывает для передачи TBTDeviceResponseParams. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `btextnotifiers.h` и библиотечный файл `btextnotifiers.lib`.

## 1.25. Calendar Conversion

CChineseCalendarConverter — класс с функциями конвертации между форматами даты TDateTime и TChinese. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `calendarconverter.h` и библиотечный файл `CCon.lib`.

TArithmeticalCalendar — внутренний класс, не предназначен для общего использования.

TArithmeticalDate - класс с арифметической датой (день, месяц и год), используемой григорианским календарем. Доступен от версии Symbian OS 6.1 по 7.0. Для работы необходимо подключить заголовочный файл `calconv.h` и библиотечный файл `scon.lib`.

TAstronomicalCalendar - внутренний класс, не предназначен для общего использования.

TCalendar - класс для хранения даты в юлианском формате и преобразования в другой календарь с помощью оператора. Доступен от версии Symbian OS 6.1 по 7.0. Для работы необходимо подключить заголовочный файл `cal-convcalendar.h` и библиотечный файл `scon.lib`.

TChineseCalendar - класс для хранения даты в китайском календаре. Доступен от версии Symbian OS 6.1 по 7.0. Для работы необходимо подключить заголовочный файл `calconv.h` и библиотечный файл `scon.lib`.

TChineseDate - класс для хранения одной даты в китайском календаре (цикл, год, месяц и день). Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `calendarconverter.h` и библиотечный файл `CCon.lib`.

TGregorianCalendar — класс для хранения даты в григорианском формате и преобразования в другой календарь. Доступен от версии Symbian OS 6.1 по 7.0.

Для работы необходимо подключить заголовочный файл `calconv.h` и библиотечный файл `ccon.lib`.

## 1.26. Certificate Manager

`CAlgorithmIdentifier` - класс, содержащий идентификатор и другие закодированные параметры. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `signed.h` и библиотечный файл `cryptlib`.

`CCertificate` - базовый класс для сертификации данных. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `signed.h` и библиотечный файл `cryptlib`.

`CCertStore` - класс, интерфейс для операций с сохраненными сертификатами. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `certstore.h` и библиотечный файл `certstore.lib`.

`CCertStoreEntry` - класс, обеспечивающий информацию о сертификате приложения. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `certstore.h` и библиотечный файл `certstore.lib`.

`CCertStoreMapping` - внутренний класс, не предназначен для общего использования.

`CertStore` - класс для создания памяти сертификата, в которой хранятся корневые сертификаты. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `certstore.h` и библиотечный файл `certstore.lib`.

`CPKIXCertChain` - класс, интерфейс для управления сертификатами. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `pkixcertchain.h` и библиотечный файл `x500.lib`.

`CPKIXValidationResult` - класс для формирования результата проверки достоверности. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `pkixcertchain.h` и библиотечный файл `x500.lib`.

`CSigningAlgorithmIdentifier` — класс, состоящий из двух классов: алгоритма обзора и ассиметричного алгоритма. Реализовывает оператор равенства. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `signed.h` и библиотечный файл `cryptlib`.

`CSignedObject` - базовый класс для сертификатов. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `signed.h` и библиотечный файл `cryptlib`.

`CSigningKeyParameters` - класс, содержащий информацию параметра, нужную для некоторых алгоритмов. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `signed.h` и библиотечный файл `cryptlib`.

`CSubjectPublicKeyInfo` - базовый класс, содержит закодированный ключ и алгоритм идентификатора. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `signed.h` и библиотечный файл `cryptlib`.



CValidityPeriod - класс, содержит время, в течение которого сертификат является годным. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл signed.h и библиотечный файл cryptalib.

CX500DistinguishedName - класс, содержащий значение и атрибут объекта. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл x500dn.h и библиотечный файл x500.lib.

CX509AlgorithmIdentifier - класс с идентификатором алгоритма для сертификата X.509. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл x509cert.h и библиотечный файл x509.Hb.

CX509AltNameExt - класс, содержащий дополнительное пространство для информации. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл x509certext.h и библиотечный файл x509.lib.

CX509BasicConstraintsExt - класс с расширением сертификата, который указывает на принадлежность сертификата автору. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл x509certext.h и библиотечный файл x509.lib.

CX509CertChain - абстрактный класс для проверки правильности сертификата. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл x509certchain.h и библиотечный файл x509.lib.

CX509CertExtension - класс с универсальным расширением сертификата X.509. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл x509cert.h и библиотечный файл x509.lib.

CX509Certificate - класс, свидетельство X.509. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл x509cert.h и библиотечный файл x509.lib.

CX509CertPoliciesExt - класс, поясняющий, что означает сигнатура в сертификате. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл x509certext.h.

CX509CertPolicyInfo - класс, определяет политику сертификации. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл x509certext.h и библиотечный файл x509.lib.

CX509DNHDomainParams - класс, содержит значение параметров P и G. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл x509keys.h и библиотечный файл x500.lib.

CX509DNHPublicKey - класс, содержит информацию для смены ключей. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл x509keys.h и библиотечный файл x500.lib.

CX509DHValidationParams - класс, возвращающий указатель. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл x509keys.h и библиотечный файл x500.Hb.

CX509DNSNameSubtree - класс с 8-битными двоичными данными имени DNS. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл x509gn.h и библиотечный файл x509.lib.

CX50 9DomainName - класс для адреса электронной почты и имени домена DNS. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл x509gn.h и библиотечный файл x509.lib.

CX50 9DSAParameters - класс добавляет блокирование к назначенной схеме кодирования, оставляя суперклассы независимыми. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл x509keys.h и библиотечный файл x500.lib.

CX509DSASignature - класс добавляет блокирование к назначенной схеме кодирования, оставляя суперклассы независимыми. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл x509keys.h и библиотечный файл x500.lib.

CX509ExtendedKeyUsageExt - класс с расширением сертификата для дополнительного использования ключей. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл x509certext.h и библиотечный файл x509.lib.

CX50 9ExtensionBase - внутренний класс, не предназначен для общего использования.

CX50 9GeneralName - класс содержит тэг и определяет имя от него зависящее. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл x509gn.h и библиотечный файл x509.lib.

CX509IPAddress - класс с одним IP-адресом. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл x509gn.h и библиотечный файл x509.lib.

CX50 9IPBasedURI - класс для извлечения URI. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл x509gn.h и библиотечный файл x509.lib.

CX509IPSubnetMask - класс с 8-битной маской IP-подсети. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл x509gn.h и библиотечный файл x509.lib.

CX509KeyUsageExt - класс с расширением сертификата для дополнительного использования ключей. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл x509certext.h и библиотечный файл x509.lib.

CX509NameConstraintsExt - класс с расширением сертификата, который определяет ограничения для названия объекта. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл x509certext.h и библиотечный файл x509.lib.

CX509RFC822Name - класс с адресом электронной почты. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл x509gn.h и библиотечный файл x509.lib.

CX50 9RFC822NameSubtree - класс с полным или частичным адресом электронной почты. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл x509gn.h и библиотечный файл x509.lib.

CX50 9RSAPublicKey - класс, добавляет блокирование к назначенной схеме кодирования, оставляя суперклассы независимыми. Доступен от версии Symbi-

an OS 6.0. Для работы необходимо подключить заголовочный файл x509keys.h и библиотечный файл x500.lib.

CX509SigningAlgorithmIdentifier - класс, содержит идентификаторы для подписи сертификата. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл x509cert.h и библиотечный файл x509.НБ.

CX509SubjectKeyIdExt - класс с расширением сертификата свидетельства для идентификации по ключам или уникальному идентификатору. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл x509certext.h и библиотечный файл x509.lib.

CX509ValidityPeriod — класс с периодом времени, в течение которого сертификат является годным. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл x509cert.h и библиотечный файл x509.lib.

CX520AttributeTypeAndValue - класс, содержащий тип атрибута и значение по X.520 стандарту. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл x520ava.h и библиотечный файл x500.lib.

TAlgorithmId - перечисляемый тип с перечислениями криптографического алфавита. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл signed.h.

TCertManClientInfo - класс, содержащий уникальный идентификатор и имя. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл certstore.h и библиотечный файл certstore.lib.

TCertStoreCapabilities - класс с набором флажков для установления возможностей памяти. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл certstore.h и библиотечный файл certstore.lib.

TCertStoreInfo - класс, содержащий объект с именем файла памяти и адресом. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл certstore.h и библиотечный файл certstore.lib.

TCertType - перечисляемый тип для определения используемого сертификата. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл certstore.h.

TKeyUsageVals - перечисляемый тип со списком значений для определения ключа. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл securitydefs.h.

TValidationStatus - класс, проверяющий правильность сертификатов. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл x509certchain.h и библиотечный файл x500.НБ.

TValidationError - перечисляемый тип, определяющий ошибку, возникающую при проверке правильности сертификатов. Для работы необходимо подключить заголовочный файл x509certchain.h.

TX509PolicyConstraint - класс, определяющий применение ограничения политики. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл x509certext.h и библиотечный файл x509.НБ.

TGNType - тип с общим типом имени. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл x509gn.h.

## 1.27. Character Conversion

CCnvCharacterSetConverter - класс, преобразовывающий текст между уникадом и другими наборами символов. Доступен от версии Symbian OS 5.1. Для работы необходимо подключить заголовочный файл charconv.h и библиотечный файл charconv.lib.

CCnvCharacterSetNames - класс, содержащий список имен кодирования символов, которые не принадлежат к уникаду и поддерживаются телефоном. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл convnames.h и библиотечный файл convnames.lib.

CnvUtf Converter - класс, конвертирующий текст между уникадом и форматами UTF-7 и UTF-8. Доступен от версии Symbian OS 5.1. Для работы необходимо подключить заголовочный файл utf.h и библиотечный файл charconv.lib.

## 1.28. Character Conversion

### Plug-In Provider

CnvUtilities - класс, обеспечивающий утилиты для сложного кодирования. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл convutils.h и библиотечный файл convutils.lib.

ConvertFromUnicode () - функция для конвертации из уникада в иностранный набор символов. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл convplug.h.

ConvertToUnicode () - функция для конвертации из уникада в иностранный набор символов. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл convplug.h.

Globals - глобальный объект для сменного интерфейса. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл conv-generatedcpp.h.

IsInThisCharacterSetL () - функция для определения кодировки текста. Доступна от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл convplug.h.

ReplacementForUnconvertibleUnicodeCharacters() - функция для возвращения символа по умолчанию, который заменяет неконвертируемые символы. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл convplug.h.

## 1.29. Character Representation of Real Numbers

TRealFormat - класс, выполняющий символьное представление числа: TReal или TRealX. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

### **1.30. Circular Buffers**

CCirBuf Base - базовый класс для циркулярных буферов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32base.h и библиотечный файл euser.lib.

CCirBuf - класс циркулярного буфера, содержащий объекты, определяемые шаблоном. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32base.h и библиотечный файл euser.lib.

CCirBuf fer - класс циркулярного буфера целых чисел от -128 до +127. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32base.h и библиотечный файл euser.lib.

### **1.31. Client/Server**

CServer - абстрактный класс для серверов, обрабатывающих серверные сеансы. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32base.h и библиотечный файл euser.lib.

CSession - класс для представления канала связи между клиентом и сервером. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32base.h и библиотечный файл euser.lib.

CSharableSession - абстрактный класс для всех сеансов. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл e32base.h и библиотечный файл euser.lib.

RMessage — класс, формирует запрос клиента. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

RSessionBase — класс для обеспечения интерфейса. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

RSubSessionBase - класс для обеспечения интерфейса и открытия-закрытия подсеанса. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

TFindServer - класс для нахождения серверов с именами, совпадающими с текущим образом. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

### **1.32. Clipboard**

CClipboard - класс для сохранения данных в буфере обмена. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл baclip.h и библиотечный файл bafl.lib.

### **1.33. Clock**

RAnalogClock - класс, аналоговые часы. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл clock.h и библиотечный файл clock.lib.

RAnimWithUtils - сервисный класс для поддержки анимации часов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл clock.h и библиотечный файл clock.lib.

RClock- класс, устанавливающий время для часов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл clock.h и библиотечный файл clock.lib.

RDigitalClock - класс, цифровые часы. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл clock.h и библиотечный файл clock.lib.

RTimeDevice - класс, устанавливает параметры дисплея для часов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл clock.h и библиотечный файл clock.lib.

SAnalogDisplayAmPm - структура, определяет время AM или PM для аналоговых часов. Доступна от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл clock.h и библиотечный файл clock.lib.

STimeDeviceShadow - структура, добавляет тень к дисплею часов. Доступна от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл clock.h и библиотечный файл clock.lib.

TAnalogDisplayHand - класс, стрелка для аналоговых часов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл clock.h и библиотечный файл clock.lib.

TAnalogDisplayHandType - перечисляемый тип, разные виды стрелки аналоговых часов. Для работы необходимо подключить заголовочный файл clock.h.

TDigitalDisplayHorizontalTextAlignment - перечисляемый тип для горизонтального выравнивания текста в дисплее цифровых часов. Для работы необходимо подключить заголовочный файл clock.h.

TDigitalDisplayLayoutChar - перечисляемый тип, содержащий специальные символы, которые могут применяться в цифровом дисплее часов. Для работы необходимо подключить заголовочный файл clock.h.

TDigitalDisplayTextSection - класс, текстовая область для цифровых часов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл clock.h и библиотечный файл clock.lib.

TDigitalDisplayVerticalTextAlignment - перечисляемый тип для вертикального выравнивания текста в дисплее цифровых часов. Для работы необходимо подключить заголовочный файл clock.h.

TDisplayAddition - внутренний класс, не предназначен для общего использования.

### **1.34. Command Line Parsing**

CCommandLineArguments - внутренний класс, не предназначен для общего использования.

### 1.35. CommDb

CCommDbOverrideSettings - класс параметров настройки столбцов в таблице. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл cdbover.h и библиотечный файл commdb.lib.

CCommsDatabase - класс для обращения к базе данных связи. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл commdb.h и библиотечный файл commdb.lib.

CCommsDbConnectionPref TableView - класс для привилегированного обращения к базе данных связи. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл cdbpreftable.h и библиотечный файл commdb.lib.

CCommsDbTableView - класс осуществляет действия с таблицами и отчетами базы данных. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл commdb.h и библиотечный файл commdb.lib.

CCommsDbTemplateRecord - класс, содержащий функции чтения и записи в столбцы. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл cdbtemp.h и библиотечный файл commdb.lib.

CStoreableOverrideSettings - класс, формирующий параметры настройки для столбцов таблиц и потоков для передачи в буфер. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл cdbstore.h и библиотечный файл commdb.lib.

TCommDbBearer - перечисляемый тип для поддержки каналов. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл cdbcols.h.

TCommDbConnectionDirection - перечисляемый тип направления связи. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл cdbcols.h.

TCommDbDatabaseType - перечисляемый тип базы данных. Доступен от версии Symbian OS 6.1 по 7.0. Для работы необходимо подключить заголовочный файл cdbcols.h.

TCommDbDialogPref - перечисляемый тип для выполнения запроса во время соединения. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл cdbcols.h.

TCommDbGprsClassCBearer - перечисляемый тип для глобальных настроек. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл cdbcols.h.

TCommDbOpeningMethod - перечисляемый тип для вызова CCommsDatabase::NewL(). Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл cdbcols.h.

TCommsDbBearerType - перечисляемый тип, используемый DIAL\_\*\_ISP:ISP\_BEARER\_TYPE. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл cdbcols.h.

TCommsDbIspType - перечисляемый тип, который применяется DIAL\_\*\_ISP:ISP\_TYPE, OUTGOING WCDMArGPRS AP TYPE, и CDMA2000 PACKET

SERVICE\_TABLE: CDMA\_AP\_TYPE. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл cdbcols.h.

TCommsDbModemSpeakerSetting - перечисляемый тип с настройками для динамика модема. Для работы необходимо подключить заголовочный файл cdbcols.h.

TCommsDbModemSpeakerVolume - перечисляемый тип для настроек громкости динамика модема. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл cdbcols.h.

TCommsDbWapIspType — перечисляемый тип, значения WAP\_ISP\_TYPE. Доступен от версии Symbian OS 6.0. по 7.0. Для работы необходимо подключить заголовочный файл cdbcols.h.

TCommsDbWapWspOption - перечисляемый тип с опциями WAP WSP. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл cdbcols.h.

### **1.36. Connection Management**

RConnection — класс, интерфейс для управления сетевыми подключениями. Доступен от версии Symbian OS 7.0s. Для работы необходимо подключить заголовочный файл Es\_sock.h и библиотечный файл ESocket.lib.

RNif - класс, обеспечивающий расширенный интерфейс для контроля связи. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл nifman.h и библиотечный файл nifman.lib.

RGenericAgent - класс для отслеживания ошибок при подключении и дозвона по сети. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл AgentClient.h и библиотечный файл GenConn.lib.

TNif AgentInfo - класс, содержит информацию, описывающую агента. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл nifman.h.

TNif Progress - класс содержит информацию о модемной связи. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл nifman.h.

SAGENTSMBase - внутренний класс, не предназначен для общего использования.

SAGENTSTATEBase - внутренний класс, не предназначен для общего использования.

SCommsDbAccess - внутренний класс, не предназначен для общего использования.

CDefaultRecordAccess - внутренний класс, не предназначен для общего использования.

MAGENTOBSERVER - внутренний класс, не предназначен для общего использования.

MAGENTSTATEMACHINEEnv - внутренний класс, не предназначен для общего использования.



MServiceChangeObserver - внутренний класс, не предназначен для общего использования.

RDialogNotifier - внутренний класс, не предназначен для общего использования.

TAuthenticationPair - внутренний класс, не предназначен для общего использования.

TConnectionPrefs — внутренний класс, не предназначен для общего использования.

TConnectionSettings - внутренний класс, не предназначен для общего использования.

TispConnectionNames - внутренний класс, не предназначен для общего использования.

TNewIapConnectionPrefs - внутренний класс, не предназначен для общего использования.

TPctResponse — внутренний класс, не предназначен для общего использования.

### **1.37. Contacts Model**

CCntFilter - класс для фильтрации базы данных. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл cntfilt.h и библиотечный файл cntmodel.lib.

CContactAgentField - класс для хранения поля идентификатора. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл cntfilt.h и библиотечный файл cntmodel.lib.

CContactCard - класс для создания контактной карты. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл cntitem.h и библиотечный файл cntmodel.lib.

CContactCardTemplate - класс, обеспечивающий поддержку шаблонов карточек контактов. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл cntitem.h и библиотечный файл cntmodel.lib.

CContactChangeNotifier — класс, получает события и уведомляет об изменениях в базе данных. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл cntdb.h и библиотечный файл cntmodel.lib.

CContactDatabase - класс базы данных контактов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл cntdb.h и библиотечный файл cntmodel.lib.

CContactDateField - класс, устанавливает и сохраняет дату и время. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл cntfldst.h и библиотечный файл cntmodel.lib.

CContactFieldStorage - абстрактный класс для хранения разных видов полей контактов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл cntfldst.h и библиотечный файл cntmodel.lib.

`CContactGroup` - класс групп контактов. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `cntitem.h` и библиотечный файл `cntmodel.lib`.

`CContactEntry` - класс, контактная ICC-запись. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `Cntitem.h` и библиотечный файл `cntmodel.lib`.

`CContactArray` - класс массива элементов контактов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `cntdef.h` и библиотечный файл `cntmodel.lib`.

`CContactItem` - абстрактный класс для карточек контактов, шаблонов и групп. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `cntitem.h` и библиотечный файл `cntmodel.lib`.

`CContactItemField` - класс поля для сохранения данных контакта. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `cntfield.h` и библиотечный файл `cntmodel.lib`.

`CContactItemFieldDef` - класс для определения полей. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `cntfield.h` и библиотечный файл `cntmodel.lib`.

`CContactItemFieldSet` - класс массива полей контактов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `cntitem.h` и библиотечный файл `cntmodel.lib`.

`CContactItemPlusGroup` - абстрактный класс для избежания дублирования в классах `CContactGroup`, `CContactCard` и `CContactOwnCard`. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `cntitem.h` и библиотечный файл `cntmodel.lib`.

`CContactItemViewDef` - класс, определяющий вид представления контактных элементов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `cntdb.h` и библиотечный файл `cntmodel.lib`.

`CContactNumberField` — класс, представляющий целочисленное значение, сохраняемое в поле контакта. В настоящее время ST0T класс не поддерживается. Доступен в версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `cntfldst.h` и библиотечный файл `cntmodel.lib`.

`CContactOwnCard` - класс карты с информацией о владельце (визитная карточка). Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `cntitem.h` и библиотечный файл `cntmodel.lib`.

`CContactStoreField` - класс для хранения и установки бинарных полей. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `cntfldst.h` и библиотечный файл `cntmodel.lib`.

`CContactTemplate` - класс, шаблон контактов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `cntitem.h` и библиотечный файл `cntmodel.lib`.

`CContactTextDef` - класс для определения пути к элементу поля для занесения его в контактную строку. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `cntdb.h` и библиотечный файл `cntmodel.lib`.

`CContaktTextField` - класс для сохранения текста в поле контакта. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `cntfldst.h` и библиотечный файл `cntmodel.lib`.

`CContaktViewDef` — класс, задающий вид отображения контакта. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `cntdb.h` и библиотечный файл `cntmodel.lib`.

`CContentTpe` - класс, тип полей контактов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `cntfield.h` и библиотечный файл `cntmodel.lib`.

`CIdleFinder` — класс для поиска и получения результата поиска в базе данных. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `cntdb.h` и библиотечный файл `cntmodel.lib`.

`ContactGuid` - класс для запроса уникального идентификатора поля контакта. Доступен от версии Symbian OS 5.1. Для работы необходимо подключить заголовочный файл `cntitem.h` и библиотечный файл `cntmodel.lib`.

`MContactDbObserver` - класс, определяющий протокол обозревателя для обработки изменений в базе данных. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `cntdbobs.h`.

`MContactDbPrivObserver` — внутренний класс, не предназначен для общего использования.

`MIdleFindObserver` - класс, определяющий протокол для нахождения обозревателя. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `cntdb.h`.

`SFindlnTextDefWordParser` - структура для синтаксического поиска. Для работы необходимо подключить заголовочный файл `cntdb.h`.

`TContactDbObserverEvent` - структура, содержащая тип изменения в базе данных контактов и идентификатор измененного элемента. Доступна от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `cntdbobs.h`.

`TContactDbObserverEventType` - перечисляемый тип с видами возможных изменений. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `cntdbobs.h`.

`TContactItemId` - тип, идентификатор контакта. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `cntdef.h`.

`TContactIter` - класс для поиска контакта в базе данных. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `cntdb.h` и библиотечный файл `cntmodel.lib`.

`TContactSyncId`- тип, идентификатор устройства с которым была синхронизирована база данных. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `cntdef.h`.

`TContactTextDefItem` — класс, содержит тип поля и строку длиной в 4 символа. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `cntdb.h` и библиотечный файл `cntmodel.lib`.

TFieldType - тип, универсальный идентификатор, который идентифицирует вид поля контакта. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл cntdef.h.

TStorageType - тип памяти для хранения числа, идентифицирующего данные в поле контакта. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл cntdef.h.

### **1.38. Contact Views**

CContactConcatenatedView - класс, упорядочивает контакты в единое представление. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл cntview.h и библиотечный файл cntview.lib.

CContactFilteredView - класс, обеспечивает фильтрованное представление контактов. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл cntview.h и библиотечный файл cntview.lib.

CContactFindView - класс, обеспечивает представление тех контактов, которые соответствуют критериям поиска. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл cntview.h и библиотечный файл cntview.lib.

CContactGroupView - класс для группового представления контактов. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл cntview.h и библиотечный файл cntview.lib.

CContactLocalView - базовый класс для представления контактов. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл cntview.h и библиотечный файл cntview.lib.

CContactNamedRemoteView - класс, предоставляет общий доступ к контактам. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл cntviewbase.h и библиотечный файл cntmodel.lib.

CContactRemoteView - класс, предоставляет мгновенный общий доступ к контактам. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл cntviewbase.h и библиотечный файл cntmodel.lib.

CContactRemoteViewBase - базовый класс для всех классов предоставления доступа. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл cntviewbase.h и библиотечный файл cntmodel.lib.

CContactSubView - класс для представления контактов в заданном диапазоне. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл cntview.h и библиотечный файл cntview.lib.

CContactViewBase - базовый класс для всех классов представления вида контактов. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл cntviewbase.h и библиотечный файл cntview.lib.

CContactViewFindConf igInterf ace - класс интерфейса для конфигурирования пути согласования слов. Доступен от версии Symbian OS 7.0s. Для работы необходимо подключить заголовочный файл cntviewfindconfig.h.

CContactViewHighRange - класс для представления в низком sub-диапазоне. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл cntview.h и библиотечный файл cntview.lib.

CContactViewLowRange - класс для представления в высшем sub-диапазоне. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл cntview.h и библиотечный файл cntview.lib.

CContactViewRange - класс для представления в высшем и низшем sub-диапазоне. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл cntview.h и библиотечный файл cntview.lib.

CContactviewRangeBase - абстрактный класс для всех классов, представляющих контакты в заданных диапазонах. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл cntview.h и библиотечный файл cntview.lib.

CViewContact - класс, содержащий информацию о контакте для представления в обзоре контактов. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл cntviewbase.h и библиотечный файл cntmodel.lib.

MContactViewObserver - класс, интерфейс для отображения контактов. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл cntviewbase.h.

RContactRemoteView - класс, задающий порядок сортировки для контактов. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл cntviewbase.h и библиотечный файл cntmodel.lib.

RContactViewSortOrder - класс, задающий порядок сортировки для контактов. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл cntviewbase.h и библиотечный файл cntmodel.lib.

TContactIdWithMapping - структура, применяемая для фильтрации и группового представления, соотносящая идентификатор и индекс для основного представления. Доступна от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл cntviewbase.h.

TContactViewEvent - класс, идентифицирующий представление контакта. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл cntviewbase.h.

TContactViewPreferences - перечисляемый тип с настройками для сортирования контактов. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл cntviewbase.h.

### **1.39. Converter Architecture**

CSnaConverter - класс, свойства конвертера. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл concnf.h и библиотечный файл conarc.lib.

CSnaConverterList - класс, обеспечивающий список доступных конвертеров. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл conlist.h и библиотечный файл conarc.lib.

CSnaConvInf oFile - внутренний класс, не предназначен для общего использования.

CSnaConvInf oFileReader - класс, формирующий свойства конвертера. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл concnf.h и библиотечный файл conarc.lib.

CSnaConvInf oFileWriter - класс, записывающий информационный файл конвертера. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл concnf.h и библиотечный файл conarc.lib.

CConverterBase - класс интерфейса для конвертирования форматов данных. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл conarc.h и библиотечный файл conarc.lib.

CConverterLibrary - класс для конвертирования DLL-библиотек. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл conarc.h и библиотечный файл conarc.lib.

CMimeInf o — класс, содержащий локализованные имена типов данных. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл concnf.h и библиотечный файл conarc.lib.

CNF\_FILE - структура ресурсов, описывающая конвертер DLL. Доступна от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл cnftool.rh.

CONVERTER\_DATA - структура ресурсов, описывающая конвертер DLL. Доступна от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл cnftool.rh.

LANG\_DATA- структура ресурсов, имя для типа MIME. Доступна от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл cnftool.rh.

MConverterUiObserver - класс, интерфейс конвертации. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл conarc.h и библиотечный файл conarc.lib.

MIME - структура ресурсов, описывающая тип данных для конвертации. Доступна от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл cnftool.rh.

SConverterInf o - структура, описывающая конвертер. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл conlist.h и библиотечный файл conarc.lib.

TLanguage Inf o - класс, содержащий локализованные имена для типов данных. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл concnf.h и библиотечный файл conarc.lib.

TTranslation- тип, буфер для содержания локализованных имен данных. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл concnf.h.

## **1.40. Cookies Support**

C COOKIE- класс, формирующий атрибуты Cookie. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл scookie.h.

CCookieFilterInterface - класс интерфейса ECom для определения HTTP Cookie-фильтра. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл CCookieFilterInterface.h.

HTTPCookiePanic - класс для определения паники, связанной с Cookie. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл httpcookieerr.h.

MHTTPCookieManager - класс абстрактного интерфейса менеджера Cookie. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл mhttpcookiemanager.h.

### **1.41. Critical Sections**

RCriticalSection - класс, дескриптор на критический раздел. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

### **1.42. Data Application Model**

CDaActiveIncremental — класс, формирующий в активном объекте модель данных по возрастающей. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл damainc.h и библиотечный файл damodl.lib.

CDaColMap - тип, массив объектов, которые ориентируют ресурсы к столбцам. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл dammerge.h.

CDaColNameArray — тип, массив имен столбца. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл damdbdes.h.

CDaDbIncremental — класс, обеспечивающий модель данных по возрастающей. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл damainc.h и библиотечный файл damodl.lib.

CDaExporter - класс, экспортирует данные приложения в текстовый формат. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл damimpor.h и библиотечный файл damodl.lib.

CDaFileDelimitedImporter - класс, импортирующий текстовый файл с заданным разделителем в модель данных приложения. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл damimpor.h и библиотечный файл damodl.lib.

CDaFileImporter - базовый класс для импортирования файла в модель данных приложения. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл damimpor.h и библиотечный файл damodl.lib.

CDaFileSeparatedImporter - класс, импортирующий текстовый файл в модель данных приложения с заданным столбцом и разделителем. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл damimpor.h и библиотечный файл damodl.lib.

CDaFilterArray - тип, массив полей фильтрации. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл damfind.h.

CDaIncremental - базовый класс для классов, осуществляющих возрастающие операции. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл damainc.h и библиотечный файл damodl.lib.

CDaMerger - класс, утилита для объединения двух моделей данных приложения. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл dammerge.h и библиотечный файл damodl.lib.

CDaModel - класс для управления базой данных приложения. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл damdbms.h и библиотечный файл damodl.lib.

CDaRow - класс, строка в модели данных приложения. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл damform.h и библиотечный файл damodl.lib.

CDaSortArray - тип с массивом полей сортировки. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл damfind.h.

CDaSqlParams - класс, формирующий параметры для поиска, фильтрации и сортировки данных приложения. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл damfind.h и библиотечный файл damodl.lib.

CDaStoreReclaim - класс, обеспечивающий уплотнение и восстановление пространства в файле по возрастающей. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл damainc.h и библиотечный файл damodl.lib.

CDaUndoStack - класс, позволяющий удалять модель данных приложения. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл damdelet.h и библиотечный файл damodl.lib.

CDaUserColSet - класс с набором столбцов, подлежащих для отображения пользователю. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл damdbdes.h и библиотечный файл damodl.lib.

CDaUserDbDesc - класс, помощник базы данных. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл damdbdes.h и библиотечный файл damodl.lib.

KUidDataApp - UID-данные приложения. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл damdef.h.

MDaIncremental — абстрактный класс для классов, осуществляющих возрастающие операции. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл damainc.h и библиотечный файл damodl.lib.

MDaIncrementalObserver — класс, обозреватель для возрастающей операции. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл damainc.h и библиотечный файл damodl.lib.

MDaObserver - класс, обозреватель базы данных приложения. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл damobsrv.h и библиотечный файл damodl.lib.



TDaColMap - класс, направляющий ресурс к столбцу. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл dammerge.h и библиотечный файл damodl.lib.

TDaColMapFactory - класс, формирующий карты столбцов в разные модели данных приложения. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл dammerge.h и библиотечный файл damodl.lib.

TDaFileImporterFactory - класс для создания конвертировщиков файлов в модель данных приложения. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл damimport.h и библиотечный файл damodl.lib.

TDaFileImportParams - класс, содержащий параметры для импорта текстового файла. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл damimport.h и библиотечный файл damodl.lib.

TDaFilter - класс с параметрами для фильтрации поля. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл damfind.h и библиотечный файл damodl.lib.

TDaSort - класс с параметрами для сортирования в поле. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл damfind.h и библиотечный файл damodl.lib.

TDaStoreResolver - класс, осуществляющий запрос для определения памяти приложения. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл damform.h и библиотечный файл damodl.lib.

TDatabaseName - тип, имя буфера базы данных приложения. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл damdef.h.

TDaUserCol - класс, пользовательская спецификация столбца. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл damdbdes.h и библиотечный файл damodl.lib.

TFindString - тип для поиска строки в базе данных приложения. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл damdef.h.

### **1.43. Date and Time Handling**

TDateTime - класс, устанавливает дату и время и предоставляет пользователю. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

TFirstWeekRule - перечисляемый тип для определения первой недели года. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

Time - класс, члены которого могут использоваться другими классами даты и времени. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

TLongDateFormatSpec - класс, содержащий список команд, которые определяют длинный (полный) формат даты. Доступен от версии Symbian OS 6.0.

Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

TShortDateFormatSpec - класс, содержащий список команд, которые определяют короткий формат даты. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

TTime - класс для сохранения и управления датой и временем. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

TTimeFormatSpec - класс, содержащий команды для форматирования строки времени. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

TTimeIntervalBase - базовый класс для классов времени, использующих 32-битное представление. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

TTimeIntervalDays - класс для представления времени в формате дней. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

TTimeIntervalHours - класс для представления времени в формате часов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

TTimeIntervalMicroSeconds32 - класс для представления микросекунд в 32-битном формате. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

TTimeIntervalMicroSeconds — класс, сохраняет миллионную часть секунды как целое 64-битное число. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

TTimeIntervalMinutes - класс для представления времени в формате минут. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

TTimeIntervalMonths - класс для представления времени в формате месяцев. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

TTimeIntervalSeconds - класс для представления времени в формате секунд. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

TTimeIntervalYears - класс для представления времени в формате лет. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

#### **1.44. DBMS Columns, Column Sets and Keys**

CDbColSet - внутренний класс, не предназначен для общего использования.

CDbKey - внутренний класс, не предназначен для общего использования.

TDbCol - класс для определения столбца в таблице. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл d32dbms.h и библиотечный файл edbms.lib.

TDbColNameC - тип, представляющий неизменяемое DBMS-имя столбца. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл d32dbms.h.

TDbColName - тип, представляющий изменяемое DBMS-имя столбца. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл d32dbms.h.

TDbColNo - тип, выполняет идентификацию столбца в наборе столбцов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл d32dbms.h.

TDbColSetIter — класс для итерации по содержанию набора столбцов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл d32dbms.h и библиотечный файл edbms.lib.

TDbColType - перечисляемый тип, представляет поддерживаемый тип столбцов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл d32dbms.h.

TDbKeyCol - класс для определения столбца в индексе. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл d32dbms.h и библиотечный файл edbms.lib.

TDbNameC - тип, представляющий неизменяемое DBMS-имя. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл d32dbms.h.

TDbName - тип, представляющий изменяемое DBMS-имя. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл d32dbms.h.

TDbTextComparison — перечисляемый тип для определения различных путей сравнения столбцов Text и LongText. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл d32dbms.h.

#### **1.45. DBMS Database Incremental Operations**

RDbIncremental - класс, интерфейс для выполнения возрастающей длинной операции на базе данных. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл d32dbms.h и библиотечный файл edbms.lib.

RDbUpdate - класс, интерфейс для возрастающего выполнения DML-инструкции. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл d32dbms.h и библиотечный файл edbms.lib.

#### **1.46. Interface to DBMS Databases**

CDblIndexNames — возвращающий тип для списка всех имен в таблице. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл d32dbms.h.

CDbNames - внутренний класс, не предназначен для общего использования.

CDbTableNames - возвращающий тип для таблицы имен базы данных. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл d32dbms.h.

RDbDatabase - абстрактный класс функциональных возможностей базы данных. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл d32dbms.h и библиотечный файл edbms.lib.

RDbNamedDatabase - класс, интерфейс для создания и открытия базы данных. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл d32dbms.h и библиотечный файл edbms.lib.

RDbStoreDatabase - класс, обеспечивающий выполнение API DBMS для представления базы данных. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл d32dbms.h и библиотечный файл edbms.lib.

#### **1.47. DBMS Rowsets/**

RDbColReadStream- класс для чтения значений столбца как потоковые данные. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл d32dbms.h и библиотечный файл edbms.lib.

RDbColWriteStream - класс, создает длинные столбцы, когда происходит вставка или редактирование строк в наборе. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл d32dbms.h и библиотечный файл edbms.lib.

RDbRowConstraint - класс, представляет пред-компиляционный SQL-поиск для проверки строки в наборе. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл d32dbms.h и библиотечный файл edbms.lib.

RDbRowSet - внутренний класс, не предназначен для общего использования.

RDbTable - класс для операций с именем таблицы в базе данных. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл d32dbms.h и библиотечный файл edbms.lib.

RDbView - класс для создания наборов строк из SQL. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл d32dbms.h и библиотечный файл edbms.lib.

TDbBookmark - класс для хранения текущего размещения набора строк. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл d32dbms.h.

TDbLookupKey - внутренний класс, не предназначен для общего использования.

TDbQuery - класс SQL для режима сравнения текста. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл d32dbms.h.

TDbSeekKey- класс ключей базы данных. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл d32dbms.h и библиотечный файл edbms.lib.

TDbSeekMultiKey - класс ключей базы данных. Доступен от версии Symbi-an OS 5.0. Для работы необходимо подключить заголовочный файл d32dbms.h и библиотечный файл edbms.lib.

TDbWindow - класс для описания формы окна. Доступен от версии Symbi-an OS 5.0. Для работы необходимо подключить заголовочный файл d32dbms.h.

TUnion - внутренний класс, не предназначен для общего использования.

### **1.48. DBMS Sharing Databases**

RDbHandleBase - внутренний класс, не предназначен для общего использования.

RDbHandle - внутренний класс, не предназначен для общего использования.

RDbNotifier - класс для уведомления об изменениях в базе данных. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл d32dbms.h и библиотечный файл edbms.lib.

RDBs - класс для создания сессии с DBMS-сервером. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл d32dbms.h и библиотечный файл edbms.lib.

### **1.49. Descriptor Arrays**

CDesCl6Array - абстрактный базовый класс для дескрипторных 16-битных массивов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл badescah и библиотечный файл bafl.lib.

CDesCl6ArrayFlat — класс, 16-битный массив для использования единого буфера. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл badescah и библиотечный файл bafl.lib.

CDesCl6ArraySeg - класс, 16-битный массив для использования сегментного буфера. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл badescah и библиотечный файл bafl.lib.

CDesC8Array - класс для 8-битных дескрипторных массивов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл badescah и библиотечный файл bafl.lib.

CDesC8ArrayFlat - класс, 8-битный массив для использования единого буфера. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл badescah и библиотечный файл bafl.lib.

CDesC8ArraySeg - класс, 8-битный массив для использования сегментного буфера. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл badescah и библиотечный файл bafl.lib.

CDesCArrayFlat - тип для формирования массива дескрипторов, чтобы использовать буфер. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл badescah.

CDesCArraySeg - тип для формирования массива дескрипторов, чтобы использовать сегментный буфер. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл badescah.

CDesCArray - тип для формирования массива дескрипторов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл badescah.

CPtrC16Array - класс для массива неизменяемых 16-битных дескрипторов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл badescah и библиотечный файл bafl.lib.

CPtrC8Array - класс для массива неизменяемых 8-битных дескрипторов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл badescah и библиотечный файл bafl.lib.

CPtrCArray - тип для создания неизменяемых дескрипторов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл badescah.

MDesC16Array - интерфейсный класс для массивов 16-битных дескрипторов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл bamdescah.

MDesC8Array - интерфейсный класс для массивов 8-битных дескрипторов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл bamdescah.

MDesCArray - класс для создания независимого класса интерфейса для массива дескрипторов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл bamdescah.

## 1.50. Descriptors

HBufC16 - класс, 16-битный дескриптор динамической памяти. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32des16.h и библиотечные файлы euser.lib, estor.lib.

HBufC8 - класс, 8-битный дескриптор динамической памяти. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32des8.h и библиотечные файлы euser.lib, estor.lib.

HBufC - тип, независимый дескриптор динамической памяти. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

TAlign - перечисляемый тип для операций над данными, которые скопированы или форматированы в дескрипторе. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

TBuf16 - класс, изменяемый 16-битный дескриптор буфера. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32des16.h и библиотечный файл euser.lib.

TBuf8 - класс для выделения буфера заданной длины. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32des8.h и библиотечный файл euser.lib.

TBufBase8 - внутренний класс, не предназначен для общего использования.

TBufBase16 - внутренний класс, не предназначен для общего использования.

TBuf CBase8 - внутренний класс, не предназначен для общего использования.

TBuf CBase16 - внутренний класс, не предназначен для общего использования.

TBuf C16 - класс, неизменяемый 16-битный дескриптор буфера. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32des16.h и библиотечные файлы euser.lib.

TBufC8 - класс, неизменяемый 8-битный дескриптор буфера. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32des8.h и библиотечный файл euser.lib.

TBuf C - класс, независимый неизменяемый дескриптор буфера. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

TBuf - класс, независимый изменяемый дескриптор буфера. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

TDes16 - абстрактный класс, изменяемый 16-битный дескриптор. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32des16.h и библиотечный файл euser.lib.

TDes16overflow - класс, интерфейс для 16-битного дескриптора. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32des16.h и библиотечный файл euser.lib.

TDes8 - абстрактный класс для изменяемых 8-битных дескрипторов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32des8.h и библиотечный файл euser.lib.

TDes8overflow - класс, интерфейс для 8-битного дескриптора. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32des8.h и библиотечный файл euser.lib.

TDesC16 - абстрактный класс для 16-битных дескрипторов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32des16.h и библиотечный файл euser.lib.

TDesC8 - абстрактный класс для неизменяемых 8-битных дескрипторов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32des8.h и библиотечный файл euser.lib.

TDesC - тип для независимого неизменяемого дескриптора. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

TDes - тип для независимого изменяемого дескриптора. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

TDesOverflow - тип, независимый дескриптор. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

TPtr16 - класс, изменяемый 16-битный дескриптор. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32des16.h и библиотечный файл euser.lib.

TPtr8 - класс для реализации и создания дескриптора на 8-битные данные. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32des8.h и библиотечный файл euser.lib.

TPtrCl 6 - класс, неизменяемый 16-битный дескриптор. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32des16.h и библиотечный файл euser.lib.

TPtrC8 - класс, неизменяемый 8-битный дескриптор. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32des8.h и библиотечный файл euser.lib.

TPtrC - тип для независимого неизменяемого дескриптора. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

TPtr - тип для независимого изменяемого дескриптора. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

TRadix - перечисляемый тип, определяющий систему исчисления. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

## 1.51. Device drivers

DLogicalChannel - класс, логический канал ядра. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл k32std.h и библиотечный файл ekern.lib.

DLogicalDevice - класс для LDD-объекта. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл k32std.h и библиотечный файл ekern.lib.

DPhysicalDevice - класс для PDD-объекта. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл k32std.h и библиотечный файл ekern.lib.

RBusLogicalChannel - класс, обработка логического канала. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

RDevice - класс, дескриптор к LDD-объекту. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

TFindLogicalChannel - класс для поиска логических каналов путем сравнения имен логических объектов канала. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

TFindLogicalDevice - класс для поиска LDD-объектов путем сравнения имен. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

TFindPhysicalDevice - класс для поиска PDD-объектов путем сравнения имен. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.



## 1.52. Dial

TChargeCard- класс для сохранения записей на сменной карте. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл dial.h и библиотечный файл dial.lib.

TDialLocation - класс, сохраняющий основную конфигурацию телефона. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл dial.h и библиотечный файл dial.lib.

TelephoneNumber - статический класс для синтаксического анализа номеров телефона. Для работы необходимо подключить заголовочный файл dial.h и библиотечный файл dial.lib.

TChargeCardAccount - тип для сохранения в буфере номера карты. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл dial.h.

TChargeCardPin - тип для сохранения в буфере PIN кода. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл dial.h.

TChargeCardRule - тип для сохранения в буфере правил, заданных сетевым оператором для вызова номеров. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл dial.h.

TCityDialCodes - класс, международные и местные префиксы к номеру. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл dial.h.

TCityDialOptions — перечисляемый тип, определяющий запись кода города в строку при наборе номера, если телефон находится в том же городе. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл dial.h.

TDialOutLocalCode - тип для сохранения в буфере номера без кода для местного вызова. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл dial.h.

TDialOutLongDistanceCode - тип для сохранения в буфере номера без кода для междугороднего вызова. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл dial.h.

TDisableCallWaitingCode - тип, сохраняющий в буфере код для отключения режима отложенных звонков. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл dial.h.

TParseMode — перечисляемый тип, определяющий результирующую строку для показа на дисплее или для вызова номера. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл dial.h.

TPhoneNumber - тип для сохранения в буфере номер телефона. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл dial.h.

TPlusFormatDialOptions - перечисляемый тип, определяющий использование символа «+» при наборе номера. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл dial.h.

### **1.53. Dynamically Loading Link Libraries**

RLibrary - внутренний класс, не предназначен для общего использования.

TFindLibrary — класс для нахождения DLL, чьи имена соответствуют установленному образцу. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

TLibraryEntry - тип, определяющий функцию, принимающую параметр TDllReason и возвращающую Tint. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

TLibraryFunction - тип, не определяет никакую функцию, но возвращает Tint. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

### **1.54. Dynamic Arrays**

CArrayFixBase - класс массива объектов фиксированной длины. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32base.h и библиотечный файл euser.lib.

CArrayFix - базовый класс массива объектов фиксированной длины. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32base.h и библиотечный файл euser.lib.

CArrayFixFlat - класс, массив объектов фиксированной длины, содержащихся в динамическом буфере. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32base.h и библиотечный файл euser.lib.

CArrayFixFlat<TAny> - класс, массив объектов (без типов) фиксированной длины, использующих динамический буфер. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32base.h и библиотечный файл euser.lib.

CArrayFixFlat<TInt> - базовый класс для массивов типа TInt, содержащихся в динамическом буфере. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл e32base.h и библиотечный файл euser.lib.

CArrayFixFlat<TUid> - базовый класс для массивов типа TUid, содержащихся в динамическом буфере. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл e32base.h и библиотечный файл euser.lib.

CArrayFixSeg - класс, массив объектов фиксированной длины, содержащихся в сегментном буфере. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32base.h и библиотечный файл euser.lib.

CArrayFixSeg<TAny> - класс, массив объектов (без типов) фиксированной длины, использующих сегментный буфер. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32base.h и библиотечный файл euser.lib.

`CArrayFix<TAny>` - класс, массив объектов (без типов) фиксированной длины. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `e32base.h` и библиотечный файл `euser.lib`.

`CArrayPakBase` - базовый класс для изменения длины, упаковки и массивов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `e32base.h` и библиотечный файл `euser.lib`.

`CArrayPak` - шаблонный базовый класс изменения длины, упаковки и массивов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `e32base.h` и библиотечный файл `euser.lib`.

`CArrayPakFlat` — класс, массив объектов переменной длины, упакованных в буфер. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `e32base.h` и библиотечный файл `euser.lib`.

`CArrayPak<TAny>` — шаблонный специализированный базовый класс изменения длины, упаковки и массивов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `e32base.h` и библиотечный файл `euser.lib`.

`CArrayPtr` — класс для массива указателей на объект. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `e32base.h` и библиотечный файл `euser.lib`.

`CArrayPtrFlat` - класс для массива указателей на объект, использующих буфер. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `e32base.h` и библиотечный файл `euser.lib`.

`CArrayPtrSeg` - класс для массива указателей на объект, использующих сегментный буфер. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `e32base.h` и библиотечный файл `euser.lib`.

`CArrayVarBase` - класс для массивов переменной длины. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `e32base.h` и библиотечный файл `euser.lib`.

`CArrayVar` - шаблонный базовый класс для массивов переменной длины. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `e32base.h` и библиотечный файл `euser.lib`.

`CArrayVarFlat` - класс для массива объектов переменной длины, использующих динамический буфер. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `e32base.h` и библиотечный файл `euser.lib`.

`CArrayVarSeg` — класс для массива объектов переменной длины, использующих сегментный динамический буфер. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `e32base.h` и библиотечный файл `euser.lib`.

`CArrayVar<TAny>` - шаблонный специализированный базовый класс для массивов переменной длины. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `e32base.h` и библиотечный файл `euser.lib`.

`RArrayBase` - внутренний класс, не предназначен для общего использования.

RArray - класс, массив объектов заданной длины. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

RPointerArrayBase - внутренний класс, не предназначен для общего использования.

RArray<TInt> - класс, массив целых чисел со знаком. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

RArray<TUint> - класс, массив целых чисел без знака. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

RPointerArray - класс, массив указателей на объекты. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

TArray - класс универсального массива. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

TIdentityRelation - класс, упаковывающий функцию, которая сравнивает два объекта на соответствие типу класса. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

TLinearOrder - класс, упаковывающий функцию, которая определяет порядок двух объектов класса. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

### **1.55. Dynamic Buffers**

CBuf Base - класс интерфейса для динамических буферов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32base.h и библиотечный файл euser.lib.

CBuf Flat - класс, обеспечивает память динамического буфера. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32base.h и библиотечный файл euser.lib.

CBuf Seg - класс сегментного динамического буфера. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32base.h и библиотечный файл euser.lib.

### **1.56. ECom Plug-in Architecture**

ImplementationInformation - класс контейнера для ECom данных. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл ImplementationInformation.h и библиотечный файл ecom.lib.

CResolver - абстрактный класс для классов, осуществляющих выполнение интерфейса по заданным установкам. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл resolver.h и библиотечный файл ecom.lib.

IMPLEMENTATION\_INFO - структура ресурсов, описывает выполнение интерфейса. Доступна от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл RegistryInfo.rh.

INTERFACE\_INFO - структура ресурсов, объявляет выполнение интерфейса. Доступна от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл RegistryInfo.rh.

MPublicRegistry — класс, обеспечивающий список выполнения интерфейса. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл PublicRegistry.h и библиотечный файл esom.lib.

REComSession - класс для идентификации, реализации и уничтожения интерфейса. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл esom.h и библиотечный файл esom.lib.

REGISTRY\_INFO - структура ресурсов для выполнения интерфейса доступного в коллекции. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл RegistryInfo.h.

RImplInf oArray — тип, массив выполняемых указателей. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл ImplementationInformation.h и библиотечный файл esom.lib.

RImplInfoPtrArray - тип, массив выполняемых указателей. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл ImplementationInformation.h и библиотечный файл esom.lib.

TEComResolverParams - класс с данными для выбора выполнения интерфейса. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл EComResolverParams.h и библиотечный файл esom.lib.

TImplementationProxy - тип, сопоставляющий выполняемый UID с указателем на выполняемую функцию. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл ImplementationProxy.h и библиотечный файл esom.lib.

## **1.57. Embedding**

CPicture — класс для рисования в графическом контексте, сохранения и восстановления изображения. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл gdi.h и библиотечный файл gdi.lib.

MPictureFactory - класс интерфейса для иллюстрации и восстановления класса CPicture. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл gdi.h.

TPictureCapability - класс для масштабирования изображения. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл gdi.h.

TPictureHeader — класс, содержащий заголовок изображения, обеспечивающий интерфейс. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл gdi.h и библиотечный файл gdi.lib.

## 1.58. Encrypted Streams and Stores

CSecureStore — класс памяти с зашифрованными потоками. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл s32crypt.h и библиотечный файл estor.lib.

RDecryptStream — класс, расшифровка потока. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл s32crypt.h и библиотечный файл estor.lib.

REncryptStream- класс, кодирование потока. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл s32crypt.h и библиотечный файл estor.lib.

TDecryptFilter - класс, расшифровывающий фильтр. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл s32crypt.h и библиотечный файл estor.lib.

TEncryptFilter - класс, шифрующий фильтр. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл s32crypt.h и библиотечный файл estor.lib.

TSecureFilter - внутренний класс, не предназначен для общего использования.

## 1.59. Environment Change Notifier

CEnvironmentChangeNotifier - класс для обработки изменений состояния среды. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл basctf.h и библиотечный файл bafl.lib.

## 1.60. ETel Core

RCall - внутренний класс, не предназначен для общего использования.

RLine - внутренний класс, не предназначен для общего использования.

RPhone - внутренний класс, не предназначен для общего использования.

RTelServer— внутренний класс, не предназначен для общего использования.

RTelSubSessionBase - внутренний класс, не предназначен для общего использования.

## 1.61. Extended Notifier Framework

MEikSrvNotifierBase - класс интерфейса для уведомлений сервера. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл eiknotapi.h.

TNotifierCapabilities - перечисляемый тип, флажки для определения возможностей уведомления. Для работы необходимо подключить заголовочный файл eiknotapi.h.

## **1.62. Fax Client**

CFaxTransfer - класс для отправки или приема факса. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл cfax32.h и библиотечный файл faxcli.lib.

RFax - класс, клиентская информация о прогрессе. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл etel.h и библиотечный файл etel.lib.

TFaxBuf SenderId - тип для определения буфера, который содержит ID-отправителя факса. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл faxdefn.h.

TFaxClass - перечисляемый тип, определяющий поддерживаемые факс-модемные классы. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл faxdefn.h.

TFaxCompression - перечисляемый тип, определяющий поддерживаемый тип сжатия. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл faxdefn.h.

TFaxMode - перечисляемый тип, определяющий семь поддерживаемых режимов факсимильного сеанса. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл faxstd.h.

TFaxPhase - перечисляемый тип, определяющий стадии факсимильного сеанса. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл faxdefn.h.

TFaxResolution - перечисляемый тип, определяющий поддерживаемое разрешение. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл faxdefn.h.

TFaxSettings - класс с факсимильной конфигурацией. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл faxset.h и библиотечный файл faxcli.lib.

TRawScanLine - тип, определяет необработанную строку. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл faxset.h и библиотечный файл faxcli.lib.

## **1.63. Fax Client MTM**

CFaxHeader - класс, массив объектов с информацией для адресата факса. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл FXUT.h и библиотечный файл fxcm.lib.

CFaxMtmClient - класс факсимильного интерфейса MTM. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл FAXC.h и библиотечный файл fxcm.lib.

CFaxRecipient - класс, создающий информацию адресата. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл FXUT.h и библиотечный файл fxcm.lib.

KFaxSettingsVersion - идентификация версии класса настроек факса. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл fxut.h.

TFaxCmds - перечисляемый тип со специфическими командами факса. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл FAXCMDS.h.

TFaxProgress — базовый класс для информационных классов прогрессов. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл FXUT.h и библиотечный файл fxcm.lib.

TFaxSessionProgressBuf - тип для упаковки объекта TFaxSession-Progress. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл FXUT.h и библиотечный файл fxcm.lib.

TFaxSessionProgress - класс с информацией о прогрессе асинхронной факсимильной операции. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл FXUT.h и библиотечный файл fxcm.lib.

TMSvFaxEntry - класс с параметрами настройки входящего сообщения. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл FXUT.h и библиотечный файл fxcm.lib.

TMTMFaxSettings - класс с параметрами настройки MTM. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл fxut.h и библиотечный файл fxcm.lib.

#### **1.64. Fax Header Line**

CFaxHeaderLines - класс, содержащий заголовок для чтения и записи данных. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл cfaxiah и библиотечный файл FaxIO.lib.

CFaxT4 - класс для кодирования-декодирования факсимильных строк. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл cfaxiah и библиотечный файл FaxIO.lib.

TFaxHeaderInfo - класс для информации заголовка. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл cfaxiah.

TFaxHeaderInfoPkg - тип для упаковки информации заголовка. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл cfaxiah.

#### **1.65. Fax Store**

CReadFaxFile - класс для открытия факсимильного файла для чтения. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл faxstore.h и библиотечный файл FaxStrm.lib.

CReadFaxPages - класс для чтения страниц факсимильного файла. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл faxstore.h и библиотечный файл FaxStrm.lib.



CWriteFaxFile - класс, создающий и открывающий файл для записи. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл faxstore.h и библиотечный файл FaxStrm.lib.

CWriteFaxPages - класс для добавления страниц к файлу. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл faxstore.h и библиотечный файл FaxStrm.lib.

TFaxPageInfol - класс для хранения факсимильной информации страницы. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл faxstore.h.

### **1.66. File Logging**

FLogger - внутренний класс, не предназначен для общего использования.

RFileLogger - внутренний класс, не предназначен для общего использования.

TFileLoggingMode — внутренний перечисляемый тип, не предназначен для общего использования.

TLogFile - внутренний класс, не предназначен для общего использования.

TLogFormatter - внутренний класс, не предназначен для общего использования.

TLogFormatter 8Overf low - внутренний класс, не предназначен для общего использования.

TLogFormatter 16Overf low - внутренний класс, не предназначен для общего использования.

### **1.67. File Server Client Side**

CBaf IFileSortTable - внутренний класс, не предназначен для общего использования.

CDir - внутренний класс, не предназначен для общего использования.

CDirScan - внутренний класс, не предназначен для общего использования.

CFileBase - класс для управления файлами. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл f32file.h и библиотечный файл efsrv.lib.

CFileList - тип для хранения списка с открываемыми файлами. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл f32file.h.

CFileMan - класс сервиса управления файлами. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл f32file.h и библиотечный файл efsrv.lib.

CFindFileByType - класс для поиска файлов по заданному формату. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл bafindf.h.

MFileManObserver - класс интерфейса для отображения прогресса операций управления файлами. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл f32file.h и библиотечный файл efsrv.lib.

RDir - класс для чтения содержимого каталога. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл f32file.h и библиотечный файл efsrv.lib.

RFile - класс для выполнения операций над файлом. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл f32file.h и библиотечный файл efsrv.lib.

RFormat - класс для форматирования устройства. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл f32file.h и библиотечный файл efsrv.lib.

RFs - внутренний класс, не предназначен для общего использования.

RFsBase - внутренний класс, не предназначен для общего использования.

RRawDisk - внутренний класс, не предназначен для общего использования.

TDriveInfo - структура с информацией о диске. Доступна от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл f32file.h.

TDriveList - тип с модифицируемым буферным дескриптором, содержащим список имен диска. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл f32file.h.

TDriveName - тип с модифицируемым буферным дескриптором, содержащим имя диска. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл f32file.h.

TDriveNumber - перечисляемый тип с перечислением номеров диска. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл f32file.h.

TDriveUnit - класс, содержащий числа и символы для представления диска. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл f32file.h и библиотечный файл efsrv.lib.

TEntry - класс с уникальным идентификатором для входа в директорию с вложением. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл f32file.h и библиотечный файл efsrv.lib.

TEntryArray - класс с массивом запросов для чтения директории. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл f32file.h и библиотечный файл efsrv.lib.

TEntryKey - перечисляемый тип для назначения порядка сортировки входов в директорию. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл f32file.h.

TFileManError - перечисляемый тип, содержащий коды ошибки класса CFileMan. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл f32file.h.

TFileMode - перечисляемый тип, содержащий режимы доступа при открытии файла. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл f32file.h.

TFileName - тип, имя файла. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

TFileText - класс для чтения или записи одиночных строк текста. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл f32file.h и библиотечный файл efsrv.lib.

TFindFile - класс для поиска файлов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл f32file.h и библиотечный файл efsrv.lib.

TFormatMode — перечисляемый тип для установления метода форматирования. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл f32file.h.

TMediaPassword - тип, 8-битный модифицируемый буферный дескриптор, содержащий пароль. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл e32std.h.

TNotifyType - перечисляемый тип, содержащий флажки для назначения уведомления об изменении. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл f32file.h.

TOpenFileScan - класс для идентификации открытых файлов в сеансе для составления списка входов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл f32file.h и библиотечный файл efsrv.lib.

TParse - класс для анализа имен файлов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл f32file.h и библиотечный файл efsrv.lib.

TParseBase - базовый класс для анализа имен файлов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл f32file.h и библиотечный файл efsrv.lib.

TParsePtr - класс для анализа имен файлов на использование пространства стека. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл f32file.h и библиотечный файл efsrv.lib.

TParsePtrC - класс для анализа (без изменений) имен файлов на использование пространства стека. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл f32file.h и библиотечный файл efsrv.lib.

TSeek- перечисляемый тип с флажками для назначения адресата поиска. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл f32file.h.

TVolumeInfo - класс, том диска. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл f32file.h.

## **1.68. Writing a file system**

CFileSystem— класс интерфейса файлового сервера. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл f32fsys.h.

CDirCB - класс интерфейса файлового сервера, который представляет открытый каталог. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл f32fsys.h.

CFileCB - класс интерфейса файлового сервера, который представляет открытый файл. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл f32fsys.h.

CFileShare - внутренний класс, не предназначен для общего использования.

CFormatCB - класс, интерфейс файлового сервера, который представляет операцию форматирования. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл f32fsys.h.

CMountCB - класс, интерфейс файлового сервера, который представляет маркирование для диска подлежащего операции форматирования. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл f32fsys.h.

CRawDiskCB - класс файлового сервера, который представляет первозданный диск. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл f32fsys.h.

CAsyncNotifier - внутренний класс, не предназначен для общего использования.

CNonCriticalNotifier - внутренний класс не, предназначен для общего использования.

SFileShareLock - структура для блокировка части файла. Доступна от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл f32fsys.h.

TDrive - класс для представления диска в файловом сервере. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл f32fsys.h.

TFileOpen - перечисляемый тип для назначения операции над файлом. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл f32fsys.h.

TNotifierType - перечисляемый тип для определения критического или некритического уведомления. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл f32fsys.h.

TShare - тип файловой доли. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл f32fsys.h.

AllocBufferL () - функция для копирования данных в буфер. Доступна от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл f32fsys.h.

DebugNotifySessions () - функция, уведомитель сеанса отладки. Доступна от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл f32fsys.h.

FileServerIsHungO - функция для проверки зависания файлового сервера. Доступна от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл f32fsys.h.

IsDriveHung () - функция для проверки зависания диска. Доступна от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл f32fsys.h.

LocalDrive () - функция для получения локальной шины устройства. Доступна от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл f32fsys.h.

TheNotifier () — функция для асинхронного уведомления. Доступна от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл f32fsys.h.

WriteToDisk () - функция для записи данных из буфера в файл. Доступна от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл f32fsys.h.

### **1.69. File stores**

CDictionaryFileStore - класс, файл памяти словаря, в котором ассоциируется поток с UID. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл s32file.h и библиотечный файл estor.lib.

CDirectFileStore - класс, память файла прямого доступа для осуществления операций над потоками. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл s32file.h и библиотечный файл estor.lib.

CFileStore - класс, основная память файла. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл s32file.h и библиотечный файл estor.lib.

CPermanentFileStore — класс, постоянная память файла. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл s32file.h и библиотечный файл estor.lib.

FileStoreFactory — класс с функциями для открытия памяти файла. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл s32file.h и библиотечный файл estor.lib.

RFileBuf - класс, потоковый буфер, связанный с файлом. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл s32file.h и библиотечный файл estor.lib.

RFileReadStream - класс для чтения потока из файла. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл s32file.h и библиотечный файл estor.lib.

RFileWriteStream - класс для записи потока в файл. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл s32file.h и библиотечный файл estor.lib.

TFileStoreFactoryFunction - тип для установки указателя на функцию, которая открывает память файла. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл s32file.h.

### **1.70. Fixed Size Arrays**

TFixedArray - класс массивов C++ для проверки доступности значений индексов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

## 1.71. Fonts

**CBitmapFont** - класс растровых шрифтов. Для работы необходимо подключить заголовочный файл `fntstore.h`.

**CFbsFont** - класс для управления шрифтами. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `fbs.h` и библиотечный файл `fbscli.lib`.

**CFbsTypefaceStore** - класс, память шрифтов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `fbs.h` и библиотечный файл `gdi.lib`.

**CFontCache** - класс, содержащий кэш шрифта. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `gdi.h` и библиотечный файл `gdi.lib`.

**CFontStore** - класс памяти для хранения шрифтов. Для работы необходимо подключить заголовочный файл `fntstore.h`.

**CFont** - класс интерфейса шрифтов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `gdi.h` и библиотечный файл `gdi.lib`.

**CTypefaceStore** - класс интерфейса памяти шрифтов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `gdi.h` и библиотечный файл `gdi.lib`.

**SCharWidth** - класс, ширина символа. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `fbs.h`.

**TAlgStyle** - класс для стиля шрифта. Для работы необходимо подключить заголовочный файл `fntstore.h`.

**TCodeSection** - класс, определяющий раздел в коде для расположения информации. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `gdi.h`.

**TFontPosture** - перечисляемый тип с флажками для установления курсивного шрифта. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `gdi.h`.

**TFontPrintPosition** - перечисляемый тип с флажками для установления верхнего или нижнего индекса шрифта. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `gdi.h`.

**TFontSpec** — класс, спецификация шрифта. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `gdi.h` и библиотечный файл `gdi.lib`.

**TFontStrikethrough** - перечисляемый тип с флажками для назначения зачеркнутого шрифта. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `gdi.h`.

**TFontStrokeWeight** — перечисляемый тип с флажками для установления полужирного шрифта. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `gdi.h`.

TFontStyle - класс для установления стиля начертания шрифта. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл gdi.h.

TFontUnderline - перечисляемый тип с флажками для установления подчеркивания. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл gdi.h.

TGlyphBitmapType - перечисляемый тип для сглаживания масштабируемых шрифтов. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл gdi.h.

TTypef ace - класс для идентификации шрифтов по атрибутам. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл gdi.h.

TTypef aceSupport - класс для информации о шрифте. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл gdi.h.

## **1.72. Fonts and Bitmaps**

CFbsColor256BitmapUtil — класс для копирования точеного рисунка с 256-цветовой палитрой. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл fbs.h и библиотечный файл fbcli.lib.

RFbsSession- класс для сеанса с сервером шрифтов и bitmap-сервером. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл fbs.h и библиотечный файл fbcli.lib.

## **1.73. Front End Processors**

CCoeFepParameters - внутренний класс, не предназначен для общего использования.

CCoeFep - абстрактный класс для FEP. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл fepbase.h и библиотечный файл fepbase.lib.

CFepFastFileAccessor - внутренний класс, не предназначен для общего использования.

CFepGener icGlobalSettings - класс для записи и чтения параметров настройки FEP. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл fepbase.h и библиотечный файл fepbase.lib.

CFepGlobalDynamicFrequencyTable — внутренний класс, не предназначен для общего использования.

FepObserverHandleStartOf TransactionL () - функция для обработки старта FEP. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл fepbase.h и библиотечный файл fepbase.lib.

FEP\_END\_KEY\_EVENT\_HANDLER\_L - макрос, вызываемый с клавиатуры для возвращения FEP. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл fepbase.h и библиотечный файл fepbase.lib.

FEP\_END\_KEY\_EVENT\_HANDLER\_NO\_DOWN\_UP\_FILTER\_L - макрос, вызываемый с клавиатуры для возвращения FEP. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл feabase.h и библиотечный файл feabase.lib.

FEP\_START\_KEY\_EVENT\_HANDLER\_L - макрос, вызываемый с клавиатуры для запуска FEP. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл feabase.h и библиотечный файл feabase.lib.

FEP\_START\_KEY\_EVENT\_HANDLER\_NO\_DOWN\_UP\_FILTER\_L - макрос, вызываемый с клавиатуры для запуска FEP. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл feabase.h и библиотечный файл feabase.lib.

MCoeCaptionRetrieverForFep - класс для нахождения нередатируемого заголовка для использования FEP. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл feabase.h и библиотечный файл feabase.lib.

MCoeFepAwareTextEditor - класс протокола для FEP-редактора текста. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл feabase.h и библиотечный файл feabase.lib.

MCoeFepAwareTextEditor\_Extension1 — класс интерфейса для текстовых редакторов. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл feabase.h и библиотечный файл feabase.lib.

MCoeFepObserver - класс, протокол для отображения процесса FEP. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл feabase.h.

MFepAttributeStorer - класс, протокол атрибутов FEP. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл feabase.h и библиотечный файл feabase.lib.

MFepInlineTextFormatRetriever - класс, протокол для идентификации формата текста. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл fepitfr.h.

MFepPointerEventHandlerDuringInlineEdit - класс, протокол обработки событий указателя на изменения в тексте. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл feabase.h.

TFepOnOrOff KeyData - класс, данные включения или выключения FEP. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл feabase.h и библиотечный файл feabase.lib.

#### **1.74. FTP Engine**

CFtpProtocol — класс для обращения к командам FTP. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл FTPPROT.h и библиотечный файл ftpprot.lib.

CFTPSession - класс для регулировки сложности протокола FTP. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл FTPSESS.h и библиотечный файл ftpsess.lib.



MFTPProtocolNotifier - класс, интерфейс FTP для получения результата и состояния асинхронных команд. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл FTPPROT.h.

MFTPSessionNotifier — класс, интерфейс FTP для получения результата и состояния асинхронных операций. Для работы необходимо подключить заголовочный файл FTPSESS.h.

FTPPROTDLL\_VERSION\_MAJOR- главный номер версии FTPPROT.DLL. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл FTPPROT.h.

FTPPROTDLL\_VERSION\_MINOR- меньший номер версии FTPPROT.DLL. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл FTPPROT.h.

FTPPROTDLL\_VERSION\_NUMBER- номер версии FTPPROT.DLL. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл FTPPROT.h.

FTPSESS\_VERSION\_MINOR - меньший номер версии FTPSESS.DLL. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл FTPSESS.h.

FTPSESS\_VERSION\_MAJOR- главный номер версии FTPSESS.DLL. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл FTPSESS.h.

FTPSESS\_VERSION\_NUMBER- номер версии FTPSESS.DLL. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл FTPSESS.h.

KDefaultServerPort - порт сервера по умолчанию. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл FTPSESS.h.

## **1.75. Graphics**

CGraphicsContext - абстрактный класс графических контекстов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл gdi.h и библиотечный файл gdi.lib.

CGraphicsDevice - класс интерфейса для заданных устройств. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл gdi.h.

CPalette - класс для обеспечения поддержки GDI пользовательской палитры. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл gdi.h и библиотечный файл gdi.lib.

DynamicPalette - класс для переключения палитры в 256-цветовом режиме. Доступен от версии Symbian OS 7.0s. Для работы необходимо подключить заголовочный файл palette.h и библиотечный файл palette.lib.

MGraphicsDeviceMap - класс интерфейса для отображения определяемых устройством модулей. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл gdi.h и библиотечный файл gdi.lib.

MLineBreaker - класс для настройки алгоритма нарушения уникада. Для работы необходимо подключить заголовочный файл LineBreak.h.

MContingentLineBreaker - класс для определения конца строки. Доступен от версии Symbian OS 7.0s. Для работы необходимо подключить заголовочный файл LineBreak.h.

MContingentLineBreakerL - класс для определения конца строки. Доступен от версии Symbian OS 7.0s. Для работы необходимо подключить заголовочный файл LineBreak.h.

NumberConversion - класс для конвертирования чисел между разным начертанием. Доступен от версии Symbian OS 7.0s. Для работы необходимо подключить заголовочный файл NumberConversion.h.

TBidiLogicalToVisual - класс для упорядочивания текста. Для работы необходимо подключить заголовочный файл BidiVisual.h.

TBidiText - класс для простого форматирования текста. Для работы необходимо подключить заголовочный файл BidiText.h.

TColor256Util — класс для преобразования между объектом TRgb и индексом 256-цветовой палитры. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл gdi.h и библиотечный файл gdi.lib.

TDisplayMode - перечисляемый тип, режимы визуального отображения. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл gdi.h.

TDisplayModeUtils — класс с набором функций для получения информации о режиме визуального отображения. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл gdi.h и библиотечный файл gdi.lib.

TLinearDDA - класс, цифровой анализатор. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл gdi.h и библиотечный файл gdi.lib.

TMargins - класс с набором шаблонов для обрезания изображения. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл gdi.h и библиотечный файл gdi.lib.

TRgb - класс, 24-битное RGB-значение. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл gdi.h и библиотечный файл gdi.lib.

TZoomFactor - класс, обеспечивающий масштабирование изображения. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл gdi.h и библиотечный файл gdi.lib.

TBidirectionalState - класс с функциями алгоритм уникада для отображения текста на дисплее справа налево. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл bidi.h и библиотечный файл gdi.lib.

## **1.76. Graphics Foundations**

RRegionBuf - класс, буфер с предраспределенным пространством. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

RRegion — класс детализации для расширения региона. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

TPoint - класс для хранения двухмерной точки в декартовых координатах. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

TRect — класс для представления прямоугольника. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

TRegion - класс для представления региона. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

TRegionFix - класс для представления фиксированного региона. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

TSize - класс, значения ширины и высоты. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

### **1.77. Grid Foundations**

TCellRef — класс для идентификации ячейки по номеру столбца и строки. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл bacell.h и библиотечный файл bafl.lib.

TRangeRef - класс, идентифицирующий ячейки по ссылке на начальную и конечную ячейки. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл bacell.h и библиотечный файл bafl.lib.

operator+ () — добавляет число к строке и столбцу и возвращает результирующую ячейку. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл bacell.h и библиотечный файл bafl.lib.

operator- () - вычитает число строки и столбца и возвращает результирующую ячейку. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл bacell.h и библиотечный файл bafl.lib.

### **1.78. Grid**

CGridCellmg - класс, рисует содержимое ячейки сетки. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл grdstd.h и библиотечный файл grid.lib.

CGridCellRegion - класс, определяет выделенный регион сетки. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл grdcells.h и библиотечный файл grid.lib.

CGridlmg - класс, рисует содержимое сетки. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл grdstd.h и библиотечный файл grid.lib.

CGridLabelling - класс, рисует метку ячейки. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл grdstd.h и библиотечный файл grid.lib.

CGridLay - класс для контроля над размещением сетки. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл grdstd.h и библиотечный файл grid.lib.

CGridPrinter - класс для печати и предварительного просмотра сетки. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл gridprint.h и библиотечный файл grid.lib.

MGridCursorMoveCallBack - класс интерфейса функции обратного вызова, которая вызывается на каждое изменение положения курсора. Для работы необходимо подключить заголовочный файл grdstd.h.

MGridTable - класс интерфейса для отображения информации строки и столбца. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл grdstd.h.

TGridColors - класс, определяет цвет сетки. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл grdstd.h и библиотечный файл grid.lib.

TGridUtils - класс утилит сетки. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл grdstd.h и библиотечный файл grid.lib.

TMoveDirectionAndAmount - перечисляемый тип для определения расстояния и направления, в котором курсор должен двигаться при прокрутке. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл grddef.h.

## **1.79. Hardware Abstraction Layer (HAL)**

HAL - класс для получения информации и состояния от устройства. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл hal.h и библиотечный файл hal.lib.

HALData - класс, атрибуты HAL. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл haldata.h.

HalInternal - класс, внутреннее выполнение HAL. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл halint.h.

THalImplementation - внутренний тип, не предназначен для общего использования.

## **1.80. Handles**

RHandleBase - класс, дескриптор к объекту. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

TFindHandleBase - класс для поиска объектов типа Kernel. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

THandleInfo - класс с информацией об объекте. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

TOwnerType - перечисляемый тип для определения принадлежности дескриптора потоку. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

### **1.81. Hardware Accelerator**

RHwaDevice - класс для обработки аппаратного ускорителя. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл d32hwa.h и библиотечный файл euser.lib.

RHwaTask - класс, обработка задачи логическим аппаратным ускорителем. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл d32hwa.h и библиотечный файл euser.lib.

DHwaDevice — класс, логический канал. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл k32hwa.h.

DHwaTask - класс, задача логического аппаратного ускорителя. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл k32hwa.h.

DLogicalDeviceHwa - класс для создания универсального устройства логического аппаратного ускорения. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл k32hwa.h.

DHwaDevicePdd — класс, логический канал. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл k32hwa.h.

THwaTaskHwInterface - класс, задача физического аппаратного ускорителя. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл k32hwa.h.

DHwaPowerHandler — класс для управления питанием универсального логического аппаратного ускорителя. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл k32hwa.h.

THwaDataTransferDir - перечисляемый тип для руководства передачей данных. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл k32hwa.h.

THwaDataTransferReq - класс, запрос на передачу данных между задачей и потоком. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл k32hwa.h.

THwaDeviceStatus - перечисляемый тип, состояние аппаратного ускорителя. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл d32hwa.h.

THwaDeviceCapabilities - класс с информацией о возможностях аппаратного ускорителя. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл d32hwa.h.

THwaDeviceCapsBuf - тип, пакетный буфер для объекта THwaDeviceCapabilities типа. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл d32hwa.h.

TNwaDeviceResources - класс, информация о ресурсах доступных аппаратному ускорителю. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл d32hwa.h.

TNwaDeviceResourcesBuf — тип, пакетный буфер для объекта TNwaDeviceResources типа. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл d32hwa.h.

TNwaTaskResources - класс с информацией, описывающей требуемые ресурсы. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл d32hwa.h.

TNwaTaskResourcesBuf - тип, пакетный буфер для объекта TNwaTaskResources типа. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл d32hwa.h.

TNwaTaskStatus - перечисляемый тип, описывающий состояние выполнения задачи. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл d32hwa.h.

TTransferBufferInfo — класс с информацией, описывающей буфер для передачи данных. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл d32hwa.h.

TTransferBufferInfoBuf - тип, пакетный буфер для объекта TTransferBufferInfo типа. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл d32hwa.h.

TTaskConnectInfo - класс с информацией об отдельном подключении. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл d32hwa.h.

TTaskConnectInfoBuf - тип, пакетный буфер для объекта TTaskConnectInfo типа. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл d32hwa.h.

TTaskCreateInfo - класс с информацией для создания новой задачи. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл d32hwa.h.

TTaskCreateInfoBuf - тип, пакетный буфер для объекта TTaskCreateInfo типа. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл d32hwa.h.

TTaskDataInfo - класс с информацией для передачи данных между задачей и буфером. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл d32hwa.h.

TTaskDataInfoBuf - тип, пакетный буфер для объекта TTaskDataInfo типа. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл d32hwa.h.

TTaskGetMsgInfo - класс с информацией для получения сообщения от задачи. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл d32hwa.h.

TTaskLogonInfo - класс с информацией для запроса уведомления. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл d32hwa.h.

TTaskPriorityInfo - класс для установления приоритета задач. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл d32hwa.h.

TTaskPropertyInfo - класс с информацией, описывающей задачу аппаратного ускорителя. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл d32hwa.h.

TTaskPropertyInfoBuf - тип, пакетный буфер для объекта TTaskPropertyInfo типа. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл d32hwa.h.

TTaskSendMsgInfo - класс с информацией для посылки сообщения задаче. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл d32hwa.h.

TTaskStatusInfo - класс, информация о состоянии задачи. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл d32hwa.h.

TTaskStatusInfoBuf - тип, пакетный буфер для объекта TTaskStatusInfo типа. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл d32hwa.h.

TCapsDevHwaVOI - класс, информация о возможностях аппаратного ускорителя. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл d32hwa.h.

## 1.82. Help Model

CHlpCmdLine - внутренний класс, не предназначен для общего использования.

CHlpDatabases - внутренний тип, не предназначен для общего использования.

CHlpFileList - внутренний тип, не предназначен для общего использования.

CHlpItem - класс для создания элемента в файле справки. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл HLPMODEL.h и библиотечный файл HLPMODEL.LIB.

CHlpList - класс, содержащий список элементов справки. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл HLPMODEL.h и библиотечный файл HLPMODEL.LIB.

CHlpMode I - класс интерфейса справки. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл HLPMODEL.h и библиотечный файл HLPMODELLIB.

CHlpTopic - класс для создания раздела справки. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл HLPMODEL.h и библиотечный файл HLPMODELLIB.

HlpLauncher - класс, запускающий приложение справки. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл HLPPLCH.h и библиотечный файл HLPMODELLIB.

MHlpDbObserver - внутренний класс, не предназначен для общего использования.

MNhpModelObserver - класс, клиентский обратный интерфейс. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл HLPMODEL.h.

MNhpTitleArray - класс интерфейса для получения ID-раздела из массива. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл HLPMODEL.h.

MNhpZoomStateChangeNotifier - внутренний класс, не предназначен для общего использования.

MNhpZoomStateManager - внутренний класс, не предназначен для общего использования.

MNhpZoomStateObserver - внутренний класс, не предназначен для общего использования.

THhpZoomState - перечисляемый тип для масштабирования размеров. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл HLPMODEL.h.

## 1.83. NIP Client

CAuthenticationFilterInterface - класс для установки опознавательного фильтра. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл http\CAuthenticationFilterInterface.h и библиотечный файл http.lib.

CEComFilter - класс для дополнительно добавляемого фильтра к программе. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл http\CEComFilter.h и библиотечный файл http.lib.

CHttpFormEncoder - класс, поставляющий данные для HTML-представления. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл httpformencoder.h и библиотечный файл http.lib.

HTTP - класс, строковая таблица для HTTP-клиента. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл HttpStringConstants.h.

HTTPPanic - класс, содержащий коды паники. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл httperr.h.

HTTPStatus - класс с набором кодов состояния HTTP. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл httperr.h.

KUIdFilterPluginInterface - фильтр ECom. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл http\ce-comfilter.h.

MHTTPAuthenticationCallback - класс интерфейса для идентификации. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл http\mhttpauthenticationcallback.h.

MHTTPDataSupplier - класс интерфейса для получения сообщений с данными от каркаса. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл http\mhttpdatasupplier.h.



MHTTPFilterBase - базовый класс для фильтров и классов обратного вызова. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `http\mhttpfilterbase.h` и библиотечный файл `http.lib`.

MHTTPFilter - базовый класс для фильтров. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `http\mhttpfilter.h` и библиотечный файл `http.lib`.

MHTTPFilterCreationCallback - класс интерфейса повторного вызова для изменения фильтров. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `http\MHTTPFilterCreationCallback.h`.

MHTTPSessionEventCallback - класс интерфейса для повторного вызова обработчика событий. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `http\mhttpsessioneventcallback.h`.

MHTTPTransactionCallback - класс повторного вызова для получения событий. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `http\mhttptransactioncallback.h`.

RHTTPConnectionInfo - класс установки свойств для установления связи по протоколу. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `http\rhttpconnectioninfo.h` и библиотечный файл `http.lib`.

RHTTPFilterCollection - класс, дескриптор к набору фильтров. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `http\rhttpfiltercollection.h` и библиотечный файл `http.lib`.

RHTTPHeaders - класс, содержащий поля заголовка сообщения. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `http\rhttpheaders.h` и библиотечный файл `http.lib`.

RHTTPMessage - базовый класс сообщений. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `http\rhttpmessage.h` и библиотечный файл `http.lib`.

RHTTPPropertySet - базовый класс для установления свойств. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `http\rhttppropertyset.h` и библиотечный файл `http.lib`.

RHTTPRequest - класс, сообщение-запрос. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `http\rhttprequest.h` и библиотечный файл `http.lib`.

RHTTPResponse - класс, сообщение-ответ. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `http\rhttpresponse.h` и библиотечный файл `http.lib`.

RHTTPSession - класс, дескриптор сеанса подключения. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `http\rhttpsession.h` и библиотечный файл `http.lib`.

RHTTPTransaction - класс, структура запросов. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `http\RHTTPTransaction.h` и библиотечный файл `http.lib`.

RHTTPTransactionPropertySet - класс, содержащий набор свойств транзакции. Доступен от версии Symbian OS 7.0. Для работы необходимо под-

ключить заголовочный файл `http\rhttptransactionpropertyset.h` и библиотечный файл `http.lib`.

`TFilterConfigurationIterator` - класс для операций со списком фильтров. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `http\TFilterConfigurationIter.h` и библиотечный файл `http.lib`.

`TFilterInformation` - структура с информацией о текущем фильтре. Доступна от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `http\TFilterInformation.h` и библиотечный файл `http.lib`.

`THTTPEvent` - класс, сообщение структуры запросов. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `http\thttpevent.h` и библиотечный файл `http.lib`.

`THTTPSessionEvent` - класс, сообщение сеанса структуры запросов. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `http\thttpevent.h` и библиотечный файл `http.lib`.

`THTTPFilterHandle` - класс, дескриптор регистрации фильтра. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `http\thttpfilterhandle.h` и библиотечный файл `http.lib`.

`THTTPFilterIterator` — класс, итерация для регистрации фильтра со специфическим именем. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `http\thttpfilteriterator.h` и библиотечный файл `http.lib`.

`THTTPFilterRegistration` - класс для регистрации фильтра в очереди фильтров. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `http\thttpfilterregistration.h` и библиотечный файл `http.lib`.

`THTTPHdrFieldIter` - класс для выполнения итерации в пределах `RHTTPHeaders`. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `http\thttphdrfielditer.h` и библиотечный файл `http.lib`.

`THTTPHdrVal` - класс для отображения значения из HTTP-заголовка. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `http\thttphdrval.h` и библиотечный файл `http.lib`.

`WSP` - класс, строковая таблица HTTP-клиента для хранения WSP- полей заголовка. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `WspStringConstants.h`.

`WSPCharacterSets` - класс, строковая таблица HTTP-клиента для определения допустимых символов, используемых в WSP. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `WspCharacterSets.h`.

`WSPContentTypes` - класс, строковая таблица HTTP-клиента для определения типа MIME для использования с WSP. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `WspContentTypes.h`.

`WSPLanguages` - класс для определения символов, поддерживаемых в кодировке WSP. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `WspLanguages.h`.

WSPParam - класс, строковая таблица HTTP-клиента для определения параметра имен используемых WSP. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл WspParamConstants.h.

WSPRegContentTypes - класс, строковая таблица HTTP-клиента для определения зарегистрированного типа, используемого WSP. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл WspReg-ContentTypes.h.

WSPStdConstants - класс, строковая таблица HTTP-клиента для определения констант, используемых WSP. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл WspStdConstants.h.

WSPTypeConstants - класс, строковая таблица HTTP-клиента для определения типа WSP PDU. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл WspTypeConstants.h.

### **1.84. HTTP Message**

RHttpRequestComposer — класс для создания HTTP/1.1 -сообщения. Для работы необходимо подключить заголовочный файл rhttpmessagecomposer.h и библиотечный файл httpmessage.lib.

RHttpRequestParser - класс для анализа HTTP/1.1-сообщений. Для работы необходимо подключить заголовочный файл rhttpmessageparser.h и библиотечный файл httpmessage.lib.

MHttpRequestParserObserver - класс, поставляющий данные HTTP-сообщения для синтаксического анализатора. Для работы необходимо подключить заголовочный файл mhttpmessageparserobserver.h.

MHttpRequestComposerObserver - класс, поставляющий данные HTTP-сообщения для HTTP-композера и получающий события и ошибки. Для работы необходимо подключить заголовочный файл mhttpmessagecomposerobserver.h.

THttpRequestPanic - класс с функцией паники и кодами для компонентов HTTP-сообщения. Для работы необходимо подключить заголовочный файл thttpmessagepanic.h.

### **1.85. HTTP Transport Layer**

CHttpTransportLayer - класс, интерфейс для поддержки протокола для HTTP. Для работы необходимо подключить заголовочный файл chttptransport-layer.h.

MConnectionPrefsProvider - класс, интерфейс настроек подключения к транспортному уровню HTTP. Для работы необходимо подключить заголовочный файл mconnectionprefsprovider.h.

MInputStream - класс, управляющий входным потоком сокета подключения. Для работы необходимо подключить заголовочный файл minputstream.h.

MInputStreamObserver - класс для получения данных и уведомлений от входного потока сокета подключения. Для работы необходимо подключить заголовочный файл minputstreamobserver.h.

MOutputStream - класс для передачи данных входному потоку сокета подключения. Для работы необходимо подключить заголовочный файл moutputstream.h.

MOutputStreamObserver - класс для получения уведомлений от выходного потока. Для работы необходимо подключить заголовочный файл moutputstreamobserver.h.

MSocketConnectObserver - класс интерфейса для получения уведомлений от сокета подключения транспортного уровня HTTP. Для работы необходимо подключить заголовочный файл msocketconnectobserver.h.

MSocketConnector - класс, управляющий запросом для связи. Для работы необходимо подключить заголовочный файл msocketconnector.h.

MSocketListenObserver - класс интерфейса для получения уведомлений от транспортного уровня HTTP. Для работы необходимо подключить заголовочный файл msocketlistenobserver.h.

MSocketFactory - класс, сокет подключения для HTTP. Для работы необходимо подключить заголовочный файл msocketfactory.h.

THttpTransportConstructionParams - класс, содержащий параметры для ECom интерфейса. Для работы необходимо подключить заголовочный файл chhttptransportlayer.h.

THttpTrLayerPanic - класс с функцией паники и кодами для HTTP-уровня. Для работы необходимо подключить заголовочный файл thhttptrlayerpanic.h.

## **1.86. Image Converter**

CMdaBitmapRotator - класс для преобразования точечного рисунка. Доступен в Symbian OS с 5.0 по 7.0. Для работы необходимо подключить заголовочный файл MdalmageConverter.h и библиотечный файл MediaClientImage.lib.

CMdaBitmapScaler - класс для преобразования точечного рисунка. Доступен в Symbian OS с 5.0 по 7.0. Для работы необходимо подключить заголовочный файл MdalmageConverter.h и библиотечный файл MediaClientImage.lib.

CMdalmageBitmapToBitmapUtility - класс, часть библиотеки конвертирования изображения. Доступен в Symbian OS с 5.0 по 7.0 версию. Для работы необходимо подключить заголовочный файл MdalmageConverter.h и библиотечный файл MediaClientImage.lib.

CMdalmageBitmapToDescUtility - класс, часть библиотеки конвертирования изображения. Доступен в Symbian OS с 5.0 по 7.0 версию. Для работы необходимо подключить заголовочный файл MdalmageConverter.h и библиотечный файл MediaClientImage.lib.

CMdalmageBitmapToFileUtility - класс, часть библиотеки конвертирования изображения. Доступен в Symbian OS с 5.0 по 7.0 версию. Для работы необходимо подключить заголовочный файл MdalmageConverter.h и библиотечный файл MediaClientImage.lib.

CMdalmageDataReadUtility - доступен в Symbian OS с 6.0 по 7.0 версию. Для работы необходимо подключить заголовочный файл mdaimageconverter.h и библиотечный файл mediaclientimage.lib.

CMdalmageDataWriteUtility - доступен в Symbian OS с 6.0 по 7.0 версию. Для работы необходимо подключить заголовочный файл mdaimageconverter.h и библиотечный файл mediaclientimage.lib.

CMdalmageDescToBitmapUtility - класс, часть библиотеки конвертирования изображения. Доступен в Symbian OS с 5.0 по 7.0 версию. Для работы необходимо подключить заголовочный файл MdalmageConverter.h и библиотечный файл MediaClientImage.lib.

CMdalmageFileToBitmapUtility - класс, часть библиотеки конвертирования изображения. Доступен в Symbian OS с 5.0 по 7.0 версию. Для работы необходимо подключить заголовочный файл MdalmageConverter.h и библиотечный файл MediaClientImage.lib.

CMdalmageUtility - класс, часть библиотеки конвертирования изображения. Доступен в Symbian OS с 5.0 по 7.0 версию. Для работы необходимо подключить заголовочный файл MdalmageConverter.h и библиотечный файл MediaClientImage.lib.

CMdaResource - доступен в Symbian OS с 6.0 по 7.0 версию. Для работы необходимо подключить заголовочный файл mda\client\utility.h.

MMdalmageUtilObserver - доступен в Symbian OS с 5.0 по 7.0 версию. Для работы необходимо подключить заголовочный файл MdalmageConverter.h и библиотечный файл MediaClientImage.lib.

TFrameInfo - класс, содержащий информацию о фрейме. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл mdaimageconverter.h.

### **1.87. Incremental Matcher**

RIncrMatcherBase - базовый класс для возрастающего сравнения двух текстовых буферов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл bamatch.h и библиотечный файл bafl.lib.

RIncrMatcherBuf - класс для сравнения текста с модифицируемым дескрипторным буфером. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл bamatch.h и библиотечный файл bafl.lib.

RIncrMatcherPtr - класс для сравнения текста с дескриптором. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл bamatch.h и библиотечный файл bafl.lib.

RIncrMatcherTextBuf - класс для сравнения текста с текстовым буфером переменной длины. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл bamatch.h и библиотечный файл bafl.lib.

RTextBuf - класс для формирования текстовой строки. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл bamatch.h и библиотечный файл bafl.lib.

### **1.88. Interface to Resource Files**

RResourceFile - класс для чтения данных ресурса в буфер. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл barsc.h и библиотечный файл bafl.lib.

TResourceReader - класс для чтения данных из файла ресурса. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл barsread.h и библиотечный файл bafl.lib.

ARRAY - структура ресурсов, содержит массив переменной длины. Доступна от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл badef.h.

BA\_RSS\_SIGNATURE - структура ресурсов, использует ресурс из файла ресурса. Доступна от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл badef.h.

LBUF - структура ресурсов, содержит одну нулевую строку. Доступна от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл badef.h.

TBUF - структура ресурсов, содержит одну ненулевую строку. Доступна от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл badef.h.

### **1.89. Internet Mail**

CImap4ClientMtm - класс, клиентский IMAP4 MTM-интерфейс. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл impcmtm.h и библиотечный файл imcm.lib.

CImBaseEmailSettings - базовый класс для классов, которые формируют параметры почтового Интернет-сервиса. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл MIUTSET.h и библиотечный файл imcm.lib.

CImCacheManager - класс, менеджер локального кэша в сообщениях удаленных почтовых ящиков. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл CACHEMAN.himcm.h.

CImEmailMessage - класс, создает почтовое сообщение. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл MIUTMSG.h и библиотечный файл imcm.lib.

CImEmailOperation - класс, обеспечивающий почтовые функции для операций с сообщениями. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл MIUTMSG.h и библиотечный файл imcm.lib.

CImHeader - класс для создания заголовка Интернет-сообщения. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл MIUTHDR.h и библиотечный файл imcm.lib.

CImIAPPferences - класс, настройки для почтовой службы. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл IAPPrefs.h и библиотечный файл imcm.lib.

CImImap4GetMail - класс для загрузки сообщения из отдаленного ящика в локальную папку. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл IMPCMTM.h и библиотечный файл imcm.lib.

CImImap4Settings — класс, параметры сеанса IMAP4 MTM. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл imapset.h и библиотечный файл imcm.lib.

`CI mMimeHeader` - класс для хранения заголовка электронной почты. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `MIUTHDR.h` и библиотечный файл `imcm.lib`.

`CI mPOP3GetMail` - класс для копирования или перемещения электронных сообщений в локальную папку. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `POP3MTM.h` и библиотечный файл `imcm.lib`.

`CI mPop3Settings` - класс, параметры настройки POP3. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `POP3SET.h` и библиотечный файл `imcm.lib`.

`CI mPruneMessage` - класс для удаления данных и вложения сообщения. Для работы необходимо подключить заголовочный файл `CACHEMAN.h` и библиотечный файл `imcm.lib`.

`CI mSmtпSettings` - класс, содержащий параметры настройки SMTP. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `SMTPSET.h` и библиотечный файл `imcm.lib`.

`CI mPop3ClientMtm` — класс, клиентский интерфейс POP3 MTM. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `POP3MTM.h` и библиотечный файл `imcm.lib`.

`CI mSmtпClientMtm` - класс, клиентский интерфейс SMTP MTM. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `SMTPCMTM.h` и библиотечный файл `imcm.lib`.

`MI mUndoOf f LineOperation` - внутренний класс, не предназначен для общего использования.

`MI mURITranslator` - класс для транслирования ссылок URI в XHTML. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл `miutmsg.h`.

`MI msvImapConnectionObserver` - класс для обработки IMAP4-подключения. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `ImapConnectionObserver.h`.

`TI mFolderSubscribeType` - перечисляемый тип для синхронизации IMAP4 субскриптовой информации с сервером. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `IMAPSET.h`.

`TI mFolderSyncType` - перечисляемый тип для синхронизации IMAP4-информации папки с сервером. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `IMAPSET.h`.

`TI mImap4Cmds` - перечисляемый тип с набором IMAP4-командами. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `IMAPCMDS.h`.

`TI mImap4CompoundProgress` - класс для отображения прогресса IMAP-операции. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `IMAPSET.h`.

`TI mImap4GenericProgress` - класс для отображения прогресса IMAP операции получения. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `IMAPSET.h`.

**TMap4GetMailOptions** - перечисляемый тип для назначения компонентов сообщения, которые должны быть скопированы с сервера. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл **IMAPSET.h**.

**TMap4ProgressType** - перечисляемый тип с набором флажков для установки типа возвращаемой IMAP4-информации прогресса. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл **IMAPSET.h**.

**TMap4RenameFolder** - класс, буфер для имени IMAP-папки. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл **IMAPCMDS.h**.

**TMap4SyncProgress** - класс для отображения информации о прогрессе синхронизации. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл **IMAPSET.h**.

**TMapConnectionEvent** - перечисляемый тип, отображение событий подключения IMAP4. Для работы необходимо подключить заголовочный файл **ImapConnectionObserver.h**.

**TImAttachmentInfo** - класс, свойства почтового вложения. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл **MIUTMSG.h**.

**TImCacheManagerProgress** - структура, операция очистки кэша. Доступна от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл **CACHEMAN.h**.

**TImDisconnectedOperationType** - перечисляемый тип для описания локальных почтовых операций. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл **MIUTHDR.h**.

**TImEmailFolderType** - перечисляемый тип, флажки для определения типа папки. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл **MIUTHDR.h**.

**TImEmailTransformingInfo** - класс для преобразования символов и настройки кодировки для отправки SMTP-сообщения. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл **miutconv.h** и библиотечный файл **imcm.lib**.

**TImEncodingType** — перечисляемый тип, определяющий тип кодировки. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл **MIUTHDR.h**.

**TImHeaderEncodingInfo** - класс для сохранения данных кодирования из

заголовков. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл **MIUTHDR.h** и библиотечный файл **imcm.lib**.

**TImIAPChoice** - класс, использование диалога подключения для почтового Интернет-сервиса. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл **IAPPrefs.h**.

**TImMailFileProgress** - класс, содержащий информацию о состоянии отправленного сообщения. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл **SMTPSET.h** и библиотечный файл **imcm.lib**.



TImMap4GetMailInfo - класс, информация о сообщении. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл IMAPSET.h и библиотечный файл imcm.lib.

TImPop3GetMailInfo — класс, информация о сообщении для POP3-операции получения. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл POP3SET.h и библиотечный файл imcm.lib.

TImSendMethod — перечисляемый тип, определяющий метод посылки почтового сообщения. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл miutconv.h.

TImSmtpProgress - класс, содержащий информацию о прогрессе операции SMTP. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл SMTPSET.h и библиотечный файл imcm.lib.

TImSMTPSendCopyToSelf - перечисляемый тип для посылки пользователю по электронной почте копий всех электронных сообщений, отправленных с телефона. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл SMTPSET.h.

TImSMTPSendMessageOption - перечисляемый тип для определения способа отсылки почтовых сообщений. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл SMTPSET.h.

TMsglmOutboxSendState — перечисляемый тип, отображающий состояние SMTP MTM. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл SMTPSET.h.

TMsgOutboxBodyEncoding - перечисляемый тип для назначения способа кодирования электронного сообщения. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл MIUTHDR.h.

TMsvEmailEntry - класс для хранения дополнительной информации сообщения. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл miuthdr.h и библиотечный файл imcm.lib.

TMsvEmailTypeList — тип, флажки для создания Интернет-сообщения, используемые в aMsvEmailTypeList параметре. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл MIUTDEF.h.

TPop3Cmds - перечисляемый тип с набором POP3-команд. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл POP3CMDS.h.

TPop3GetMailOptions - перечисляемый тип, флажки для определения типа загрузки сообщения. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл POP3SET.h.

TPop3Progress - класс, содержащий информацию о прогрессе POP3- операции. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл POP3SET.h и библиотечный файл imcm.lib.

TSmtpCmds - перечисляемый тип с набором SMTP-команд. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл SMTPCMDS.h.

TSmtpSessionState - перечисляемый тип, состояние посылки SMTP. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл SMTPSET.h.

### **1.90. Internet Protocol Utility**

CAuthority - тип для класса, который осуществляет создание и изменение анализируемых компонентов. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл Authority16.h.

CAuthority8 - управляющий класс, формирующий 8-битные данные. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл Authority8.h и библиотечный файл InetProtUtil.lib.

CAuthority16 - управляющий класс, формирующий 16-битные данные. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл Authority16.h и библиотечный файл InetProtUtil.lib.

CDelimitedDataBase - тип для класса, который осуществляет создание и изменение 16-битных данных. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл DelimitedParser16.h.

CDelimitedDataBase8 - класс, который осуществляет создание и изменение 8-битных данных. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл DelimitedParser8.h и библиотечный файл InetProtUtil.lib.

CDelimitedDataBase16 — класс, который осуществляет создание и изменение 16-битных данных. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл DelimitedParser16.h и библиотечный файл InetProtUtil.lib.

CDelimitedPath - тип для класса, который осуществляет создание и изменение 16-битных путей, разделенных символом «/». Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл DelimitedParser16.h.

CDelimitedPath8 - класс, который осуществляет создание и изменение 8-битных путей, разделенных символом «/». Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл DelimitedParser8.h и библиотечный файл InetProtUtil.lib.

CDelimitedPath16 — класс, который осуществляет создание и изменение 16-битных путей, разделенных символом «/». Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл DelimitedParser16.h и библиотечный файл InetProtUtil.lib.

CDelimitedPathSegment - тип для класса, который осуществляет создание и изменение 16-битных долей пути, разделенных символом «;». Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл DelimitedPathSegment16.h.

CDelimitedPathSegment8 - класс, который осуществляет создание и изменение 8-битных долей пути, разделенных символом «;». Доступен от версии

Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `DelimitedPathSegment8.h` и библиотечный файл `InetProtUtil.lib`.

`CDelimitedPathSegment16` - класс, который осуществляет создание и изменение 16-битных долей пути, разделенных символом «; >>». Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `DelimitedPathSegment16.h` и библиотечный файл `InetProtUtil.lib`.

`CDelimitedQuery` - тип для класса, который осуществляет создание и изменение 16-битных неограниченных запросов, разделенных символом «&». Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `DelimitedQuery16.h`.

`CDelimitedQuery8` - класс, который осуществляет создание и изменение 8-битного неограниченного запроса, компоненты которого разделены символом «&». Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `DelimitedQuery8.h` и библиотечный файл `InetProtUtil.lib`.

`CDelimitedQuery16` - класс, который осуществляет создание и изменение 16-битного неограниченного запроса, компоненты которого разделены символом «&». Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `DelimitedQuery16.h` и библиотечный файл `InetProtUtil.lib`.

`CUri` - тип для класса, который осуществляет создание и изменение анализируемых 16-битных компонентов URI. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `Uri16.h`.

`CUri8` - класс, который осуществляет создание и изменение анализируемых 8-битных компонентов URI. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `Uri8.h` и библиотечный файл `InetProtUtil.lib`.

`CUri16` - класс, который осуществляет создание и изменение анализируемых 16-битных компонентов URI. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `Uri16.h` и библиотечный файл `InetProtUtil.lib`.

`CWspHeaderEncoder` - класс для кодирования поля заголовка. Для работы необходимо подключить заголовочный файл `WSPEncoder.h` и библиотечный файл `InetProtUtil.lib`.

`EscapeUtils` - класс для кодирования-декодирования и конвертирования данных. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `EscapeUtils.h` и библиотечный файл `InetProtUtil.lib`.

`InetProtTextUtils` - класс, набор утилит для работы с текстом. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `InetProtTextUtils.h` и библиотечный файл `InetProtUtil.lib`.

`TAuthorityC` - тип для класса, который анализирует 16-битные компоненты. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `Authority16.h`.

`TAuthorityC8` — класс для создания неизменяемого обращения к анализируемым компонентам на 8 бит. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `Authority8.h` и библиотечный файл `InetProtUtil.lib`.

TAuthorityCl 6 - класс для создания неизменяемого обращения к анализируемым компонентам на 8 бит. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл Authority 16.h и библиотечный файл InetProtUtil.lib.

TAuthorityComponent - перечисляемый тип для идентификации компонентов управления. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл AuthorityCommon.h.

TAuthorityParser - тип для класса, который анализирует 16-битные компоненты управления. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл Authority 16.h.

TAuthorityParser8 - класс, который анализирует 8-битные компоненты управления. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл Authority8.h и библиотечный файл euser.lib.

TAuthorityParser16 - класс, который анализирует 16-битные компоненты управления. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл Authority16.h и библиотечный файл InetProtUtil.lib.

TDelimitedDataParseMode — перечисляемый тип, режимы анализатора неограниченных данных. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл DelimitedParserCommon.h.

TDelimitedParserBase - тип для класса, который анализирует 16-битные данные. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл DelimitedParser16.h.

TDelimitedParserBase8 - класс, который осуществляет анализ 8-битных данных, разделенных символом. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл DelimitedParser8.h и библиотечный файл InetProtUtil.lib.

TDelimitedParserBase16 - класс, который осуществляет анализ 16-битных данных, разделенных символом. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл DelimitedParser16.h и библиотечный файл InetProtUtil.lib.

TDelimitedPathParser — тип для класса, который анализирует пути разделенные символом «/». Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл DelimitedParser16.h.

TDelimitedPathParser8 - класс, который анализирует 8-битные пути разделенные символом «/». Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл DelimitedParser8.h и библиотечный файл InetProtUtil.lib.

TDelimitedPathParser16 - класс, который анализирует 16-битные пути разделенные символом «/». Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл DelimitedParser16.h и библиотечный файл InetProtUtil.lib.

TDelimitedQueryParser - тип для класса, который анализирует запросы, разделенные символом «&». Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл DelimitedQuery16.h.

`TDelimitedQueryParser8` - класс, который анализирует 8-битные запросы, разделенные символом «&». Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `DelimitedQuery8.h` и библиотечный файл `InetProtUtil.lib`.

`TDelimitedQueryParser16` - класс, который анализирует 16-битные запросы, разделенные символом «&». Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `DelimitedQuery16.h` и библиотечный файл `InetProtUtil.lib`.

`TDelimitedPathSegmentParser` - тип для класса, который анализирует доли пути, разделенные символом «:». Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `DelimitedPathSegment16.h`.

`TDelimitedPathSegmentParser8` — класс, который анализирует 8-битные доли пути, разделенные символом «;». Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `DelimitedPathSegment8.h` и библиотечный файл `InetProtUtil.lib`.

`TDelimitedPathSegmentParser16` - класс, который анализирует 16-битные доли пути, разделенные символом «;». Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `DelimitedPathSegment16.h` и библиотечный файл `InetProtUtil.lib`.

`TInternetDate` - класс для хранения даты в универсальном формате времени. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `tinternetdate.h` и библиотечный файл `InetProtUtil.lib`.

`TInternetDateParser` - внутренний класс, не предназначен для общего использования.

`TUriComponent` — перечисляемый тип для идентификации типа URI-компонента. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `UriCommon.h`.

`TUriC` - тип для класса, который обеспечивает неизменяемое обращение к анализируемым 16-битным компонентам URI. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `Uri16.h`.

`TUriC8` - класс, который обеспечивает неизменяемое обращение к анализируемым 8-битным компонентам URI. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `Uri8.h` и библиотечный файл `InetProtUtil.lib`.

`TUriC16` - класс, который обеспечивает неизменяемое обращение к анализируемым 16-битным компонентам URI. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `Uri16.h` и библиотечный файл `InetProtUtil.lib`.

`TUriParser` - тип для класса, который анализирует 16-битные данные в компонентах URI. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `Uri16.h`.

`TUriParser8` - класс, который анализирует 8-битные данные в компонентах URI. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `Uri8.h` и библиотечный файл `InetProtUtil.lib`.

TUriParser16 - класс, который анализирует 16-битные данные в компонентах URI. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл Uri16.h и библиотечный файл InetProtUtil.lib.

TWspField - класс для создания пары значений имен заголовка. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл WSPDecoder.h и библиотечный файл InetProtUtil.lib.

TWspHeaderSegmenter - класс для разделения WSP-буфера на пары значений имен заголовка WSP. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл WSPDecoder.h и библиотечный файл InetProtUtil.lib.

TWspPrimitiveEncoder - класс, кодирующие функции стандарта WSP, 8-битные данные. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл WSPEncoder.h и библиотечный файл InetProtUtil.lib.

TWspPrimitiveDecoder - класс для декодирования WSP. Для работы необходимо подключить заголовочный файл WSPDecoder.h и библиотечный файл InetProtUtil.lib.

UriUtils - класс, набор функций для обработки URI. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл UriUtils.h и библиотечный файл InetProtUtil.lib.

TUriUtilsError - внутренний перечисляемый тип, не предназначен для общего использования.

TWspCodecPanic - внутренний перечисляемый тип, не предназначен для общего использования.

TWspDecoderPanic - внутренний перечисляемый тип, не предназначен для общего использования.

### **1.91. Interrupt architecture**

TDFc - класс, функциональные возможности DFC. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл k32std.h и библиотечный файл ekern.lib.

TInterrupt - класс, функциональные возможности прерываний. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл m32std.h и библиотечный файл ekern.lib.

### **1.92. IPSec**

CIPSecAPI - класс, высокоуровневый IPSec, служит для управления ключами. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл ipsecapi.h и библиотечный файл ipsecapi.lib.

TIPSecPolicy - класс, содержащий IPSec-идентификатор, имя и статус. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл ipsectypes.h.

TIPSecPolicyDetails - класс, содержащий информацию IPSec. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл ipsectypes.h.

TKeyType - перечисляемый тип, определяющий тип секретного ключа. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл ipsectypes.h.

TPolicyID - класс, идентификатор безопасности. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл ipsectypes.h.

TPolicyStatus - перечисляемый тип, состояние IPSec. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл ipsectypes.h.

TPolicyType - перечисляемый тип для определения типа IPSec. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл ipsectypes.h.

### **1.93. IrDA Sockets**

TIASCharSet - перечисляемый тип, содержащий набор для кодирования символов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл ir\_sock.h

TIASDatabaseEntry - класс, запись IAS в сетевой базе данных. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл ir\_sock.h и библиотечный файл irda.lib.

TIASDatabaseEntryVOOI - класс, запись данных для IAS в сетевой базе данных. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл ir\_sock.h и библиотечный файл irdalib.

TIASDataType - перечисляемый тип для идентификации ответа от запроса IAS. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл ir\_sock.h.

TIASQueue - класс для формирования IAS-запроса на IAS-сервер другого устройства. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл ir\_sock.h и библиотечный файл irdalib.

TIASResponse - класс, ответ на IAS-запрос на IAS-сервер другого устройства. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл ir\_sock.h и библиотечный файл **irda.lib**.

TIrdaSockAddr - класс, адрес IrDA-сокета. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл irsock.h и библиотечный файл irda.lib.

### **1.94. InfraRed Transfer Picture Protocol**

CTranpSession - класс, назначающий состояние для отправки и получения изображения от устройства. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл tranp.h и библиотечный файл irtranp.lib.

MTranpNotif ication - класс интерфейса, уведомляющий о событиях, которые происходят во время передачи изображения. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл tranp.h.

TLatticeSize - перечисляемый тип для кодирования размера изображения. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл tranp.h.

TTranpConf ig - класс, содержащий параметры конфигурации сеанса передачи. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл tranp.h и библиотечный файл irtranp.lib.

TTranpPicture - класс, формат для сохранения полученного по инфракрасному порту изображения. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл tranp.h и библиотечный файл irtranp.lib.

### **1.95. Lexical Analysis**

TLex16 - класс с функциями для синтаксического анализа и числового преобразования для 16-битной строки в уникоде. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

TLex8 - класс с функциями для синтаксического анализа и числового преобразования для 8-битной строки не в уникоде. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

TLexMar kl 6 - класс, метка для класса TLex1 6, указывающая на элемент для анализа. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

TLexMark8 - класс, метка для класса TLex8, указывающая на элемент для анализа. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

TLexMar k - тип, метка для класса TLex, указывающая на элемент для анализа. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

TLex - тип, обеспечивающий доступ к функциям для синтаксического анализа и числового преобразования для уникода и не уникода. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

### **1.96. Literal Descriptors**

\_L16 - макрос, который определяет константную 16-битную литеру для уникода. Доступна от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32def.h.

\_L8 - макрос, который определяет константную 8-битную литеру для не уникода. Доступна от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32def.h.



`_LIT16` - константный литеральный дескриптор на 16 бит для уника и не уника. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `e32def.h`.

`_BIT8` - константный литеральный дескриптор на 8 бит для уника и не уника. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `e32def.h`.

`_LIT` - независимый константный литеральный дескриптор. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `e32def.h`.

`_L` - макрос, который определяет константную 8-битную литеру для не уника или 16-битную для уника. Доступна от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `e32def.h`.

`_S16` - определяет 16-битную строку. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `e32def.h`.

`_S8` — определяет 8-битную строку. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `e32def.h`.

`_S` - определяет 8-битную или 16-битную строку. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `e32def.h`.

`TLitC16`— класс для создания литерального текста с использованием `_BIT16`. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `e32des16.h` и библиотечный файл `euser.lib`.

`TLitC8` - класс для создания литерального текста с использованием `_BIT8`. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `e32des8.h` и библиотечный файл `euser.lib`.

`TLitC` — класс для создания литерального текста с использованием `LIT`. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `e32std.h` и библиотечный файл `euser.lib`.

`TRefDesC16`- тип, ссылка для использования в `TLitC16::_____TRefDesC16()`. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `e32des16.h`.

`TRefDesC8` - тип, ссылка для использования в `TLitC8::_____TRefDesC8()`. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `e32des8.h`.

`TRefDesC`- тип, ссылка для использования в `TLitC::_____TRefDesC()`. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `e32std.h`.

## 1.97. Locale Settings

`TAmPm` - перечисляемый тип для определения времени до полудня или после полудня. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `e32std.h`.

`TAmPmName` - класс для определения местного обозначения времени до полудня и после полудня. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `e32std.h` и библиотечный файл `euser.lib`.

TClockFormat - перечисляемый тип для отображения аналоговых или цифровых часов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

TCollationMethod - структура для определения метода сортировки текста. Доступна от версии Symbian OS 5.1. Для работы необходимо подключить заголовочный файл collate.h.

TCurrencySymbol - класс, получает копию символа, который служит для местного обозначения денежной единицы. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

TDateFormat - перечисляемый тип, формат даты. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

TDateSuffix - класс, получает суффикс для прибавления к каждому дню месяца. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

TDay - перечисляемый тип с перечислением дней недели. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

TDaylightSavingZone - перечисляемый тип с перечислением зон для перевода времени. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

TDayNameAbb - класс, получает сокращенное название дней недели. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

TDayName - класс, получает полное название дней недели. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

TDigitType - перечисляемый тип для определения режима отображения чисел. Доступен от версии Symbian OS 7.0s. Для работы необходимо подключить заголовочный файл e32std.h.

TLanguage - перечисляемый тип с перечислением значений для идентификации языков. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

TLocale - класс, устанавливающий локальные системные настройки. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

TLocalePos — перечисляемый тип для установления символа денежной единицы или времени до или после валюты или времени. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и.

TMonth - перечисляемый тип с перечислением месяцев. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

TMonthNameAbb - класс, получает сокращенное название месяцев. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

TMonthName - класс, получает полное название месяцев. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

TTimeFormat - перечисляемый тип для 12- или 24-часового формата времени. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

TUnitsFormat — перечисляемый тип для перечисления модулей измерения. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

## **1.98. Log Engine**

CLogActive - абстрактный класс, определяющий поведение объектов для регистрационных классов. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл logwgr.h и библиотечный файл logwgr.lib.

CLogBase - базовый класс регистрации. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл logwgr.h и библиотечный файл logwgr.lib.

CLogClient - класс для сеанса регистрационной базы данных и нахождения событий регистрации. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл logcli.h и библиотечный файл logcli.lib.

CLogEvent - класс для формирования деталей события регистрации. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл logwgr.h и библиотечный файл logwgr.lib.

CLogEventType - класс для сопоставления события регистрации с данными пользователя и другой информацией. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл logcli.h и библиотечный файл logcli.lib.

CLogFilter — класс для фильтрации регистрационных событий. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл logcli.h и библиотечный файл logcli.lib.

CLogFilterList - класс, список событий для фильтрации. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл logcli.h и библиотечный файл logcli.lib.

CLogView - класс для представления регистрационной базы данных. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл logview.h и библиотечный файл logcli.lib.

CLogViewDuplicate - класс для отображения дублирующих событий. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл logview.h и библиотечный файл logcli.lib.

CLogViewEvent - класс для отображения событий регистрации. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл logview.h и библиотечный файл logcli.lib.

CLogViewRecent - класс для отображения недавних событий регистрации. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл logview.h и библиотечный файл logcli.lib.

CLogWrapper - класс интерфейса регистрации. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл logwrap.h и библиотечный файл logwrap.lib.

TLogConfig - класс для создания конфигурации регистрационных данных. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл logcli.h и библиотечный файл logcli.lib.

TLogDuration - тип, время в секундах, определяющее продолжительность события регистрации. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл logwrap.h.

TLogDurationType — тип, определяющий тип продолжительности регистрации. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл logwrap.h.

TLogFlags - тип, содержащий флажки для случая регистрации. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл logwrap.h.

TLogId - тип, идентификатор события регистрации. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл logwrap.h.

TLogLink - тип, содержащий ссылку, связывающую случай регистрации с объектом другого приложения. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл logwrap.h.

TLogString - тип, изменяемый буферный дескриптор, в который помещается регистрационная строка. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл logwrap.h.

## **1.99. Maths Services**

Math - класс, содержащий набор математических функций. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32math.h и библиотечный файл euser.lib.

TReal196 - класс, расширенное точное значение. Доступен в Symbian OS с 5.0 по 6.0 версию. Для работы необходимо подключить заголовочный файл e32math.h и библиотечный файл euser.lib.

TRealX - класс, создающий расширенное точное значение. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32math.h и библиотечный файл euser.lib.

## **1.100. Media Server Common Classes**

CMdaServer - доступен в Symbian OS с 6.0 по 7.0 версию. Для работы необходимо подключить заголовочный файл mda\client\utility.h и библиотечный файл mediaclient.lib.

MMdaObjectEventListener - доступен в Symbian OS с 6.0 по 7.0 версию. Для работы необходимо подключить заголовочный файл mda\client\utility.h.

TMdaPackage - класс со структурой данных для упаковки сообщений, которые должны быть отправлены сервером клиенту. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл mda\com-mon\base.h.

TMdaRawPackage - внутренний класс, не предназначен для общего использования.

### **1.101. Memory Streams**

RBufReadStream - класс для открытия потока из динамического буфера. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл s32mem.h и библиотечный файл estor.lib.

RBufWriteStream - класс для записи потока из динамического буфера. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл s32mem.h и библиотечный файл estor.lib.

RDesReadStream — класс для открытия потока из дескриптора. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл s32mem.h и библиотечный файл estor.lib.

RDesWriteStream — класс для записи потока в потоковый буфер дескриптора. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл s32mem.h и библиотечный файл estor.lib.

RMemReadStream - класс для открытия и осуществления действий над потоком из памяти. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл s32mem.h и библиотечный файл estor.lib.

RMemWriteStream - класс для записи потока из памяти. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл s32mem.h и библиотечный файл estor.lib.

### **1.102. Message Scheduled Sending**

CMsvOffPeakTimes - класс, содержащий массив еженедельных непиковых периодов времени. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл MsvOffPeakTime.h и библиотечный файл schsend.lib.

CMsvScheduledEntries - тип, массив динамических объектов. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл MsvScheduledEntry.h.

CMsvScheduledEntry - класс, сохраняет список данных и получателей сообщения. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл MsvScheduledEntry.h и библиотечный файл schsend.lib.

CMsvScheduleSend - класс интерфейса планировщика передачи сообщений. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл MsvScheduleSend.h и библиотечный файл schsend.lib.

CMsvScheduleSettings - класс, сохраняющий параметры настройки планировщика. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл MsvScheduleSettings.h и библиотечный файл schsend.lib.

CMsvSendErrorActions - класс для определения действия на ошибку во время передачи сообщения. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл MsvSendErrorAction.h и библиотечный файл schsend.lib.

CMsvSysAgentActions - класс с массивом состояний системы, при которых допускается передача сообщения. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл msvsysagentaction.h и библиотечный файл schsend.lib.

CScheduleBaseServerMtm - базовый класс для сервера MTM. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл ScheduleBaseServerMtm.h и библиотечный файл schsend.lib.

SEND\_ERROR - структура ресурсов, которая определяет код ошибки. Доступна от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл schsend.rh.

SEND\_ERROR\_ACTION - структура ресурсов, определяющая действие, которое нужно выполнить если ошибка происходит во время передачи сообщения. Доступна от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл schsend.rh.

SEND\_ERROR\_ACTIONS - структура ресурсов, определяющая массив действий, которые нужно выполнить, если ошибка происходит во время передачи сообщения. Доступна от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл schsend.rh.

TMsvEntryScheduleData - класс, сохраняющий данные для планирования сообщения. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл MsvEntryScheduleData.h и библиотечный файл schsend.lib.

TMsvOffPeakTime — класс для определения непикового времени для отправки сообщения. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл MsvOffPeakTime.h и библиотечный файл schsend.lib.

TMsvSchedulePackage - класс, содержащий упакованную информацию запланированного сообщения. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл MsvSchedulePackage.h и библиотечный файл schsend.lib.

TMsvSendAction - перечисляемый тип, определяющий действие, которое должно произойти, если происходит ошибка при отсылке сообщения. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл schsend.h.

TMsvSendErrorAction - класс, назначает действие для ошибки. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл MsvSendErrorAction.h и библиотечный файл schsend.lib.

TMsvSendRetries - перечисляемый тип с флажками для определения поведения при повторных сбоях передачи сообщения. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл schsend.h.

TMsvSendRetrySpacing - перечисляемый тип для определения интервалов при повторном отправлении сообщения. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл schsend.h.

TMsvSysAgentConditionAction - класс для определения состояния системы, при которых допускается передача сообщения. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл msvsysagent-action.h и библиотечный файл schsend.lib.

### **1.103. Message Window**

RMessageWindow - класс, окно с настраиваемой конфигурацией, которое временно отображает сообщение и исчезает. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл clock.h и библиотечный файл clock.lib.

### **1.104. MMSMTM Client**

CMmsClientMtm - класс, интерфейс MTM для операций над MMS-сообщениями. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл MmsClientMtm.h и библиотечный файл MmsClientMtm.lib.

CMmsForwardOperation - класс для создания MMS-сообщения. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл MmsForwardOp.h и библиотечный файл MmsClientMtm.lib.

CMmsReplyOperation — класс для создания ответа на полученное MMS-сообщение. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл MmsReplyOp.h и библиотечный файл MmsClientMtm.lib.

TMmsFwdOpProgBuf - тип, пакетный буфер для структуры TMmsForwardOpProgress. Для работы необходимо подключить заголовочный файл MmsForwardOp.h.

TMmsForwardOpProgress - класс, содержащий данные для отображения прогресса операции передачи. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл MmsForwardOp.h.

TMmsMtmCmds - перечисляемый тип, содержащий команды для MMS MTM. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл MmsMtmCmds.h.

TMmsReplyOpProgBuf - тип, пакетный буфер для структуры TMmsReplyOpProgress. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл MmsReplyOp.h.

TMmsReplyOpProgress - класс, содержащий данные для отображения прогресса операции ответа. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл MmsReplyOp.h.

## 1.105. MMS Utilities

`CMmsClientMessage` - класс интерфейса для операций над MMS-сообщениями. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `MmsClientMessage.h` и библиотечный файл `MmsUtils.lib`.

`CMmsCodecMediaObjectCreator` - класс, MTM-кодировщик для сервера MMS, служащий для добавления входящих медиа-объектов к сообщению. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `MmsCodecMediaObjectCreator.h` и библиотечный файл `mmsutils.lib`.

`CMmsHeaders` - класс для создания имен и значений заголовков MMS. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `mmsheaders.h` и библиотечный файл `mmsutils.lib`.

`CMmsMediaObject` - класс для создания медиа-объекта MMS-сообщения. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `mmsmediaobject.h` и библиотечный файл `mmsutils.lib`.

`CMmsMediaObjectList` - класс для управления набором медиа-объектов. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `mmsmediaobjectlist.h` и библиотечный файл `mmsutils.lib`.

`CMmsMessage` - абстрактный класс, представляющий MMS-сообщение. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `MmsMessage.h` и библиотечный файл `mmsutils.lib`.

`CMmsRecipientArray` - тип с динамическим массивом `CMmsRecipient` объектов. Для работы необходимо подключить заголовочный файл `mmsheaders.h`.

`CMmsRecipient` - класс, обеспечивающий отчет о доставке MMS-сообщения получателю. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `MmsRecipient.h` и библиотечный файл `mmsutils.lib`.

`CMmsSettings` - класс, параметры настройки MMS. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `MmsSettings.h` и библиотечный файл `mmsutils.lib`.

`CMmsUtilsHeaderArray` - класс, содержащий набор заголовков. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `MmsUtilsHeaderArray.h` и библиотечный файл `mmsutils.lib`.

`CMmsUtilsHeaderParameter` - класс, параметры MMS-заголовка. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `MmsUtilsHeaderParameter.h` и библиотечный файл `mmsutils.lib`.

`CMmsUtilsHeaderParameterArray` — тип с динамическим массивом `CMmsUtilsHeaderParameter` объектов. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `MmsUtilsHeaderParameter.h`.

`MmsAddressParser` --класс для проверки достоверности MMS-адресов. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `MmsAddressParser.h` и библиотечный файл `mmsutils.lib`.



MmsUtils - класс, статические утилитные MMS-функции. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл MmsMessage.h и библиотечный файл mmsutils.lib.

TMmsMediaObjectId - тип для идентификации объекта в MMS-сообщении. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл MmsUtils.h.

TMmsMessageClass - перечисляемый тип, обеспечивающий класс, кодирующий значения для серверных компонентов. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл MmsUtils.h и библиотечный файл mmsutils.lib.

TMmsMessageMIMECategory - перечисляемый тип для идентификации категории MIME для MMS-сообщения. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл MmsUtils.h.

TMmsMessageType - перечисляемый тип для идентификации допустимых значений типа заголовка MMS. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл MmsUtils.h.

TMmsPriority — перечисляемый тип для идентификации допустимых значений заголовка приоритета MMS. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл MmsUtils.h.

TMmsResponseStatus - перечисляемый тип для идентификации допустимых значений заголовка ответа. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл MmsUtils.h.

TMmsSenderVisibility - перечисляемый тип для идентификации допустимых значений заголовка отправителя. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл MmsUtils.h.

TMmsStatus - перечисляемый тип для идентификации допустимых значений заголовка состояния. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл MmsUtils.h.

TMmsTime - класс, представляет значение времени в заголовке MMS. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл MmsUtils.h и библиотечный файл mmsutils.lib.

TMmsTransferProgress - класс, содержит данные для отображения прогресса от MMS-сервера или от операции выбора. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл MmsUtils.h и библиотечный файл mmsutils.lib.

TMmsVersion — класс, номер версии MMS. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл MmsUtils.h и библиотечный файл mmsutils.lib.

TMmsYesNo- перечисляемый тип для идентификации допустимых значений заголовка, которые принимают значения «Да» или «Нет». Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл MmsUtils.h.

TMsvMmsEntry - класс, добавляющий функциональные возможности типу MMS. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл MmsUtils.h и библиотечный файл mmsutils.lib.

## 1.106. MultiMediaCard

DMMCController - класс, контроллер карты MMC. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл p32mmc.h и библиотечный файл erbus.lib.

DMMCSession - класс для назначения и передачи команд стеку. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл p32mmc.h и библиотечный файл erbus.lib.

DMMCStack - класс для осуществления доступа к стеку MMC-карты. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл p32mmc.h и библиотечный файл erbus.lib.

TMMC - класс, набор сервисных функций. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл p32mmc.h.

TMMCAppCommand - перечисляемый тип, для определения команды MMC-карты. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл p32mmc.h.

TMMCArgument - класс, 32-битное значение. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл p32mmc.h и библиотечный файл erbus.lib.

TMMCBusConfig - класс, для настройки параметров шины. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл p32mmc.h.

TMMCCard - класс, содержащий контекстную информацию и специфику MMC-карты. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл p32mmc.h и библиотечный файл erbus.lib.

TMMCCardStateEnum - перечисляемый тип, определяющий возможные состояния MMC-карты. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл p32mmc.h.

TMMCCommandDesc - класс, параметры для команды, которая будет послана по шине, и ответ на нее. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл p32mmc.h.

TMMCCommandEnum - перечисляемый тип, содержащий символы, соответствующие командам шины MMC-карты. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл p32mmc.h.

TMMCCommandTypeEnum - перечисляемый тип, содержащий набор типов команд. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл p32mmc.h.

TMMCControllerType - перечисляемый тип для определения типа контроллера. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл p32mmc.h.

TMMCError - тип, набор значений ошибок MMC-карты закодированных как целое 32-битное число без знака. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл p32mmc.h.

TMMCMediaChange - класс для представления события смены MMC-карты. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл p32mmc.h и библиотечный файл erbus.lib.

TMMCMediaTypeEnum - перечисляемый тип для определения типа MMC-карты. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл p32mmc.h.

TMMCPsu - класс, PSU карты MMC. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл p32mmc.h и библиотечный файл erbus.lib.

TMMCResponseTypeEnum- перечисляемый тип, определяющий набор типов ответов на команды. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл p32mmc.h.

TMMCSessionTypeEnum- перечисляемый тип, определяющий тип сеанса. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл p32mmc.h.

TMMCStackConfig - класс, содержит набор параметров для настройки конфигурации стека. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл p32mmc.h.

TMMCStateMachine - класс, регулирующий состояние machine стека. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл p32mmc.h.

TMMCStatus - класс, определяющий статус MMC-карты, выраженный целым 32-битным числом без знака. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл p32mmc.h.

TCID - класс, содержащий идентификационный регистр карты (CID). Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл p32mmc.h и библиотечный файл erbus.lib.

TDSR - класс, регистр уровня устройства (DSR). Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл p32mmc.h и библиотечный файл erbus.lib.

TRCA - класс, релятивный адресный регистр карты (RCA). Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл p32mmc.h и библиотечный файл erbus.lib.

TCSD- класс, регистр данных карты (CDR). Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл p32mmc.h и библиотечный файл erbus.lib.

SMF\_NEXTS - макрос для назначения состояния следующего за текущим в nexts. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл p32mmc.h.

SMF\_CALL - макрос для вызова дочерней функции func и перехода к следующему состоянию. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл p32mmc.h.

SMF\_CALLWAIT - макрос для вызова дочерней функции func (но при этом ожидает в точке входа) и перехода к следующему состоянию. Доступен от версии

Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `p32mmc.h`.

`SMF_CALLMYS` - макрос для вызова текущей функции в точке входа `nexpts`. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `p32mmc.h`.

`SMF_CALLMEWR` - макрос для собственного вызова с возвращением состояния `retst`. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `p32mmc.h`.

`SMF_INVOKES` - макрос для вызова дочерней функции `f unc` и перехода к состоянию `nexpts`. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `p32mmc.h`.

`SMF_INVOKEWAITS` - макрос для вызова дочерней функции `f unc` (но при этом ожидает в точке входа) и перехода к состоянию `nexpts`. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `p32mmc.h`.

`SMF_WAIT` - макрос для ожидания при переходе в следующее состояние. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `p32mmc.h`.

`SMF_WAITS` - макрос для установления следующего состояния в `nexpts` и последующего ожидания. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `p32mmc.h`.

`SMF_RETURN` - макрос для возвращения ошибки вызываемой функции. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `p32mmc.h`.

`SMF_EXIT` - макрос для возвращения к вызываемой функции. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `p32mmc.h`.

`SMF_EXITWAIT` - макрос для возвращения к вызываемой функции, но ожидает в точке выхода. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `p32mmc.h`.

`SMF_JUMP` - макрос для передачи управления функции `f unc`. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `p32mmc.h`.

`SMF_JUMPWAIT` - макрос для передачи управления функции `f unc`, но ожидает в точке входа. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `p32mmc.h`.

`SMF_GOTONEXTS` - макрос для перехода в следующее состояние. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `p32mmc.h`.

`SMF_GOTOS` - макрос для установки следующего состояния в `nexpts` и для перехода в него. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `p32mmc.h`.

`SMF_STATE` — макрос для объявления имени состояния `sname`. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `p32mmc.h`.

SMF\_BPOINT - макрос для объявления состояния sname и переходит в режим «спячки», если оно статически заблокировано. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл p32mmc.h.

### **1.107. NetDial**

TNetDialProgress - перечисляемый тип для определения стадий подключения с дозвоном по сети. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл connectprog.h.

### **1.108. Notification Services**

RChangeNotifier - класс, дескриптор к изменению уведомителя. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

RNotifier - класс, дескриптор к сеансу сервера уведомления, который обеспечивает дополнение к программе уведомления. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

RUndertaker - класс, дескриптор к закрывающемуся потоку уведомления. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

TChanges - перечисляемый тип для определения происходящих изменений, о которых может быть сообщено измененным уведомлением. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

### **1.109. Onboard Camera**

CCamera - базовый класс для устройства камеры. Доступен от версии Symbian OS 7.0s. Для работы необходимо подключить заголовочный файл ecam.h и библиотечный файл ecam.lib.

MCameraObserver - базовый класс для клиентов камеры. Для работы необходимо подключить заголовочный файл ecam.h и библиотечный файл ecam.lib.

MFrameBuffer - класс буфера для передачи видеок кадров от камеры клиенту. Для работы необходимо подключить заголовочный файл ecam.h и библиотечный файл ecam.lib.

TCameraInfo - класс, содержащий информацию об устройстве камеры. Для работы необходимо подключить заголовочный файл ecam.h и библиотечный файл ecam.lib.

### **1.110. Open Font System**

SOpenFont - класс системы шрифтов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл openfont.h и библиотечный файл fntstr.lib.

COpenFontFile - абстрактный базовый класс шрифтов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл openfont.h и библиотечный файл fntstr.lib.

COpenFontRasterizer - абстрактный класс растеризации шрифтов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл openfont.h и библиотечный файл fntstr.lib.

COpenFontRasterizerContext - класс, из которого могут быть получены контексты растеризации. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл openfont.h и библиотечный файл fntstr.lib.

TOpenFontCharMetri.es - класс, определяющий метрику шрифта. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл openfont.h и библиотечный файл fntstr.lib.

TOpenFontFaceAttribBase - внутренний класс, не предназначен для общего использования.

TOpenFontFaceAttrib - класс, определяющий атрибуты шрифта. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл openfont.h.

TOpenFontGlyphData — класс, определяющий данные глифа шрифта. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл openfont.h и библиотечный файл fntstr.lib.

TOpenFontMetri.es - класс, определяющий метрику шрифта. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл openfont.h и библиотечный файл fntstr.lib.

TOpenFontSpec - класс, содержащий спецификацию шрифта. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл openfont.h и библиотечный файл fntstr.lib.

### **1.111. Package Buffers**

TPckgBuf - класс для упаковки объекта в модифицируемый буферный дескриптор. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

TPckgC - класс для упаковки немодифицируемого дескриптора. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

TPckg - класс для упаковки модифицируемого дескриптора. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

### **1.112. PC Connect Device-side BAL**

BAL - класс интерфейса для абстрактного уровня однонаправленного канала. Доступен от версии Symbian OS 8.0. Для работы необходимо подключить заголовочный файл BALClient.h и библиотечный файл BALClient.lib.

### **1.113. PhoneBook Synchroniser**

RPhoneBookSession - класс для обеспечения обращения класса CPhoneBookSyncPlugin к сеансу сервера синхронизации телефонной книги. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл phbksync.h и библиотечный файл phbksyncsvr.lib.

TContactFieldFormat - класс для сохранения информации из контактных полей. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл phbksync.h и библиотечный файл phbksyncsvr.lib.

TPhonebookSyncRequestCancel - перечисляемый тип для отмены запроса синхронизации. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл phbksync.h.

### **1.114. Power management framework**

DPowerHandler - класс для обеспечения средств управления питанием устройства. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл k32power.h и библиотечный файл ekern.lib.

DPowerModel - класс для поддержки обработчиков питания и проверки каждого их обращения к питанию. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл k32power.h и библиотечный файл ekern.lib.

Power - класс, интерфейс структуры управления питанием. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл k32power.h и библиотечный файл ekern.lib.

TPowerState - перечисляемый тип для определения состояния обработчика питания. Для работы необходимо подключить заголовочный файл k32power.h.

### **1.115. Print Framework**

CHeaderFooter — класс, определяющий заголовок документа или нижний колонтитул. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл prninf.h и библиотечный файл print.lib.

CPrintSetup - класс, отображающий предпочтательную информацию. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл prnsetup.h и библиотечный файл print.lib.

MPrintProcessObserver - класс, интерфейс для отображения прогресса печати. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл prninf.h.

TPageMargins - класс, определяющий поля страницы. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл prninf.h и библиотечный файл print.lib.

TPrintParameters — класс, определяющий параметры печати. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл prninf.h и библиотечный файл print.lib.

### 1.116. Print Preview

CPr intPreviewImage — класс для предварительного просмотра перед печатью. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `mpprev.h` и библиотечный файл `prev.lib`.

TPrintPreviewFormat - класс, содержащий информацию предварительного просмотра. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `mpprev.h`.

InternalizeL () - функция, принимающая поля страницы из потока чтения. Доступна от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `mpprev.h`.

### 1.117. Printing

CPDrModelList - класс интерфейса для модели списка файлов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `gdi.h` и библиотечный файл `gdi.lib`.

CPrinterControl - класс, интерфейс управления принтером. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `gdi.h` и библиотечный файл `gdi.lib`.

CPrinterDevice - класс, интерфейс графического устройства принтера. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `gdi.h` и библиотечный файл `gdi.lib`.

CPrinterDriver - класс, обеспечивающий доступ к памяти, содержащей данные принтера. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `gdi.h` и библиотечный файл `gdi.lib`.

CPrinterDriverUI - класс, пользовательский интерфейс печатающего устройства. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `gdi.h` и библиотечный файл `gdi.lib`.

CPrinterModelList - класс, список моделей принтеров. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `gdi.h`.

CPrinterPort - класс интерфейса порта принтера. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `gdi.h`.

MPageRegionPrinter - класс интерфейса для печати в виде колонок. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `gdi.h`.

TBandAttributes - класс, атрибуты колонки. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `gdi.h`.

TPageSpec - класс, содержащий спецификацию страницы для печати. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `gdi.h` и библиотечный файл `gdi.lib`.

TPrinterModelEntry - класс, содержащий подробную информацию о модели принтера. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `gdi.h` и библиотечный файл `gdi.lib`.



TPrinterModelHeader - класс, содержащий информацию о модели принтера. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл gdi.h и библиотечный файл gdi.lib.

TPrinterModelName — тип, модифицируемый буферный дескриптор, содержащий имя модели принтера. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл gdi.h.

### **1.118. Raw Memory**

Mem - класс, содержащий статические функции для операций с данными в памяти. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

### **1.119. Recognizers**

CAraAppLocator — внутренний класс, не предназначен для общего использования.

CAraAppLocatorProху - внутренний класс, не предназначен для общего использования.

CAraDataRecognizer - внутренний класс, не предназначен для общего использования.

CAraDataRecognizerType - абстрактный класс для устройства распознавания. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл armrec.h и библиотечный файл armime.lib.

CAraFileRecognizer - внутренний класс, не предназначен для общего использования.

CAraFileRecognizerType - внутренний класс, не предназначен для общего использования.

CAraRecognizerDll — внутренний класс, не предназначен для общего использования.

CAraScanningDataRecognizer - внутренний класс, не предназначен для общего использования.

CAraScanningFileRecognizer - внутренний класс, не предназначен для общего использования.

MAraAppStarter — внутренний класс, не предназначен для общего использования.

TDataRecognitionResult - класс, содержащий результаты распознавания данных. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл armrec.h и библиотечный файл armime.lib.

TDataType - класс, тип данных. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл armstd.h и библиотечный файл armime.lib.

TDataTypePriority - класс для определения приоритета типов данных. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл armstd.h.

TDataTypeWithPriority-класс, содержащий тип данных и приоритет этого типа. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл armstd.h и библиотечный файл armmime.lib.

### **1.120. Reference counting objects**

CObject- класс, выполняющий расчет ссылок для отслеживания параллельных ссылок на себя. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32base.h и библиотечный файл euser.lib.

CObjectCon— класс, контейнер для объектов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32base.h и библиотечный файл euser.lib.

CObjectConlx - класс, контейнер для контейнеров объектов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32base.h и библиотечный файл euser.lib.

CObjectCntlx - класс, создает номера для дескрипторов для подсчета обращений объектов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32base.h и библиотечный файл euser.lib.

CObjectCntlxRec - внутренняя структура, не предназначена для общего использования.

TFullName - тип для определения изменяемого буферного дескриптора, который содержит полное имя ссылающегося объекта. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

TName - тип для определения изменяемого буферного дескриптора, который содержит имя ссылающегося объекта. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

### **1.121. Security**

CBoundedSecurityBase - класс для создания дополнительной модели защиты, которая расширяет интерфейс. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32base.h и библиотечный файл euser.lib.

CSecurityBase - класс интерфейса для модели защиты. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32base.h и библиотечный файл euser.lib.

CSecurityDecryptBase - класс интерфейса к дескриптору, который создает алгоритм для расшифровки данных. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32base.h и библиотечный файл euser.lib.

CSecurityEncryptBase - класс интерфейса к дескриптору, который создает алгоритм для кодирования данных. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32base.h и библиотечный файл euser.lib.

Password - класс, содержащий набор функций для обработки паролей. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

TPassword - тип, буфер для хранения пароля. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

### **1.122. Semaphores**

RSemaphore - класс, содержащий дескриптор на семафор. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

TFindSemaphore - класс для нахождения семафоров по именам, которые соответствуют установленному образцу. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

### **1.123. Serial Protocol Module**

CPort - класс для выполнения последовательных протокольных портов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл cs\_port.h и библиотечный файл c32.lib.

CSerial - класс, отвечающий за создание классов, получаемых из класса CPort. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл cs\_port.h и библиотечный файл c32.lib.

TSerialNewL - тип для определения первой функции в точке входа к последовательному модулю протокола. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл cs\_port.h.

### **1.124. SIM Application Toolkit**

RSat - класс, обеспечивающий доступ к классу 2+ GSM, SIM Toolkit. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл Etelsat.h и библиотечный файл etelsat.lib.

ESatIpcOf f set - внутренний элемент, не предназначен для общего использования.

IPC\_SAT\_EXT - внутренний элемент, не предназначен для общего использования.

ETELSAT\_H\_ - макрос, подлежит определению, если подключен файл Etelsat.h. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл Etelsat.h.

\_\_SATCS\_H\_ - макрос, подлежит определению, если подключен файл satcs.h. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл satcs.h.

## 1.125. SMS GSM Utilities

CGsmuBackupObserver - внутренний класс, не предназначен для общего использования.

CSARStore - класс для управления сегментацией и повторной сборкой памяти. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл gsmustor.h и библиотечный файл gsmu.lib.

CSmsAddress — внутренний класс, не предназначен для общего использования.

CSmsAlphabetConverter - класс с утилитой для преобразования набора символов для/от распакованных данных пользователя. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл Gsmuelem.h и библиотечный файл gsmu.lib.

CSmsBuf fer - класс для сохранения буфера в массиве TText. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл gsmubuf.h и библиотечный файл gsmu.lib.

CSmsBuf ferBase - базовый класс буферов SMS. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл gsmubuf.h и библиотечный файл gsmu.lib.

CSmsEditorBuf fer - класс, буфер для SMS. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл gsmubuf.h и библиотечный файл gsmu.lib.

CSmsInformationElement - класс, сохраняющий элемент SMS. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл Gsmuelem.h и библиотечный файл gsmu.lib.

CSmsMessage - класс, представляющий законченное SMS-сообщение. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл Gsmumsg.h и библиотечный файл gsmu.lib.

CSmsPDU- класс, интерфейс GSM SMS PDU. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл gsmupdu.h и библиотечный файл gsmu.lib.

CSmsUserData - класс для операций с пользовательскими данными, описанными в TP-UD. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл Gsmuelem.h и библиотечный файл gsmu.lib.

KSARStoreUid - константа для хранения второго UID в SAR памяти. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл gsmustor.h.

TGsmSms - основной класс SMS, который содержит мультитип SMS. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл gsmuetel.h и библиотечный файл gsmu.lib.

TGsmSmsNumberingPlanIdentification - перечисляемый тип, нумерация, определенная в ETSI 3GPP TS 23.040. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл Gsmuelem.h.

TGsmSmsTelNumber - класс для создания пакета основной адресной информации. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл Gsmuelem.h и библиотечный файл gsmu.lib.

TGsmSmsTypeOf Address - класс, заполняемые адресные поля, определенные в ETSI 3GPP TS 23.040. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл Gsmuelem.h и библиотечный файл gsmu.lib.

TGsmSmsTypeOf Number - перечисляемый тип для хранения типов номеров как определено в ETSI 3GPP TS 23.040. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл Gsmuelem.h.

TGsmuLex8 — класс для обеспечения уходящих функций при получении следующего символа. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл Gsmuelem.h и библиотечный файл gsmu.lib.

TSAREntry - класс, информация для сегментации и повторной сборки SMS-сообщений или WAP-датаграмм. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл gsmustor.h и библиотечный файл gsmu.lib.

TSmsCommandType - класс, содержащий перечисления и операции для SMS-команд. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл Gsmuelem.h и библиотечный файл gsmu.lib.

TSmsDataCodingScheme - класс, TP-DCS, кодирующие схемы данных определенные в 3GPP TS 23.038. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл Gsmuelem.h и библиотечный файл gsmu.lib.

TSmsFailureCause - класс, TP-FCP. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл Gsmuelem.h и библиотечный файл gsmu.lib.

TSmsFirstOctet - класс, первый PDU сообщения. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл Gsmuelem.h и библиотечный файл gsmu.lib.

TSmsOctet - базовый класс для всех операций. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл Gsmuelem.h и библиотечный файл gsmu.lib.

TSmsParameterIndicator — класс, TP-PI, найденный в отчетах и командах. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл Gsmuelem.h и библиотечный файл gsmu.lib.

TSmsProtocolIdentifier - класс, TP-PID PDU. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл Gsmuelem.h и библиотечный файл gsmu.lib.

TSmsReassemblyEntry - класс, для сохранения части готового SMS, которое может быть применено в следующем сообщении. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл gsmustor.h и библиотечный файл gsmu.lib.

TSmsReportResponsibility - перечисляемый тип, отвечает за отчет о доставке сообщения. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл gsmuettel.h.

TSmsReportSetting - структура, содержащая флажки отчетов о доставке сообщения. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл gsmuetel.h.

TSmsSegmentationEntry - класс для сохранения части исходящего SMS, которое может быть использовано в следующем сообщении. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл gsmustor.h и библиотечный файл gsmu.lib.

TSmsServiceCenterTimeStamp - класс, сервисная временная служба TP-SCTS. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл Gsmuelem.h и библиотечный файл gsmu.lib.

TSmsStatus- класс, TP-ST октет. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл Gsmuelem.h и библиотечный файл gsmu.lib.

TSmsUserDataSettings - класс для выполнения операций на TP-UD пользовательских данных. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл gsmuset.h и библиотечный файл gsmu.lib.

TSmsValidityPeriod - класс, TP-VT период действия, найденный в SUBMIT PDU. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл Gsmuelem.h и библиотечный файл gsmu.lib.

TWapReassemblyEntry- класс для повторной сборки WAP-сообщения, формирующей конечное сообщение. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл gsmustor.h и библиотечный файл gsmu.lib.

### **1.126. SMS Utilities**

CSmsEventLogger — класс, регистрирующий события связанные с SMS. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл smsulog.h и библиотечный файл smsu.lib.

CSmsuActiveBase - базовый класс для активных объектов SMS. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл smsuact.h и библиотечный файл smsu.lib.

RSmsSocketBuf - класс, потоковый SMS-буфер для записи и чтения из со-кета. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл smsustrm.h и библиотечный файл smsu.lib.

RSmsSocketReadStream - класс, содержит поток, считывающий CSmsMessage объект через сокет. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл smsustrm.h.

RSmsSocketWriteStream - класс, содержит поток, записывающий CSmsMessage объект через сокет. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл smsustrm.h и библиотечный файл smsu.lib.

TSmsAddr - класс, SMS-адрес для сокета. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл smsuaddr.h и библиотечный файл smsu.lib.

TSmsAddrFamily - перечисляемый тип, содержащий набор SMS-адресов для сокета. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл smsuaddr.h.

TSmsServiceCenterAddress - тип, буфер для хранения адресной сервисной службы. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл smsuaddr.h.

TSmsServiceCenterAddressBuf - тип, пакетный буфер для TSmsServiceCenterAddress объектов. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл smsuaddr.h.

TSmsSettings - класс, параметры настройки SMS. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл smsuset.h и библиотечный файл smsu.lib.

### **1.127. Sockets Client**

BigEndian - класс, содержит вставленные и извлеченные целые числа большого формата. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл es\_sock.h и библиотечный файл esock.lib.

ByteOrder - класс, изменяет порядок байт в 16- и 32-битных значениях. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл es\_sock.h и библиотечный файл esock.lib.

LittleEndian - класс, вставленные и извлеченные целые числа небольшого формата. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл es\_sock.h и библиотечный файл esock.lib.

RNetDatabase - класс, интерфейс для сетевых баз данных. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл es\_sock.h и библиотечный файл esock.lib.

RServiceResolver - класс, интерфейс для сервисных номеров и портов. Для работы необходимо подключить заголовочный файл ES\_SOCKET.h.

RSocket - класс для обеспечения конечной клиентской точки протокола. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл es\_sock.h и библиотечный файл esock.lib.

RSocketServ - класс, обеспечивающий Connect () функцию для создания канала связи ICP. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл es\_sock.h и библиотечный файл esock.lib.

SSockAddr - структура, представляющая Socket address. Доступна от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл es\_sock.h.

TByteOrder - перечисляемый тип, применяемый в структуре TProtocolDesc для описания протокола. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл es\_sock.h.

TESockFault - перечисляемый тип, содержащий номера паники, связанные с eSock Fault категорией. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл ss\_std.h.

TESockPanic - перечисляемый тип, содержащий номера паники, связанные с eSock категорией. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл ss\_std.h.

THostName - тип, определяющий дескриптор, содержащий строку Host name. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл es\_sock.h.

TNameEntry - тип для упаковки TNameRecord класса. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл es\_sock.h.

TNameRecord - класс, содержащий результаты запросов имени. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл es\_sock.h и библиотечный файл esock.lib.

TProtocolDesc - структура, содержащая протокольную информацию соке-та. Доступна от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл es\_sock.h.

TProtocolName - тип, содержит имя протокола в структуре TProtocolDesc. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл es\_sock.h и библиотечный файл esock.lib.

TServiceName - тип, определяющий дескриптор, содержащий сервисную строку с именем. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл es\_sock.h.

TSockAddr — класс, конечная точка адреса. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл es\_sock.h и библиотечный файл esock.lib.

TSockXf rLength - тип, используемый RSocket при чтении и записи для передачи длины читаемых и записываемых данных. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл es\_sock.h.

## **1.128. Sound Device**

CDevSoundError - внутренний класс, не предназначен для общего использования.

CDevSoundPlayer - внутренний класс, не предназначен для общего использования.

CDevSoundRecordError - внутренний класс, не предназначен для общего использования.

CDevSoundRecorder - внутренний класс, не предназначен для общего использования.

CMMFDevSound - класс интерфейса для необработанных аудиофункций. Доступен от версии Symbian OS 7.0s. Для работы необходимо подключить заголовочный файл SoundDevice.h и библиотечный файл MMFDevSound.lib.

MDevSoundObserver - класс интерфейса для DevSound-функций. Доступен от версии Symbian OS 7.0s. Для работы необходимо подключить заголовочный файл SoundDevice.h и библиотечный файл MMFDevSound.lib.



TMMFCapabilities — класс, содержащий текущую частоту дискретизации, кодирование, моно/стерео индикатор и размеры буфера для DevSound. Доступен от версии Symbian OS 7.0s. Для работы необходимо подключить заголовочный файл SoundDevice.h и библиотечный файл MMFDevSound.lib.

TMMFMonoStereo - перечисляемый тип, содержащий аудиоиндикатор (мо-но или стерео). Доступен от версии Symbian OS 7.0s. Для работы необходимо подключить заголовочный файл SoundDevice.h и библиотечный файл MMFDevSound.lib.

TMMFSampleRate - перечисляемый тип, содержащий доступные частоты дискретизации. Доступен от версии Symbian OS 7.0s. Для работы необходимо подключить заголовочный файл SoundDevice.h и библиотечный файл MMFDevSound.lib.

TMMFSoundEncoding - перечисляемый тип, содержащий доступные режимы кодирования. Доступен от версии Symbian OS 7.0s. Для работы необходимо подключить заголовочный файл SoundDevice.h и библиотечный файл MMFDevSound.lib.

TMMFStereoSupport - перечисляемый тип, чередующаяся/нечередующаяся поддержка стерео. Доступен от версии Symbian OS 7.0s. Для работы необходимо подключить заголовочный файл SoundDevice.h и библиотечный файл MMFDevSound.lib.

TMdaPtr8 - внутренний класс, не предназначен для общего использования.

## 1.129. Stores

CBuf Store - класс, определяет несохраняемую оперативную память. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл s32mem.h и библиотечный файл estor.lib.

CDictionaryStore — класс, интерфейс для памяти словаря. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл s32stor.h и библиотечный файл estor.lib.

CEmbeddedStore - класс для создания встраиваемой памяти. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл s32stor.h и библиотечный файл estor.lib.

CPersistentStore - класс постоянной памяти. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл s32stor.h и библиотечный файл estor.lib.

CStreamStore - класс для обеспечения абстрактной структуры памяти для управления потоками. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл s32stor.h и библиотечный файл estor.lib.

MIncrementalCollector - класс, интерфейс для исправления и уплотнения пространства в потоковой памяти. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл s32stor.h.

RDictionaryReadStream - класс для управления и открытия потока памяти словаря. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл s32stor.h и библиотечный файл estor.lib.

`RDictionaryWriteStream` - класс, поддерживающий создание и изменение потока памяти словаря. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `s32stor.h` и библиотечный файл `estor.lib`.

`RStoreReclaim` - класс для исправления и уплотнения пространства в постоянной файловой памяти. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `s32stor.h` и библиотечный файл `estor.lib`.

### **1.130. Store Streams**

`CStreamDictionary` - класс для поддержки ассоциации между UID и потоковым идентификатором. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `s32std.h` и библиотечный файл `estor.lib`.

`MEExternalizer` - внутренний класс, не предназначен для общего использования.

`RStoreReadStream` - класс для управления и открытия потока памяти. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `s32std.h` и библиотечный файл `estor.lib`.

`RStoreWriteStream` - класс для записи потока в память. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `s32std.h` и библиотечный файл `estor.lib`.

`TStreamId` - класс для назначения уникальных идентификаторов для потока в пределах памяти. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `s32std.h` и библиотечный файл `estor.lib`.

`TSwizzleC` — класс, обеспечивающий двойное представление константного объекта. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `s32std.h` и библиотечный файл `estor.lib`.

`TSwizzle` - класс, обеспечивающий двойное представление объекта. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `s32std.h` и библиотечный файл `estor.lib`.

`TSwizzle<TAny>` - класс для реализации семейства классов `TSwizzle<class T>`. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `s32std.h` и библиотечный файл `estor.lib`.

`TSwizzleC<TAny>` - класс для реализации семейства классов `TSwizzleC<class T>`. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `s32std.h` и библиотечный файл `estor.lib`.

### **1.131. System Agent**

`RSaVarChangeNotif` у - класс интерфейса системного агента для уведомлений об изменениях состояния переменной. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `savarset.h` и библиотечный файл `sysagt.lib`.

RSystemAgentBase - класс для клиентской обработки системного агента. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл saclient.h и библиотечный файл sysagt.lib.

RSystemAgent - класс интерфейса для обработки системного агента. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл saclient.h и библиотечный файл sysagt.lib.

### 1.132. System Sounds

BaSystemSound - сервисный класс для обработки системных звуков. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл bassnd.h и библиотечный файл bafl.lib.

CoeSoundPlayer - сервисный класс для проигрывания простых звуков. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл coesndpy.h и библиотечный файл bafl.lib.

TBaSystemSoundInfo - класс, обеспечивающий системные функции, которые содержат информацию для системного звука. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл bassnd.h и библиотечный файл bafl.lib.

TBaSystemSoundName — тип для определения имени файла системного звука. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл bassnd.h и библиотечный файл bafl.lib.

TBaSystemSoundType - класс для обеспечения логической звуковой оболочки. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл bassnd.h и библиотечный файл bafl.lib.

TBaSystemSoundUid - тип для определения системного звука для звонка, будильника, сообщения, ошибки или случая. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл bassnd.h и библиотечный файл bafl.lib.

### 1.133. TCP/IP

INET\_ADDR - создает целый 32-битный IPv4-адрес. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл in\_sock.h.

TIfStatus - перечисляемый тип для описания состояния интерфейса. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл in\_sock.h.

TInetAddr - класс, содержащий IP-адрес. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл insock.h и библиотечный файл insocLib.

TInetIfConfig - класс для описания направляющей IP-опции. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл in\_iface.h.

TInterfaceName - тип, содержит имя сетевого интерфейса. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл in\_iface.h.

TPrbAddr - класс, содержит необработанный 128-битный IPv6-адрес. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл in\_sock.h и библиотечный файл insock.lib.

TNameRecordFlags - перечисляемый тип, обеспечивающий флажки для описания результата DNS запроса. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл in\_sock.h.

TRouteState - перечисляемый тип для идентификации поддерживаемого пути. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл in\_sock.h.

TRouteType - перечисляемый тип для идентификации типа создания входа в таблицу IP-маршрутов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл in\_sock.h.

TSoIfConfigBase - базовый класс для TSoInetIfConfig, который идентифицирует семейство протоколов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл in\_iface.h.

TSoIfHardwareAddr - класс, локальный аппаратный адрес интерфейса. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл IN\_IFACE.h.

TSoIfInfo - класс, параметры операции сетевого интерфейса. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл in\_iface.h.

TSoInet6InterfaceInfo - класс информации интерфейса. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл in\_sock.h.

TSoInetCachedRouteInfo - класс, содержащий информацию, которая описывает кэшируемый вход. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл in\_sock.h и библиотечный файл insock.lib.

TSoInetIfConfig - класс для описания интерфейса, который адресует конфигурацию. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл IN\_IFACE.h.

TSoInetIfQuery - класс, содержащий информацию для запроса интерфейса. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл in\_sock.h и библиотечный файл insock.lib.

TSoInetInterfaceInfo - класс, содержащий информацию для описания образца сетевого интерфейса. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл in\_sock.h и библиотечный файл insock.lib.

TSoInetLastError - класс, содержащий сведения об ошибке для TCP/IP-протоколов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл in\_sock.h и библиотечный файл insock.lib.

TSoInetRouteInfo - класс, создает информацию, описывающую вход в IP-таблицу маршрутов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл in\_sock.h и библиотечный файл insock.lib.

### **1.134. Test Console**

RTest - класс, создает консольное окно для тестирования. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32test.h и библиотечный файл euser.lib.

### **1.135. Text and Text Attributes**

CCharFormatLayer - класс, уровень символического форматирования. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл txtfmlyr.h и библиотечный файл etext.lib.

CDateTimeField - класс для сохранения поля даты или времени. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл fldbltin.h и библиотечный файл field.lib.

CEditableText - абстрактный класс для определения поведения классов редактирования. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл txtetext.h и библиотечный файл etext.lib.

CFileNameField - класс, содержащий поле для имени файла. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл fldbltin.h и библиотечный файл field.lib.

CFormatLayer — абстрактный класс для атрибутов параграфа и уровня символического редактирования. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл txtfmlyr.h и библиотечный файл etext.lib.

CGlobalText - класс, содержащий текст с глобальным форматированием. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл txtglobl.h и библиотечный файл etext.lib.

CNumPagesField - класс, содержащий поле, которое определяет страницы в документе. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл fldbltin.h и библиотечный файл field.lib.

CPageFieldBase - класс, сохраняющий стиль для отображения значений полей при преобразованиях. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл fldbltin.h и библиотечный файл field.lib.

CPageNumField - класс, содержащий поле, которое определяет текущий номер страницы в документе. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл fldbltin.h и библиотечный файл field.lib.

CParaFormat - класс, сохраняющий атрибуты формата параграфа, табуляции и другое форматирование. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл txtfmat.h и библиотечный файл etext.lib.

CParaFormatLayer - класс, уровень форматирования параграфов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл txtfmlyr.h и библиотечный файл etext.lib.

CParagraphStyle - класс для определения стиля параграфа. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл txtstyle.h и библиотечный файл etext.lib.

CPlainText - класс для сохранения и действий над открытым текстом. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл txtetext.h и библиотечный файл etext.lib.

CRichText - класс, текст с различным форматированием. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл txtrich.h и библиотечный файл etext.lib.

CStyleList - класс, контейнер для стилей параграфа. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл txtstyle.h и библиотечный файл etext.lib.

CTextField - абстрактный класс для всех типов полей. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл fldbase.h и библиотечный файл field.lib.

MEditObserver - класс, определяющий протокол для обозревателя текста с различным форматированием. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл medobsrv.h.

MFieldFileNameInfo - класс, устанавливающий протокол для определения поля имени файла. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл fldbltin.h.

MFieldNumPagesInfo - класс, устанавливающий протокол для определения общего количества полей страницы. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл fldbltin.h.

MFieldPageNumInfo - класс, устанавливающий протокол для определения текущего поля страницы. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл fldbltin.h.

MFormatText - класс, определяющий протокол для получения и установки символа и атрибутов параграфа. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл txtmfmtx.h.

MLayDoc - класс, интерфейс для обеспечения информацией. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл txtlaydc.h.

MTextFieldFactory - абстрактный класс, допускающий использование полей в доступном для редактирования тексте. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл fldbase.h.

RParagraphStyleInfo - класс, набор имен символьных атрибутов форматирования. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл txtstyle.h.

TBullet - класс, точки маркеров. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл txtformat.h и библиотечный файл etext.lib.

TCharFormat - класс, контейнер символьных атрибутов форматирования. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл txtformat.h и библиотечный файл etext.lib.

TCharFormatMask - класс, маски символьных атрибутов форматирования. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл txtformat.h и библиотечный файл etext.lib.

TEtextComponentInfolf о - класс, обеспечивающий информацию о числе компонентов доступных для редактирования. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл txtetext.h и библиотечный файл etext.lib.

TFindFieldInfo - класс, получающий информацию о найденных полях в диапазоне символов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл fldinfah и библиотечный файл field.lib.

TFontPresentation — класс для определения атрибутов форматирования шрифта. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл txtformat.h и библиотечный файл etext.lib.

TLogicalRgb - класс, обеспечивающий поддержку системных цветов в дополнение к литеральным. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл txtformat.h и библиотечный файл etext.lib.

TPageTable - тип, таблица страниц, содержащая массив целых чисел. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл txtetext.h.

TParaBorderArray - класс для хранения четырех сторон границ параграфа. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл txtformat.h.

TParaBorder - класс для определения характеристик одной из четырех границ параграфа. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл txtformat.h и библиотечный файл etext.lib.

TParaFormatMask - класс, заносающий атрибуты форматирования параграфа в маску. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл txtformat.h и библиотечный файл etext.lib.

TParagraphStyleName - тип, содержащий имя стиля параграфа. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл txtstyle.h.

TTabStop — класс, табуляция для выравнивания строк в тексте. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл txtformat.h и библиотечный файл etext.lib.

TTextFormatAttribute - перечисляемый тип для идентификации установленных в тексте атрибутов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл txtformat.h.

### **1.136. Text Views**

CLayoutData - класс, определяет данные разметки. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл frmlyadt.h.

CTextLayout - класс, текстовая разметка. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл frmtdlay.h и библиотечный файл form.lib.

CTextPaginator - класс для разбивки документа на страницы. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл frmpage.h и библиотечный файл form.lib.

CTextView - класс, размещает отформатированный текст на дисплее. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл frmview.h и библиотечный файл form.lib.

MFormCustomDraw - абстрактный класс, определяющий протокол для настройки прорисовывания текста и заднего фона. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл frmtdraw.h.

MFormCustomWrap - класс интерфейса для осуществления принудительного выравнивания текста. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл frmtdraw.h и библиотечный файл form.lib.

MFormParam - класс, определяющий протокол для получения системных цветов при представлении текста. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл frmparam.h.

MPaginateObserver — абстрактный класс, определяющий протокол для назначения обозревателя процесса разбиения на страницы. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл frmpage.h.

TCursor - класс, определяющий видимость курсора. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл frmview.h.

TCursorPosition - класс, определяющий позицию курсора. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл frmtdraw.h.

TCursorSelection - класс для выделения курсором в пределах документа. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл frmtdraw.h.

TDrawTextLayoutContext - класс, содержащий параметры, которые используются функциями для рисования текста. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл frmtdraw.h и библиотечный файл form.lib.

TFormPanic - внутренний перечисляемый тип, не предназначен для общего использования.

TFrameOverlay - класс, определяющий восемь прямоугольных квадратов, окружающих изображение для изменения его размера. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл frmframe.h и библиотечный файл form.lib.

TNonPrintingCharVisibility - класс с набором флажков для отображения, какие из непечатаемых символов должны быть отображены. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл frmvis.h и библиотечный файл form.lib.

TTmCursorPlacement - перечисляемый тип, размещение курсора в тексте. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл tagmah.

TViewRectChanges - класс, результаты операции переформатирования. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл frmtdraw.h.



### **1.137. Timers and Timing Services**

`CDeltaTimer` - класс, выстраивающий очередь временных событий. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `e32std.h` и библиотечный файл `euser.lib`.

`CHeartbeat` - класс, устанавливающий корректный отсчет времени. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `e32base.h` и библиотечный файл `euser.lib`.

`CPeriodic` - класс объекта «периодический таймер». Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `e32base.h` и библиотечный файл `euser.lib`.

`CTimer` - базовый класс для объекта «таймер». Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `e32base.h` и библиотечный файл `euser.lib`.

`MBeating` - класс интерфейса таймера. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `e32base.h` и библиотечный файл `euser.lib`.

`TDeltaTimerEntry` - класс, создающий временной вход. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `e32std.h`.

`RTimer` - класс, асинхронные услуги таймера. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `e32std.h` и библиотечный файл `euser.lib`.

`TTimerLockSpec` - перечисляемый тип, содержащий спецификацию блокировки таймера. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `e32std.h`.

### **1.138. Transfer Buffer**

`RTransferWindow` — класс, дескриптор к окну передачи данных. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `E32TransBuf.h` и библиотечный файл `euser.lib`.

`RTransferBuffer` - класс, дескриптор к буферу для обмена данными между процессами. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `E32TransBuf.h` и библиотечный файл `euser.lib`.

`TTransferBufferType` - перечисляемый тип для описания буфера передачи данных. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `E32TransBuf.h`.

### **1.139. To-do List**

`CAgnToDo` - класс, представляющий задачу, которая должна быть выполнена. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `agmentry.h` и библиотечный файл `agnmodel.lib`.

CAgNtodoInstanceList - класс, содержащий список идентификаторов задач. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmlists.h и библиотечный файл agnmodel.lib.

CAgNtodoList - класс, содержащий список задач. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmtodos.h и библиотечный файл agnmodel.lib.

CAgNtodoListList - класс, список идентификаторов задач. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmtodos.h и библиотечный файл agnmodel.lib.

CAgNtodoListNames - класс, список имен задач и их идентификаторов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmtodos.h и библиотечный файл agnmodel.lib.

TAgnAlarmDe faults - класс, параметры настройки по умолчанию звукового оповещения. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmcomon.h и библиотечный файл agnmodel.lib.

TAgnBasicTodo - класс, используемый классом CAgNtodo для сохранения некоторых подробностей задачи. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmbasic.h и библиотечный файл agnmodel.lib.

TAgnTodoDef aults - класс, содержащий настройки по умолчанию для новых задач. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmtodos.h и библиотечный файл agnmodel.lib.

TAgnTodoListId - класс для идентификации задачи в списке задач. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл agmids.h и библиотечный файл agnmodel.lib.

#### **1.140. UID Manipulation**

TCheckedUid - класс для упаковки UID с суммой контроля. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

TUid - класс, глобальный уникальный 32-битный номер. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

TUidName - тип для определения модифицируемого буферного дескриптора для текстовой формы UID. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h.

TUidType - класс для формирования набора их трех универсальных идентификаторов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл e32std.h и библиотечный файл euser.lib.

#### **1.141. UI Control Framework**

CCoeAppUiBase - абстрактный базовый класс UI-приложения. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл coeui.h и библиотечный файл cone.lib.

CCoeAppUi - универсальный класс интерфейса пользователя. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл coeauri.h и библиотечный файл cone.lib.

CCoeAppUiSimple - класс, определяющий Ш-управление для приложения. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл coeauris.h и библиотечный файл cone.lib.

CCoeBrushAndPenContext — класс для установки свойств кисти и карандаша. Доступен от версии Symbian OS 5.1. Для работы необходимо подключить заголовочный файл coecntx.h и библиотечный файл cone.lib.

CCoeControl - базовый класс для классов управления. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл coecntrl.h и библиотечный файл cone.lib.

CCoeEnv - класс, обеспечивающий среду для создания управления. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл coemain.h и библиотечный файл cone.lib.

CCoeScheduler - класс, определяющий Ш-структуру планировщика. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл coemain.h.

CCoeStatic - базовый класс для создания отдельных объектов, сохраняемых CCoeEnv. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл coemain.h и библиотечный файл cone.lib.

ConeUtils - класс, обеспечивающий утилитные функции структуры UI-контроля. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл coeutils.h и библиотечный файл cone.lib.

MCoeControlBrushContext - класс для назначения совместного использования параметров кисти и карандаша. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл coeccntx.h и библиотечный файл cone.lib.

MCoeControlContext - класс интерфейса для назначения совместного использования параметров кисти и карандаша. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл coeccntx.h и библиотечный файл cone.lib.

MCoeControlObserver - класс интерфейса для контроля над процессом отсылки события другому управлению. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл coecobs.h.

MCoeFocusObserver - класс интерфейса, обеспечивающего уведомление на случай потери управления. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл coemain.h.

MCoeForegroundObserver - класс интерфейса, обеспечивающего уведомление об приоритетных или фоновых изменениях. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл coemain.h.

MCoeMessageObserver - класс интерфейса для обработки входящих сообщений сервера окна. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл coemain.h.

`MCoeObserverOfLoadedFep` - класс интерфейса, обеспечивающего уведомление об изменении FEP. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `coemain.h`.

`MObjectProvider` - класс интерфейса позволяющего объекту быть частью сетевых средств доступа. Доступен от версии Symbian OS 7.0s. Для работы необходимо подключить заголовочный файл `coemop.h` и библиотечный файл `cone.lib`.

`NewFepL ()` - функция для возвращения готового объекта классу, полученному из `CCoeFep`. Доступна от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `coefepff.h` и библиотечный файл `cone.lib`.

`RCoeExtensionStorage` - внутренний класс, не предназначен для общего использования.

`SynchronouslyExecuteSettingsDialogL ()` - функция для запуска диалога для настройки FEP. Доступна от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `coefepff.h` и библиотечный файл `cone.lib`.

`TActivePriority` - перечисляемый тип, содержащий приоритеты активных объектов. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `coemain.h`.

`TCoeColorUse` — класс, логический цвет. Доступен от версии Symbian OS 5.1. Для работы необходимо подключить заголовочный файл `coecntrl.h` и библиотечный файл `cone.lib`.

`TCoeContextName` - тип, контекстное имя. Доступен от версии Symbian OS 5.1. Для работы необходимо подключить заголовочный файл `coehelp.h`.

`TCoeHelpContext` - класс, содержащий информацию для сопоставления управления с конкретной темой справки. Доступен от версии Symbian OS 5.1. Для работы необходимо подключить заголовочный файл `coehelp.h` и библиотечный файл `cone.lib`.

`TCoeInputCapabilities` - класс, описывающий используемые формы ввода. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `coeinput.h` и библиотечный файл `cone.lib`.

`TCoeWinPriority` - перечисляемый тип, содержащий значения приоритета окна. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `coedef.h`.

`TDrawNow` - перечисляемый тип, содержащий флажки перерисовки. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `coedef.h`.

`TKeyResponse` - перечисляемый тип, содержащий флажки для обработки случаев. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл `coedef.h`.

`TTypeUid` - класс для формирования UID, который идентифицирует тип объекта, к которому должен быть осуществлен доступ. Доступен от версии Symbian OS 7.0s. Для работы необходимо подключить заголовочный файл `coemop.h`.

`DECLARE_TYPE_ID` - определяет тип объекта `ETypeId` для класса. Для работы необходимо подключить заголовочный файл `coemop.h`.

EAllStdModifiers - константа для представления всех стандартных ключей клавиш. Для работы необходимо подключить заголовочный файл coedef.h.

## 1.142. UI Graphics Utilities

CColorArray - класс, обеспечивающий функции, используемые для создания массива цветов. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл gulcolor.h и библиотечный файл egul.lib.

CColorList - класс, обеспечивающий палитру цветов. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл gulcolor.h и библиотечный файл egul.lib.

CGullIcon - класс, упаковывающий два точечных рисунка (иконки и маски). Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл gulicon.h и библиотечный файл egul.lib.

ColorUtils - класс, сервисные функции для управления цветами. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл gulutil.h и библиотечный файл egul.lib.

DrawUtils - класс, обеспечивающий утилиты для рисования текста в прямоугольнике. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл gulutil.h и библиотечный файл egul.lib.

FontUtils - класс, обеспечивающий утилиты для получения информации о поддерживаемых шрифтах. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл gulutil.h и библиотечный файл egul.lib.

ResourceUtils - класс, утилиты для чтения информации из файла ресурса. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл gulutil.h и библиотечный файл egul.lib.

TextUtils - класс, утилиты для усечения и выравнивания строк. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл gulutil.h и библиотечный файл egul.lib.

TFindWidthOfWidestAbbreviatedDayName — класс для нахождения ширины в пикселях самой широкой аббревиатуры названия дня. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл gulutil.h и библиотечный файл egul.lib.

TFindWidthOfWidestAbbreviatedMonthName - класс для нахождения ширины в пикселях самой широкой аббревиатуры названия месяца. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл gulutil.h и библиотечный файл egul.lib.

TFindWidthOfWidestAmPmName - класс для нахождения ширины в пикселях самой широкой аббревиатуры обозначения дня до или после полудня. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл gulutil.h и библиотечный файл egul.lib.

TFindWidthOfWidestDigit - класс для нахождения ширины в пикселях самой широкой цифры. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл gulutil.h и библиотечный файл egul.lib.

`TFindWidthOfWidestDayName` - класс для нахождения ширины в пикселях самого широкого названия дня. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `gulutil.h` и библиотечный файл `egul.lib`.

`TFindWidthOfWidestTextItem` - класс для нахождения ширины в пикселях самого широкого элемента. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `gulutil.h` и библиотечный файл `egul.lib`.

`TFindWidthOfWidestMonthName` - класс для нахождения ширины в пикселях самого широкого названия месяца. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `gulutil.h` и библиотечный файл `egul.lib`.

`TFindWidthOfWidestDateSuffix` - класс для нахождения ширины в пикселях самого широкого суффикса даты. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `gulutil.h` и библиотечный файл `egul.lib`.

`TGulAdjacent` — перечисляемый тип для определения сторон, на которых границы являются смежными. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `guldef.h`.

`TGulAlignment` - класс, описывающий горизонтальное и вертикальное размещение прямоугольных объектов. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `gulalign.h` и библиотечный файл `egul.lib`.

`TGulAlignmentValue` - перечисляемый тип, параметры выравнивания графических объектов. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `gulalign.h`.

`TGulBorder` — класс для рисования прямоугольной границы. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `gulborder.h` и библиотечный файл `egul.lib`.

`TGulHAlignment` - перечисляемый тип, параметры для горизонтального размещения графических объектов. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `gulalign.h`.

`TGulVAlignment` - перечисляемый тип, параметры для вертикального размещения графических объектов. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `gulalign.h`.

`TLogicalColor` — перечисляемый тип, логические цвета. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `gulcolor.h`.

`TLogicalFont` - класс, упаковывающий атрибуты логического шрифта. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `gulfont.h` и библиотечный файл `egul.lib`.

### **1.143. Uikon Core**

`CCoeScreenDeviceChangeDefaultHandler` - класс, заданный по умолчанию обработчик изменений экрана. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `coeview.h` и библиотечный файл `cone.lib`.

CEikApplication - класс, основа всех Uikon-приложений. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл eikapp.h и библиотечный файл eikcore.lib.

CEikAppUi - класс, обрабатывающий прикладные функции интерфейса пользователя. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл eikappui.h и библиотечный файл eikcore.lib.

CEikAutoMenuItem - класс, определяющий заголовок меню. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл eikamnt.h и библиотечный файл eikcore.lib.

CEikAutoMenuItemArray - класс, создающий стандартный массив объектов CEikAutoMenuItem. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл eikamnt.h и библиотечный файл eikcore.lib.

CEikDocument - основной класс, являющийся базой для всех Uikon-приложений. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл eikdoc.h и библиотечный файл eikcore.lib.

CEikInfoMsgWin - класс для создания информационного окна. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл eikmsg.h и библиотечный файл eikcore.lib.

CEikMsgWin - базовый класс, содержащий функции для создания и отображения информационное сообщение. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл eikmsg.h и библиотечный файл eikcore.lib.

CEikonEnv - внутренний класс, не предназначен для общего использования.

CEikStatusPane - класс для клиентского представления панели состояния. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл eikspane.h и библиотечный файл eikcore.lib.

CEikStatusPaneBase - внутренний класс, не предназначен для общего использования.

EikBubbleHelp - класс, содержащий функции для контроля над всплывающими подсказками. Доступен от версии Symbian OS 6.1. Для работы необходимо подключить заголовочный файл eikbhelp.h и библиотечный файл eikcore.lib.

EikControlFactory - класс, абстрактная «фабрика», осуществляющая управление по типам (ID). Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл eikfctry.h и библиотечный файл eikcore.lib.

EikFileUtils - класс, обеспечивающий настройки диска, пути и утилит-ные функции. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл eikfutil.h и библиотечный файл eikcore.lib.

EikResourceUtils - класс, утилиты для чтения стандартных ресурсов. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл eikrutil.h и библиотечный файл eikcore.lib.

MCoeResourceChangeObserver - класс интерфейса, который обеспечивает информационные окна, сообщающие об изменениях ресурса. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл coemain.h.

MEikAutoMenuObserver - класс, обеспечивающий функциональные дополнительные возможности для автозаголовка меню. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл eikmobs.h и библиотечный файл eikcore.lib.

MEikCommandObserver - класс, обозреватель команд, отвечающих на команды пользователя. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл eikcmobs.h и библиотечный файл eikcostl.lib.

MEikInfdDialog - класс интерфейса для запуска диалога с заголовком и сообщением. Доступен от версии Symbian OS 5.1. Для работы необходимо подключить заголовочный файл eikenv.h.

MEikMenuObserver - класс интерфейса меню. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл eikmobs.h и библиотечный файл eikcore.lib.

MEikStatusPaneObserver - класс для наблюдения за изменением размера панели состояния. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл eikspane.h.

TEikInfdMsgBuf - тип для определения максимальной длины буфера сообщения. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл eikmsg.h и библиотечный файл eikcore.lib.

TEikVirtualCursor - класс для поддержки состояния виртуального курсора в пределах приложения. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл eikvcurs.h и библиотечный файл eikcore.lib.

ERROR\_ARRAY - структура ресурсов, массив кодов ошибки. Доступна от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл uikon.rh.

ERROR\_SET - структура ресурсов, содержащая набор таблиц ошибок. Доступна от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл uikon.rh.

SINGLE\_ERROR - структура ресурсов, локализованный текст для сообщения об ошибке. Доступна от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл uikon.rh.

#### **1.144. Uikon Resources**

BMPBUT - структура ресурсов, кнопка, которая может содержать два изображения, но не текстовую метку. Доступна в Symbian OS с 6.0 по 7.0s. Для работы необходимо подключить заголовочный файл uikon.rh.

BTG\_RESOURCE\_COLLECTION - структура ресурсов, утилита, обеспечивающая кнопки и массив кнопок. Доступна в Symbian OS с 6.0 по 7.0s. Для работы необходимо подключить заголовочный файл uikon.rh.

CMBUT - структура ресурсов, содержащая флажки для операций с кнопками. Доступна в Symbian OS с 6.0 по 7.0s. Для работы необходимо подключить заголовочный файл uikon.rh.



COLORLIST - утилитная структура ресурсов для связи с массивом COLOR ресурсов. Доступна от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл uikon.rh.

COLOR - структура ресурсов, содержащая 3-байтное значение RGB-цвета. Доступна от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл uikon.rh.

CONTROLS - структура ресурсов, утилита, содержащая массив STRUCT. Доступна в Symbian OS с 6.0 по 7.0s. Для работы необходимо подключить заголовочный файл uikon.rh.

CTRL\_COLOR - структура ресурсов для ассоциации логического номера цвета с RGB-значением. Доступна от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл uikon.rh.

DLG\_BUTTONS - структура ресурсов, создающая группу кнопок для диалога. Доступна в Symbian OS с 6.0 по 7.0s. Для работы необходимо подключить заголовочный файл uikon.rh.

DLG\_BUTTON - структура ресурсов, создающая кнопку для диалога. Доступна в Symbian OS с 6.0 по 7.0s. Для работы необходимо подключить заголовочный файл uikon.rh.

EDWIN - структура ресурсов, ресурс текстового редактора. Доступна в Symbian OS с 6.0 по 7.0s. Для работы необходимо подключить заголовочный файл uikon.rh.

EIKCOLORLIST - утилитная структура ресурсов для связи с массивом COLOR ресурсов. Доступна от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл uikon.rh.

EIK\_APP\_INFO - структура ресурсов, информационный ресурс Ш-приложения. Доступна от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл uikon.rh.

FONT - структура ресурсов, шрифт, определяемый UID. Доступна от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл uikon.rh.

HOTKEYS - структура ресурсов для определения горячих клавиш. Доступна от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл uikon.rh.

HOTKEY - структура ресурсов для определения горячих клавиш, используемая HOTKEYS. Доступна от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл uikon.rh.

IMAGE - структура ресурсов, ресурс изображения. Доступна в Symbian OS с 6.0 по 7.0s. Для работы необходимо подключить заголовочный файл uikon.rh.

INT16 - структура ресурсов, содержащая цифровое значение слова. Доступна от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл uikon.rh.

LABEL - структура ресурсов, ресурс метки. Доступна в Symbian OS с 6.0 по 7.0s. Для работы необходимо подключить заголовочный файл uikon.rh.

LISTBOX - структура ресурсов, ресурс метки. Доступна в Symbian OS с 6.0 по 7.0s. Для работы необходимо подключить заголовочный файл uikon.rh.

MENU\_BAR - структура ресурсов, определяющая область окна меню со строкой меню. Доступна в Symbian OS с 6.0 по 7.0s. Для работы необходимо подключить заголовочный файл uikon.rh.

MENU\_ITEM- структура ресурсов, определяющая пункт меню. Доступна в Symbian OS с 6.0 по 7.0s. Для работы необходимо подключить заголовочный файл uikon.rh.

MENU\_PANE - структура ресурсов, определяющая пункты меню в области окна меню. Доступна в Symbian OS с 6.0 по 7.0s. Для работы необходимо подключить заголовочный файл uikon.rh.

MENU\_TITLE - структура ресурсов для ассоциации области окна меню с ее заголовком. Доступна в Symbian OS с 6.0 по 7.0s. Для работы необходимо подключить заголовочный файл uikon.rh.

MNBUT - структура ресурсов, кнопка меню. Доступна в Symbian OS с 6.0 по 7.0s. Для работы необходимо подключить заголовочный файл uikon.rh.

NAMED\_FONT - структура ресурсов, шрифт, определенный по имени. Доступна от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл uikon.rh.

RESOURCE\_LINK - структура ресурсов, утилита для обеспечения ссылки на другой ресурс. Доступна от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл uikon.rh.

TBAR\_CTRL — структура ресурсов для описания управляющих компонентов панели инструментов. Доступна в Symbian OS с 6.0 по 7.0s. Для работы необходимо подключить заголовочный файл uikon.rh.

TBAR\_BUTTON - структура ресурсов, кнопка инструментальной панели. Доступна в Symbian OS с 6.0 по 7.0s. Для работы необходимо подключить заголовочный файл uikon.rh.

TOOLBAND - структура ресурсов, ресурс горизонтальной панели инструментов. Доступна в Symbian OS с 6.0 по 7.0s. Для работы необходимо подключить заголовочный файл uikon.rh.

TOOLBAR - структура ресурсов, ресурс панели инструментов. Доступна в Symbian OS с 6.0 по 7.0s. Для работы необходимо подключить заголовочный файл uikon.rh.

TXTBUT - структура ресурсов, кнопка с текстовой меткой. Доступна в Symbian OS с 6.0 по 7.0s. Для работы необходимо подключить заголовочный файл uikon.rh.

VIEW\_ID - структура ресурсов для отображения ID. Доступна от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл uikon.rh.

### **1.145. USB Client**

DUsbClientController - класс интерфейса клиентского контроллера USB. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл k32usbc.h и библиотечный файл eusbcc.lib.

RDevUsbcClient - класс для пользовательской обработки драйвера USB. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл d32usbc.h.

TCSDescriptorInfo - внутренний класс, не предназначен для общего использования.

TEndpointDescriptorInfo - внутренний класс, не предназначен для общего использования.

TEndpointTransferInfo - внутренний класс, не предназначен для общего использования.

TEndpointNumber - перечисляемый тип, определяет набор возможных чисел конечной точки. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл d32usbc.h.

TEndpointState - перечисляемый тип, определяет возможное STALL состояние конечной точки. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл d32usbc.h.

TTransferDirection - перечисляемый тип, определяет направление передачи данных. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл k32usbc.h.

TUsbcClassInfo - класс, описывающий типа класса для интерфейса. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл d32usbc.h.

TUsbcConfigurationDescriptor - класс, дескриптор, описывающий конфигурацию устройства. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл k32usbc.h.

TUsbcDeviceDescriptor - класс, дескриптор, содержащий информацию, описывающую USB-устройство. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл k32usbc.h.

TUsbcDeviceEvent — перечисляемый тип, определяющий набор событий, связанных с изменением состояния USB-устройства. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл k32usbc.h.

TUsbcDeviceState - перечисляемый тип, определяет возможное состояние USB-контроллера. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл d32usbc.h.

TUsbcEndpointCaps - класс, содержащий информацию для конечной точки. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл d32usbc.h.

TUsbcEndpointData - класс, содержащий вероятность конечной точки на устройстве. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл d32usbc.h.

TUsbcEndpointInfo - класс, содержащий информацию о конечной точке. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл d32usbc.h.

TUsbcEndpointStatusCallback - класс, формирующий обратный вызов при изменении статуса конечной точки. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл k32usbc.h.

TUsbcEpOState - перечисляемый тип, определяющий возможные состояния нулевой конечной точки. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл k32usbc.h.

TUsbdnterf acelnf oBuf - тип, определяет пакетный буфер для TUsbcIn-terf acelnf o типа объекта. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл d32usbc.h.

TUsbdnterf acelnf o - класс описывает интерфейс и конечные точки. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл d32usbc.h.

TUsbcLangIdDescriptor - класс, дескриптор, содержащий поддерживаемый язык. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл k32usbc.h.

TUsbcRequestCallBack - класс, формирующий обратный вызов после завершения передачи данных. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл k32usbc.h.

TUsbcStatusCallBack - класс, формирующий обратный вызов при изменении состояния клиентского контроллера USB. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл k32usbc.h.

TUsbcStatusNotif ylnf o - внутренний класс, не предназначен для общего использования.

TUsbcStopMode — перечисляемый тип, формирующий набор факторов для остановки аппаратных средств USB-контроллера. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл k32usbc.h.

TUsbDeviceCaps - тип, определяет пакетный буфер для объектов устройства. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл d32usbc.h.

TUsbDeviceCapsVOI - класс, содержит возможности устройства. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл d32usbc.h.

### **1.146. WAP Messaging**

CWapBoundCLPushService - класс для просмотра WAP-сообщений от любого отправителя. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл wapmessage.h и библиотечный файл wapmsgcli.lib.

CWapBoundDatagramService - класс, посылает и получает датаграммы по WDP. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл wapmessage.h и библиотечный файл wapmsgcli.lib.

CWapFullySpecCLPushService - класс для просмотра WAP-сообщений от отдельного компьютера. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл wapmessage.h и библиотечный файл wapmsgcli.lib.

CWapFullySpecDatagramService - класс, посылает и получает датаграммы по WDP, применяя отдельный компьютер. Доступен от версии Symbi-

an OS 7.0. Для работы необходимо подключить заголовочный файл `wapmessage.h` и библиотечный файл `wapmsgcli.lib`.

`CWapMessageUtils` - класс, содержит утилиты для передачи сообщений по WAP. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл `wapmessage.h` и библиотечный файл `wapmsgcli.lib`.

### **1.147. WAPSMS Protocol Module**

`TWapAddr` - класс, тип адреса сокета. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `es_wsms.h`.

`TWapPortNumber` - перечисляемый тип, содержащий настройки WAP-пор-та. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `es_wsms.h`.

`TWapSmsDataCodingScheme` - перечисляемый тип, определяющий типы кодирующих схем WAP-данных. Доступен от версии Symbian OS 6.0. Для работы необходимо подключить заголовочный файл `es_wsms.h`.

### **1.148. WAP Stack**

`CCapCodec` - класс, определяет набор WAP-возможностей. Доступен в Symbian OS с 6.0 по 7.0. Для работы необходимо подключить заголовочный файл `capcodec.h` и библиотечный файл `CapCodec.lib`.

`RWAPConn` - класс, доступные операции при подключении на всех уровнях стека. Доступен в Symbian OS с 6.0 по 7.0. Для работы необходимо подключить заголовочный файл `wapcli.h` и библиотечный файл `wapstkcli.lib`.

`RWAPServ` - класс, клиентский дескриптор к WAP-сессии сервера Symbian OS. Доступен в Symbian OS с 6.0 по 7.0. Для работы необходимо подключить заголовочный файл `wapcli.h` и библиотечный файл `wapstkcli.lib`.

`RWDPCConn` - класс, WDP-обслуживание передачи датаграмм. Доступен в Symbian OS с 6.0 по 7.0. Для работы необходимо подключить заголовочный файл `wapcli.h` и библиотечный файл `wapstkcli.lib`.

`RWSPCLConn` - класс, WSP без установления соединения. Доступен в Symbian OS с 6.0 по 7.0. Для работы необходимо подключить заголовочный файл `wapcli.h` и библиотечный файл `wapstkcli.lib`.

`RWSPCOConn` - класс, WSP с соединением. Доступен в Symbian OS с 6.0 по 7.0. Для работы необходимо подключить заголовочный файл `wapcli.h` и библиотечный файл `wapstkcli.lib`.

`RWSPCOTrans` - класс, транзакция с WSP-соединением. Доступен в Symbian OS с 6.0 по 7.0. Для работы необходимо подключить заголовочный файл `wapcli.h` и библиотечный файл `wapstkcli.lib`.

`TBearer` - перечисляемый тип однонаправленного канала. Доступен в Symbian OS с 6.0 по 7.0. Для работы необходимо подключить заголовочный файл `wapcli.h`.

## 1.149. Window Server Client Side

CClickMaker - класс интерфейса для звукового сигнала на нажатую клавишу. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл w32click.h.

CDirectScreenAccess - класс, обеспечивающий прямой доступ объекта без использования сервера окна. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл w32std.h и библиотечный файл ws32.lib.

CreateClickMakerL () - функция звукового сигнала на нажатую клавишу. Доступен от версии Symbian OS 7.0.

CWindowGc - класс, графический контекст окна. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл w32std.h и библиотечный файл ws32.lib.

CWsBitmap - класс, точечный рисунок сервера окна. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл w32std.h и библиотечный файл ws32.lib.

CWsScreenDevice - класс, представляющий экран устройства. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл w32std.h и библиотечный файл ws32.lib.

MAbortDirectScreenAccess - класс, интерфейс для прямого доступа к экрану. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл w32std.h.

MDirectScreenAccess — класс интерфейса для перезапускаемого прямого доступа к экрану. Доступен от версии Symbian OS 7.0. Для работы необходимо подключить заголовочный файл w32std.h.

MWsClientClass - базовый класс для классов, отвечающих за сервер окна. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл w32std.h и библиотечный файл ws32.lib.

RAnim— класс, дескриптор к классу анимации. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл w32std.h и библиотечный файл ws32.lib.

RAnimDll — класс, интерфейс к анимации DLL. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл w32std.h и библиотечный файл ws32.lib.

RBackedUpWindow - класс, дескриптор к рисуемому окну. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл w32std.h и библиотечный файл ws32.lib.

RBlankWindow - класс, пустое окно. Доступен от версии Symbian OS 5.0. Для работы необходимо подключить заголовочный файл w32std.h и библиотечный файл ws32.lib.

# Приложение 2 Техническая документация телефонов Symbian OS

В этом приложении приведены основные технические параметры для телефонов работающих под управлением операционной системы Symbian. Вся информация взята с сайта компании Symbian Ltd. по адресу в Интернет [www.symbian.com](http://www.symbian.com). Для некоторых телефонов на базе Symbian OS 8.0 документация пока имеется не в полном объеме, поскольку телефоны еще не поступали в продажу.

## **Arima U300**



<i>Операционная система</i>	Symbian OS 7.0
<i>Платформа</i>	UIQ2.1
<i>Дисплей</i>	2.66 дюйма, TFT, LCD, 65536 цветов, сенсорный
<i>Память</i>	16Мбайт SDRAM, 32Мбайт NorFlash
<i>Java</i>	CLDC1.0/MIDP2.0
<i>Камера</i>	1.3 мегапикселя CMOS, 4x цифровой зуммер
<i>GSM</i>	900/1800/1900
<i>GPRS</i>	
<i>Bluetooth</i>	да
<i>USB</i>	да
<i>Инфракрасный порт</i>	нет
<i>Дополнительные возможности</i>	нет MPEG-4 Camcorder/DSC, SyncML, MP3

## BenQ P30



*Операционная система*  
*Платформа*  
*Дисплей*  
  
*Память*  
  
*Java*  
*Камера*  
*GSM*  
*GPRS*  
*Bluetooth*  
*USB*  
*Инфракрасный порт*  
*Дополнительные возможности*

Symbian OS 7.0  
UIQ2.1  
208 x 320 пикселей,  
65536 цветов, TFT  
12 Мбайт доступных  
пользователю,  
MMC/SD  
CLDC1.0/MIDP2.0  
да  
900/1800/1900  
нет  
да  
да  
да  
MP3, MPEG-4

## Foma F8801ES

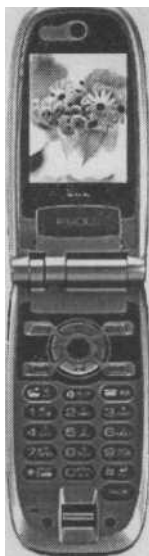


*Операционная система*  
*Платформа*  
  
*Дисплей*  
  
*Память*  
  
*Java*  
*Камера*  
*GSM*  
*GPRS*  
*Bluetooth*  
*USB*  
*Инфракрасный порт*  
*Дополнительные возможности*

Symbian OS  
NTT DoCoMo  
(Японская версия)  
240 x 320 пикселей,  
65536 цветов, TFT  
8 Мбайт доступно  
пользователю  
CLDC 1.0/MIDP 1.0  
да  
нет  
нет  
да  
нет  
да  
запись видео 85  
минут



### **FomaF900i**



<i>Операционная система</i>	SymbianOS6.1
<i>Платформа</i>	NTT DoCoMo
<i>Дисплей</i>	LCD 2.2 дюйма, 262144 цветов
<i>Память</i>	8 Мбайт доступных пользователю
<i>Java</i>	нет
<i>Камера</i>	1.23 мегапикселя
<i>GSM</i>	нет
<i>GPRS</i>	нет
<i>Bluetooth</i>	нет
<i>USB</i>	да
<i>Инфракрасный порт</i>	нет
<i>Дополнительные возможности</i>	запись видео 100 минут

### **Foma F900H**



<i>Операционная система</i>	Symbian OS 6.1
<i>Платформа</i>	NTT DoCoMo
<i>Дисплей</i>	LCD 2.2 дюйма, 262144 цветов
<i>Память</i>	8 Мбайт доступных пользователю
<i>Java</i>	нет
<i>Камера</i>	1.28 мегапикселя
<i>GSM</i>	нет
<i>GPRS</i>	нет
<i>Bluetooth</i>	нет
<i>USB</i>	да
<i>Инфракрасный порт</i>	нет
<i>Дополнительные возможности</i>	запись видео 120 минут

## FomaF901c



<i>Операционная система</i>	SymbianOS6.1
<i>Платформа</i>	NTT DoCoMo
<i>Дисплей</i>	LCD 2.2 дюйма, 262144 цветов
<i>Память</i>	8 Мбайт доступных пользователю
<i>Java</i>	нет
<i>Камера</i>	1.28 мегапикселя
<i>GSM</i>	нет
<i>GPRS</i>	нет
<i>Bluetooth</i>	нет
<i>USB</i>	да
<i>Инфракрасный порт</i>	нет
<i>Дополнительные возможности</i>	запись видео 90 минут

## Foma F2051



<i>Операционная система</i>	Symbian OS 7.0s
<i>Платформа</i>	Серия 60 версия 2.1
<i>Дисплей</i>	176 x 208 пикселей, 65536 цветов, TFT
<i>Память</i>	8 Мбайт доступных пользователю
<i>Java</i>	CLDC1.0/MIDP2.0
<i>Камера</i>	да
<i>GSM</i>	E900/1800/1900
<i>GPRS</i>	да
<i>Bluetooth</i>	да
<i>USB</i>	нет
<i>Инфракрасный порт</i>	нет
<i>Дополнительные возможности</i>	RealMedia H3GPP видео

## **Lenovo P930**



<i>Операционная система</i>	Symbian OS 8.0
<i>Платформа</i>	Серия 60 версия 2.6
<i>Дисплей</i>	176 x 208 пикселей
<i>Память</i>	32 Мбайт доступных пользователю, SD/MMS
<i>Java</i>	CLDC1.0/MIDP2.0
<i>Камера</i>	1.3 мегапикселя
<i>GSM</i>	да
<i>GPRS</i>	да
<i>Bluetooth</i>	да
<i>USB</i>	мини <i>USB</i>
<i>Инфракрасный порт</i>	да
<i>Дополнительные возможности</i>	MP3, Word, Excel, PPT, PDF

## **Motorola A920**



<i>Операционная система</i>	Symbian OS 7.0
<i>Платформа</i>	UIQ2.0
<i>Дисплей</i>	208 x 320 пикселей, 65536 цветов TFT
<i>Память</i>	8 Мбайт доступных пользователю, MMC/SD
<i>Java</i>	CLDC 1.0/MIDP 1.0
<i>Камера</i>	да
<i>GSM</i>	900/1800/1900
<i>GPRS</i>	да
<i>Bluetooth</i>	да
<i>USB</i>	нет
<i>Инфракрасный порт</i>	да
<i>Дополнительные возможности</i>	да MP3 плеер

## Motorola A925



*Операционная система*  
*Платформа*  
*Дисплей*  
  
*Память*  
*Java*  
*Камера*  
*GSM*  
*GPRS*  
*Bluetooth*  
*USB*  
*Инфракрасный порт*  
*Дополнительные возможности*

Symbian OS 7.0  
UIQ2.0  
208 x 320 пикселей,  
65536 цветов TFT  
8 Мбайт доступных  
пользователю,  
MMC/SD  
CLDC 1.0/MIDP 1.0  
да  
900/1800/1900  
да  
да  
да  
да  
MP3 плеер

## Motorola A1000



*Операционная система*  
*Платформа*  
*Дисплей*  
  
*Память*  
*Java*  
*Камера*  
*GSM*  
  
*GPRS*  
*Bluetooth*  
*USB*  
*Инфракрасный порт*  
*Дополнительные возможности*

Symbian OS 7.0  
UIQ2.1  
208 x 320 пикселей,  
65536 цветов, TFT  
24 Мбайт доступных  
пользователю, Triflash-R  
CLDC 1.0/MIDP 2.0  
1.2 мегапикселя, 4x  
зуммер  
900/1800/1900,  
WCDMA2100  
да  
да  
да  
нет  
MPEG-4, MP3, WMV,  
\УМД HTML

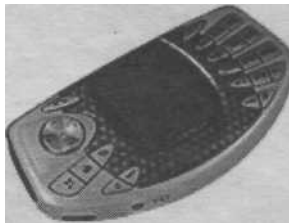
## Motorola A1010



*Операционная система*  
*Платформа*  
*Дисплей*  
*Память*  
*Java*  
*Камера*  
*GSM*  
*GPRS*  
*Bluetooth*  
*USB*  
*Инфракрасный порт*  
*Дополнительные возможности*

Symbian OS 7.0  
UIQ2.1  
208 x 320 пикселей, TFT  
48 Мегабайт доступных  
пользователю, Transflash  
CLDC1.0/MIDP 2.0  
2.0 мегапикселя, 4x  
зуммер  
900/1800/1900,  
WCDMA2100  
да  
да  
да  
нет  
HTML, SyncML, Adobe  
PDF, Unzip, Microsoft  
Word,  
Microsoft Excel, Microsoft  
PowerPoint, IMAP4, POP3

## Nokia N-Gage



*Операционная система*  
*Платформа*  
*Дисплей*  
*Память*  
*Java*  
*Камера*  
*GSM*  
*GPRS*  
*Bluetooth*  
*USB*  
*Инфракрасный порт*  
*Дополнительные возможности*

Symbian OS 6.1  
Серия 60 версия 1.2  
176 x 208 пикселей, 4096 цветов  
4 Мбайт доступных пользователю,  
MMC  
CLDC 1.0/MIDP 1.0  
нет  
900/1800/1900  
да  
да  
да  
нет  
FM радио, MP3 плеер,  
игры на картах памяти

## Nokia N-Gage QD



<i>Операционная система</i>	Symbian OS 6.1
<i>Платформа</i>	Серия 60 версия 1.2
<i>Дисплей</i>	176 x 208 пикселей, 4096 цветов
<i>Память</i>	4 Мбайт доступных пользователю, MMC
<i>Java</i>	CLDC 1.0/MIDP 1.0
<i>Камера</i>	нет
<i>GSM</i>	нет
<i>GPRS</i>	900/1800/1900
<i>Bluetooth</i>	да
<i>USB</i>	да
<i>Инфракрасный порт</i>	нет
<i>Дополнительные возможности</i>	нет игры на картах памяти

## Nokia 3230



<i>Операционная система</i>	Symbian OS 7.0s
<i>Платформа</i>	Серия 60
<i>Дисплей</i>	65536 цветов
<i>Память</i>	32 Мбайт доступных пользователю, MMC
<i>Java</i>	CLDC 1.0/MIDP 2.0
<i>Камера</i>	1.3 мегапикселя
<i>GSM</i>	900/1800/1900, EDGE
<i>GPRS</i>	
<i>Bluetooth</i>	да
<i>USB</i>	да
<i>Инфракрасный порт</i>	нет
<i>Дополнительные возможности</i>	нет HTML, MP3, FM, видео 60 минут

## **Nokia 3650/3600**



*Операционная система*

*Платформа*

*Дисплей*

*Память*

*Java*

*Камера*

*GSM*

*GPRS*

*Bluetooth*

*USB*

*Инфракрасный порт*

*Дополнительные возможности*

Symbian OS 64

Серия 60 версия 1.2

176 x 208 пикселей,

4096/65536 цветов

3,4 Мбайт

доступных

пользователю, MMC

CLDC 1.0/MIDP 1.0

да

900/1800/1900

да

нет

нет

нет

Wireless Media API

и Mobile Media API

## **Nokia 3660/3620**



*Операционная система*

*Платформа*

*Дисплей*

*Память*

*Java*

*Камера*

*GSM*

*GPRS*

*Bluetooth*

*USB*

*Инфракрасный порт*

*Дополнительные возможности*

Symbian OS 6.1

Серия 60 версия 1.2

176 x 208 пикселей,

4096/65536 цветов

4 Мбайт доступных

пользователю, MMC

CLDC 1.0/MIDP

1.0

да

900/1900

**да**

**да**

нет

нет

Wireless Media API

и Mobile Media API

## Nokia 6260



*Операционная система*  
*Платформа*  
*Дисплей*  
  
*Память*  
*Java*  
*Камера*  
*GSM*  
*GPRS*  
*Bluetooth*  
*USB*  
*Инфракрасный порт*  
*Дополнительные возможности*

Symbian OS 7.0  
Серия 60 версия 2.1  
176 x 208 пикселей,  
65536 цветов, TFT  
MMC  
CLDC 1.0/MIDP 2.0  
да  
900/1800/1900  
да  
да  
да  
нет  
HTML, Nokia  
Mobile  
VPN клиент

## Nokia 6600



*Операционная система*  
*Платформа*  
*Дисплей*  
  
*Память*  
*Java*  
*Камера*  
*GSM*  
*GPRS*  
*Bluetooth*  
*USB*  
*Инфракрасный порт*  
*Дополнительные возможности*

Symbian OS 7.0s  
Серия 60 версия 2.0  
176 x 208 пикселей,  
65536 цветов, TFT  
6 Мбайт доступно  
пользователю, MMC  
CLDC 1.0/MIDP 2.0  
да  
900/1800/1900  
да  
да  
да  
да  
RealMedia и 3GPP,  
SyncML,  
SMTP, IMAP4, POP3



## Nokia 6620



*Операционная система*

*Платформа*

*Дисплей*

*Память*

*Java*

*Камера*

*GSM*

*GPRS*

*Bluetooth*

*USB*

*Инфракрасный порт*

*Дополнительные возможности*

Symbian OS 7.0s

Серия 60 версия 2.1

176 x 208 пикселей,

65536 цветов

6 Мбайт доступно

пользователю, MMC

CLDC1.0/MIDP2.0

да

850/1900/1800, EDGE

да

да

нет

да

HTML, SyncML

## Nokia 6630



*Операционная система*

*Платформа*

*Дисплей*

*Память*

*Java*

*Камера*

*GSM*

*GPRS*

*Bluetooth*

*USB*

*Инфракрасный порт*

*Дополнительные возможности*

Symbian OS 8.0a

Серия 60 версия 2.6

176 x 208 пикселей,

65536 цветов, TFT

10 Мбайт доступных

пользователю, MMC

CLDC1.1/MIDP2.0

1.23 мегапикселя, 6x

зуммер

900/1800/1900, WCDMA

да, EDGE

да

да

нет

HTML, MP3, SMTP, POP3,

IMAP4 запись видео

60 минут

## Nokia 6670



<i>Операционная система</i>	Symbian OS 7.0s
<i>Платформа</i>	Серия 60 версия 2.1
<i>Дисплей</i>	176 x 208 пикселей, 65536 цветов, TFT
<i>Память</i>	8 Мбайт доступных пользователю, MM С
<i>Java</i>	CLDC1.0/MIDP2.0
<i>Камера</i>	1 мегапиксель, 4x зуммер
<i>GSM</i>	850/950/1800/1900
<i>GPRS</i>	да
<i>Bluetooth</i>	да
<i>USB</i>	да
<i>Инфракрасный порт</i>	нет
<i>Дополнительные возможности</i>	HTML, SMTP, POP3, IMAP4

## Nokia 66S0



<i>Операционная система</i>	Symbian OS 8.0
<i>Платформа</i>	Серия 60 версия 2.6
<i>Дисплей</i>	176 x 208 пикселей, 262144 цветов
<i>Память</i>	10 Мбайт доступных пользователю, MMC, карта 64 Мбайт в стандартной поставке
<i>Java</i>	CLDC1.1/MIDP2.0
<i>Камера</i>	1.3 мегапикселя, 6x зуммер
<i>GSM</i>	850/900/1800/1900, WCDMA
<i>GPRS</i>	да, EDGE
<i>Bluetooth</i>	да
<i>USB</i>	нет
<i>Инфракрасный порт</i>	нет
<i>Дополнительные возможности</i>	нет Видефон, Nokia XpressPrint, MP3, Word, Power Point, 28 языков вменю

## Nokia 6681



*Операционная система*  
*Платформа*  
*Дисплей*

*Память*  
*Java*  
*Камера*  
*GSM*  
*GPRS*  
*Bluetooth*  
*USB*

Symbian OS 8.0  
Серия 60 версия 2.6  
176 x 208 пикселей,  
262144 цветов  
MMC  
CLDC1.1/MIDP2.0  
1.3 мегапикселя  
900/1800/1900,  
WCDMA  
да: EDGE  
да

*Дополнительные возможности* Nokia  
*Инфракрасный порт* нет  
нет

## Nokia 6682



*Операционная система*  
*Платформа*  
*Дисплей*

*Память*  
*Java*  
*Камера*  
*GSM*  
*GPRS*  
*Bluetooth*  
*USB*  
*Инфракрасный порт*

Symbian OS 8.0  
Серия 60 версия 2.6  
176 x 208 пикселей,  
262144 цветов  
10 Мбайт доступных  
пользователю, MMC  
CLDC 1.1/MIDP 2.0  
1.3 мегапикселя  
900/1800/1900  
да  
да  
нет  
нет  
Nokia XpressPrint,  
Microsoft Office 97, 98,  
2000,  
XP, Adobe Reader PDF,  
Microsoft Word, Microsoft  
Excel, Microsoft  
PowerPoint

## Nokia 7610



*Операционная система*

*Платформа*

*Дисплей*

*Память*

*Java*

*Камера*

*GSM*

*GPRS*

*Bluetooth*

*USB*

*Инфракрасный порт*

*Дополнительные возможности*

Symbian OS 7.0s

Серия 60 версия 2.1

176 x 208 пикселей,

65536 цветов

72 Мбайт

CLDC 1.0/MIDP 2.0

да

850/950/1800/1900

да

да

да

нет

MP3/ААС

## Nokia 7650



*Операционная система*

*Платформа*

*Дисплей*

*Память*

*Java*

*Камера*

*GSM*

*GPRS*

*Bluetooth*

*USB*

*Инфракрасный порт*

*Дополнительные возможности*

Symbian OS 6.1

Серия 60

176 x 208 пикселей,

4096 цветов

4 Мбайт доступных

пользователю

CLDC 1.0/MIDP 1.0

да

900/1800

да

да

нет

да

## Nokia 7710



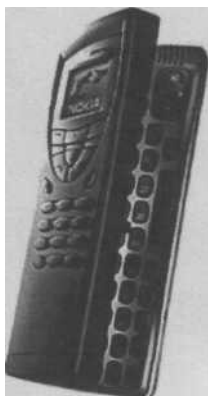
<i>Операционная система</i>	Symbian OS 7.0s
<i>Платформа</i>	Серия 90
<i>Дисплей</i>	320 x 640 пикселей, 65536 цветов, сенсорный
<i>Память</i>	90 Мбайт доступных пользователю, MMC,
<i>Java</i>	карта MMC 128 Мбайт в комплекте
<i>Камера</i>	CLDC 1.0/MIDP 2.0
<i>GSM</i>	да
<i>GPRS</i>	900/1800/1900
<i>Bluetooth</i>	да, EGPRES
<i>USB</i>	да
<i>Инфракрасный порт</i>	да
<i>Дополнительные возможности</i>	нет DVB-H mobile TV, HTML, Flash 6, FM, MP3

## Nokia 9210



<i>Операционная система</i>	Symbian OS 6.0
<i>Платформа</i>	Серия 80
<i>Дисплей</i>	640 x 200 пикселей, 4096 цветов
<i>Память</i>	40 Мбайт доступных пользователю, MMC
<i>Java</i>	CLDC 1.0/MIDP 1.0
<i>Камера</i>	нет
<i>GSM</i>	нет
<i>GPRS</i>	900/1800
<i>Bluetooth</i>	нет
<i>USB</i>	нет
<i>Инфракрасный порт</i>	нет
<i>Дополнительные возможности</i>	да HTML браузер, Word и PowerPoint

## Nokia 9290



*Операционная система*

*Платформа*

*Дисплей*

*Память*

*Java*

*Камера*

*GSM*

*GPRS*

*Bluetooth*

*USB*

*Инфракрасный порт*

*Дополнительные возможности*

Symbian OS 6.0

Серия 80

640 x 200 пикселей,  
4096 цветов

32 Мбайт доступных  
пользователю, MMC  
CLDC 1.0/MIDP 1.0

нет

1900

да

нет

да

нет

Word, PowerPoint,  
HTML

## Nokia 9300



*Операционная система*

*Платформа*

*Дисплей*

*Память*

*Java*

*Камера*

*GSM*

*GPRS*

*Bluetooth*

*USB*

*Инфракрасный порт*

*Дополнительные возможности*

Symbian OS 7.0s

Серия 80

640 x 200 пикселей,  
65536 цветов

80 Мбайт доступных  
пользователю, MMC  
CLDC1.0/MIDP2.0

нет

900/1800/1900

да

да

да

нет

MP3, HTML

4.01, JavaScript

1.3, QWERTY клавиатура,  
IMAP4, POP3, APOP,  
SMTP, MIME, IMAP4-  
SSL/

TLS, POP3-SSL/TLS,

SMTP-SSL/TLS, OMA

## **Nokia 9500**



*Операционная система*  
*Платформа*  
*Дисплей*  
*Память*  
*Java*  
*Камера*  
*GSM*  
*GPRS*  
*Bluetooth*  
*USB*  
*Инфракрасный порт*  
*Дополнительные возможности*

Symbian OS 7.0s  
Серия 80  
640 x 200 пикселей,  
65536 цветов  
80 Мбайт доступных  
пользователю, MMC  
CLDC1.0/MIDP2.0  
да  
900/1800/1900  
да, EGPRS  
да  
да  
нет  
MPEG-4, MP3, Opera  
HTML, QWERTY  
клавиатура, IMAP4,  
POP3,  
SMTP, SyncML, WiFi

## **Panasonic X700**



*Операционная система*  
*Платформа*  
*Дисплей*  
*Память*  
*Java*  
*Камера*  
*GSM*  
*GPRS*  
*Bluetooth*  
*USB*  
*Инфракрасный порт*  
*Дополнительные возможности*

Symbian OS 7.0s  
Серия 60 версия 2.0  
176 x 208 пикселей,  
65536 цветов, TFT  
miniSD  
CLDC 1.0/MIDP 2.0  
да  
900/1800/1900  
да  
да  
да  
нет  
Word, Excel,  
PowerPoint,  
Video Photo редактор

## **Panasonic X800**



*Операционная система*  
*Платформа*  
*Дисплей*  
  
*Память*  
*Java*  
*Камера*  
*GSM*  
*GPRS*  
*Bluetooth*  
*USB*  
*Инфракрасный порт*  
*Дополнительные возможности*

Symbian OS 7.0  
Серия 60 версия 2.1  
176 x 208 пикселей,  
65536 цветов, TFT  
MMC  
CLDC1.0/MIDP2.0  
да  
900/1800/1900  
да  
да  
нет  
нет  
Word, Excel,  
PowerPoint

## **SendoX**



*Операционная система*  
*Платформа*  
*Дисплей*  
  
*Память*  
  
*Java*  
*Камера*  
*GSM*  
*GPRS*  
*Bluetooth*  
*USB*  
*Инфракрасный порт*  
*Дополнительные возможности*

Symbian OS 6.1  
Серия 60 версия 1.2  
176 x 220 пикселей,  
65536 цветов  
12 Мбайт доступных  
пользователю,  
MMC/SD  
CLDC 1.0/MIDP 1.0  
Да  
900/1800/1900  
да  
да  
да  
да  
Wireless Media API  
и Mobile Media API



## SendoX2



*Операционная система*  
*Платформа*  
*Дисплей*  
  
*Память*  
  
*Java*  
*Камера*  
*GSM*  
*GPRS*  
*Bluetooth*  
*USB*  
*Инфракрасный порт*  
*Дополнительные возможности*

Symbian OS 6.1  
Серия 60 версия 1.2  
176 x 208 пикселей,  
65536 цветов, TFT  
32 Мбайт доступных  
пользователю, miniSD  
до 1Gb  
CLDC 1.0/MIDP 1.0  
1.3 мегапикселя, 8x  
зуммер  
900/1800/1900  
да  
да  
да  
нет  
HTML, MP3

## Siemens SX1



*Операционная система*  
*Платформа*  
*Дисплей*  
  
*Память*  
  
*Java*  
*Камера*  
*GSM*  
*GPRS*  
*Bluetooth*  
*USB*  
*Инфракрасный порт*  
*Дополнительные возможности*

Symbian OS 6.1  
Серия 60 версия 1.2  
176 x 208 пикселей,  
65536 цветов, TFT  
4 Мбайт доступных  
пользователю, MMC  
CLDC 1.0/MIDP 1.0  
да  
900/1800/1900  
да  
да  
да  
да  
да  
FM радио, E-mail  
клиент,  
MP3

## Sony Ericsson P800



*Операционная система*  
*Платформа*  
*Дисплей*  
*Память*  
*Java*  
*Камера*  
*GSM*  
*GPRS*  
*Bluetooth*  
*USB*  
*Инфракрасный порт*  
*Дополнительные возможности*

Symbian OS 7.0  
UIQ2.0  
208 x 320 пикселей,  
4096 цветов  
12 Мбайт доступных  
пользователю, Sony MS  
Duo  
CLDC 1.0/MIDP 1.0  
да  
900/1800/1900  
да  
да  
да  
да  
HTML браузер

## Sony Ericsson P900



*Операционная система*  
*Платформа*  
*Дисплей*  
*Память*  
*Java*  
*Камера*  
*GSM*  
*GPRS*  
*Bluetooth*  
*USB*  
*Инфракрасный порт*  
*Дополнительные возможности*

Symbian OS 7.0  
UIQ2.1  
208 x 320 пикселей,  
65536 цветов  
16 Мбайт доступных  
пользователю, Sony MS  
Duo  
CLDC 1.0/MIDP 2.0  
да  
900/1800/1900  
да  
да  
да  
да  
MP3, HTML браузер

## Sony Ericsson P900i



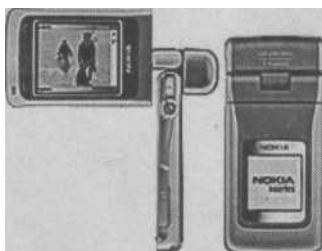
<i>Операционная система</i>	Symbian OS 7.0
<i>Платформа</i>	UIQ2.1
<i>Дисплей</i>	208 x 320 пикселей, 262114 цветов
<i>Память</i>	64 Мбайт доступных пользователю, Sony MS Duo
<i>Java</i>	CLDC1.0/MIDP2.0
<i>Камера</i>	да
<i>GSM</i>	850/900/1800/1900
<i>GPRS</i>	да
<i>Bluetooth</i>	да
<i>USB</i>	да
<i>Инфракрасный порт</i>	да
<i>Дополнительные возможности</i>	да MP3, HTML браузер, SyncML

## Nokia N70



<i>Операционная система</i>	Symbian OS 8.1a
<i>Платформа</i>	Серия 60 версия 2.6
<i>Дисплей</i>	176 x 208 пикселей, 262144 цветов
<i>Память</i>	35 Мбайт доступных пользователю, MMC CLDC1.1/MIDP2.0,
<i>Java</i>	Java 3D API
<i>Камера</i>	2 мегапикселя
<i>GSM</i>	900/1800/1900, WCDMA
<i>GPRS</i>	да, EGPRS
<i>Bluetooth</i>	да
<i>USB</i>	да
<i>Инфракрасный порт</i>	нет
<i>Дополнительные возможности</i>	Nokia XpressPrint, Adobe Reader PDF, Microsoft Word, Microsoft Excel, Microsoft PowerPoint, SMTP/POP3/IMAP4, Visual Radio, MPG-4, MP3

## **Nokia N90**



<i>Операционная система</i>	Symbian OS 8.1a
<i>Платформа</i>	Серия 60 версия 2.6
<i>Дисплей</i>	352 x 416 пикселей, 262144 цветов
<i>Память</i>	35 Мбайт доступных пользователю, MMC
<i>Java</i>	CLDC 1.1/MIDP 2.0, Java 3D API
<i>Камера</i>	2 мегапикселя
<i>GSM</i>	900/1800/1900, WCDMA
<i>GPRS</i>	да, EGPRS
<i>Bluetooth</i>	да
<i>USB</i>	да
<i>Инфракрасный порт</i>	нет
<i>Дополнительные возможности</i>	Nokia XpressPrint, Adobe Reader PDF, Microsoft Word, Microsoft Excel, Microsoft PowerPoint, SMTP/POP3/IMAP4, Visual Radio, MPG-4, MP3

## **Nokia N91**



<i>Операционная система</i>	Symbian OS 9.1
<i>Платформа</i>	Серия 60 версия 3.0
<i>Дисплей</i>	176 x 208 пикселей, 262144 цветов
<i>Память</i>	10 Мбайт доступных пользователю, ННД 4 Гбайт
<i>Java</i>	CLDC1.1/MIDP2.0, Java 3D API 2 мегапикселя
<i>Камера</i>	900/1800/1900, WCDMA
<i>GSM</i>	да, EGPRS
<i>GPRS</i>	да
<i>Bluetooth</i>	да
<i>USB</i>	нет
<i>Инфракрасный порт</i>	Nokia XpressPrint, Adobe Reader PDF, Microsoft Word, Microsoft Excel, Microsoft PowerPoint, SMTP/POP3/IMAP4, Visual Radio
<i>Дополнительные возможности</i>	

## Приложение 3. Интернет ресурсы

В этом приложении перечислены основные Интернет ресурсы, связанные с программированием под операционную систему Symbian. Ссылки разделены на тематические разделы.

### **Компания Symbian Ltd.**

<http://www.symbian.ru>

<http://www.symbian.com>

<http://www.symbiandevnet.com>

<http://www3.symbian.com/faq.nsf>

### **Инструментальные средства разработчика**

#### ***Nokia***

<http://ww-vv.forum.nokia.com>

#### ***Sony Ericsson***

<http://www.sonyericsson.com/developer>

<http://developer.sonyericsson.com>

#### ***Siemens***

<http://www.siemens-mobile.com/developer>

#### ***Sendo***

<http://www.sendo.com/dev>

#### ***Motorola***

<http://idenphone.motorola.com>

#### ***Серия 60***

<http://www.series60.com>

#### ***UIQ***

<http://www.uiq.com/developer>

#### ***NTTDoComо***

<http://www.nttdocomo.com>

#### ***SUN Microsystems***

<http://developer.javasun.com>

*Perl*

<http://www.activeperl.com>

**Интегрированные средства  
разработки приложений**

***Borland***

<http://www.borland.com>

***Metrowerks***

<http://www.metrowerks.com>

**Компании**

***Arima***

<http://www.arima.com>

***BenQ***

<http://www.benq.com>

***Motorola***

<http://www.motorolacom>

***Nokia***

<http://www.forum.nokia.com>

***Sony Ericsson***

<http://www.sonyericsson.com>

***Siemens***

<http://www.siemens-mobile.com>

***Sendo***

<http://www.sendo.com>

***Samsung***

<http://www.samsung.com>

***Sanyo***

<http://www.sanyo.com>

***Panasonic***

**<http://www.pSION.com>**

**Lenovo**

<http://www.legendgrp.com>

**LG**

<http://www.lge.com>

**Fujitsu**

<http://www.fujitsu.com>

**Тематические сайты**

<http://www.newlc.com>

<http://www.symbianone.com>

<http://www.allaboutsymbian.com>

[http://www.symbianopen  
source.com](http://www.symbianopensource.com)

<http://www.yoursymbian.com>

[http://www.wirelessdev  
net.com/symbian](http://www.wirelessdevnet.com/symbian)

[http://symbian.infosync  
world.com](http://symbian.infosyncworld.com)

[http://www.symbiandiar  
ies.com](http://www.symbiandiaries.com)

**Русскоязычные сайты**

<http://www.series60.ru>

<http://www.mworld.ru>

<http://www.3230.com.ru>

<http://ncp.alfaspace.net>

[http://www.dimonvideo.  
ru](http://www.dimonvideo.ru)

<http://wap.multifora.ru>

<http://symbianz.ru>

<http://smart60.ru>

<http://all60.com>



## Приложение 4. Обзор компакт-диска

Компакт-диск содержит все исходные коды к демонстрационным примерам, рассмотренным в книге и набор SDK для платформ UIQ, серии 60, серии 80 и серии 90 компаний Symbian, Sony Ericsson и Nokia. Структура папок компакт-диска:

- \Code - содержит все демонстрационные примеры с исходными кодами, рассмотренными в книге.
- \Nokia - в папке находится программное обеспечение компании Nokia:
  - G s60\_sdk\_21b\_cw - инструментальные средства разработчика серии 60 версии 2.1;
  - Series\_80\_SDK\_1\_0\_1\_b - инструментальные средства разработчика серии 80;
  - S90\_1\_0\_Beta2\_for\_Symbian\_OS - инструментальные средства разработчика серии 90.
- O \Sony Ericsson - в папке находится программное обеспечение компании Sony Ericsson:
  - uiq\_2\_1\_sdk\_winscw - инструментальные средства разработчика для UIQ2.1;
  - uiq21\_updatel\_winscw - обновление к инструментальным средствам разработчика UIQ 2.1;
  - tm\_symbiandevlopmentforp800p900\_v2 - архив WinZip с дополнительными библиотеками и эмуляторами для телефонов Sony Ericsson P800 и Sony Ericsson P900.
  - \Documentation - различная документация и исходные коды, связанные с операционной системой Symbian.

## Список используемых источников

1. Документация с сайта, компании Symbian Ltd.  
<http://www.symbiandevnet.com>  
<http://www.symbian.com>
2. Документация с сайта компании Sony Ericsson.  
<http://www.sonyericsson.com/developer>
3. Документация с сайта компании Siemens.  
<http://www.siemens-mobile.com/developer>
4. Документация с сайта компании Nokia.  
<http://www.forum.nokia.com>
5. Документация к SDK Series 60, 80 и 90.
6. Документация к SDK UIQ 2.0 и 2.1.
7. Документация к среде программирования Metrowerks CodeWarrior for Symbian.
8. Документация к среде программирования Borland C++ BuilderX Mobile Studio.

# Предметный указатель

## **В**

**Вкладки**, 22

**Версии Symbian OS**, 19

**Деструктор**,  
**134** **Директива**  
#define, 132  
#endif, 132  
#ifndef, 132

## **И**

**Иконка**,  
**158**

**Интернет**  
GPRS, 23  
WAP, 23  
точка доступа, 23

**Интерфейс**  
графический пользовательский  
(GUI), 127  
графический устройства (GDI), 127  
пользователя (UI), 128

## **К**

**Кисть**, 234

**Класс**  
App Ui, 128 App View, 128  
Application, 23 CAknApplication,  
127 CAknAppUi, 137  
CAknCaleMonthlyStyleGrid, 192  
CAknDocument, 134  
CAknDoubleLargeStyleListBox, 192  
CAknDoubleNumberStyleListBox,  
191 CAknDoubleStyle2ListBox, 191  
CAknDoubleStyleListBox, 191

CAknDoubleTimeStyleListBox, 191  
CAknErrorNote, 179  
CAknExSettingListContainer, 217  
CAknExSettingListItemData, 217  
CAknExSettingListListBox, 219  
CAknExSettingListView, 216  
CAknGMSStyleGrid, 192  
CAknInformationNote, 80  
CAknPinbStyleGrid, 192  
CAknSingleGraphicsHeadingStyleBox,  
191  
CAknSingleGraphicsStyleListBox, 191  
CAknSingleLargeStyleListBox, 191  
CAknSingleNumberHeadingStyleListBo  
x,  
191  
CAknSingleNumberStyleListBox, 191  
CAknSingleStyleListBox, 191  
CAknWarningNote, 179 CApaDocument,  
132 CCoeControl, 141 CFbsBitmap, 254  
CFont, 243 CTest Application, 131  
CTestAppUi, 136 CTestAppView, 140  
CTestDocument, 133 Document, 46 TBuf,  
148 TFileName, 254 TPoint, 229 TRect,  
229 TSize, 229 **Ключевое слово** AIF, 200  
break, 44 class, 42 ENUM, 146 LANG,  
150 MENU\_BAR, 170 MENUITEM, 176  
menupane, 208 MENU TITLE, 175

NAME, 55  
PRJ\_MMPFILES, 149  
RESOURCE, 151  
titles, 175

#### **Константа**

caption, 146  
IP\_EDITOR\_MAX\_FIELD\_VALUE,  
205  
IP\_EDITOR\_MIN\_FIELD\_VALUE,  
205  
shortcaption, 147

#### **Конструктор**

BaseConstructL(), 139  
двухфазный, 141

## **Л**

**Локализация**, 181

## **М**

**Меню**, 40

**Метрика**, 246

#### **Оператор**

case, 57  
default, 140  
switch, 58

## **П**

#### **Панель**

контроля, 21  
состояния, 21

#### **Паника**, 139

#### **Перечисляемый тип**

TAknExSettingItem, 202  
TAknExSettingItems, 203  
TAknExSettingListCba, 203  
TAknExSettingListMenuCommands, 202

#### **Платформа UIQ**, 15

Application Picker, 167  
Menu bar, 167  
Status bar, 51  
Toolbar, 42  
клиентская, 143  
серия 60, 152  
Control Pane, 21  
Main Pane, 164

#### **Status Pane**

Context Pane, 168  
Indicator Pane, 169  
Navigation Pane, 169  
Signal Pane, 168

#### **Title Pane**, 169 **Платформа**

**Java 2 ME** виртуальная Java-  
машина, 256  
интерфейс

Choice, 267  
CommandListener, 261

#### **класс**

Alert, 267  
Canvas, 271  
ChoiceGroup, 267  
Command, 261  
CustomItem, 269  
DateField, 268  
Display, 261  
Font, 267  
Form, 272  
GameCanvas, 272  
Gauge, 269  
Graphics, 271  
ImageItem, 269  
Layer, 273  
LayerManager, 273  
List, 267  
MIDlet, 259  
MyMidlet, 260  
Spacer, 269  
Sprite, 273  
StringItem, 268  
TextBox, 266  
TextField, 268  
Ticker, 270  
TiledLayer, 273

#### **ключевое слово**

new, 262

#### **константа**

EXIT, 262

#### **конфигурация CLDC**, 257

#### **метод**

commandAction(), 264  
destroyApp(), 260  
getDisplay(), 262  
pauseApp(), 259  
setCommandListener(), 263  
setCurrent(), 263  
startApp(), 259

#### **мидлет**, 259

- пользовательский интерфейс
  - высокоуровневый, 265
  - низкоуровневый, 270
- профиль MIDP, 257 среда
- J2ME Wireless Toolkit
  - компиляция проекта, 274
  - создание проекта, 274
  - тестирование приложения, 274
  - упаковка программы, 274
  - файл
    - JAD, 273
    - JAR, 273
- фрейм, 272
- Поток**, 134

### **Рабочий стол**, 22

#### **Ресурс**

- AIF\_DATA, 159
- caption\_list, 159

### **Среда Code Warrior**

- запуск, 38
- игнорируемые папки, 54
- импорт проекта, 65 кнопки
  - Compile, 47
  - Copy, 47
  - Cut, 47
  - Debug, 47
  - Errors and Warnings, 47
  - Find Next, 47
  - Find, 47
  - Make, 47
  - New Text File, 47
  - New, 47
  - Open, 47
  - Paste, 47
  - Preferences, 47
  - Redo, 47
  - Replace Selection, 47
  - Save, 47
  - Stop Build, 47
  - Undo, 47
  - WINSW UDEB Settings, 47
- компиляция, 67 меню Debug
  - Attach to Process, 45

- Break, 44
- Break on C++ Exception, 45
- Break on Java Exceptions, 45
- Change Program Counter, 44
- Clear All Breakpoint, 45
- Clear All Watchpoint, 45
- Clear Eventpoint Clear Log
  - Point, 45
  - Clear Script Point, 45
  - Clear Skip Point, 45
  - Clear Sound Point, 45
  - Clear Trace Collection off, 45
  - Clear Trace Collection on, 45
  - Connect, 45
- Disable Eventpoint Disable Log
  - Point, 45
  - Disable Script Point, 45
  - Disable Skip Point, 45
  - Disable Sound Point, 45
  - Disable Trace Collection off, 45
  - Disable Trace Collection on, 45
- Enable Breakpoint (Ctrl+F9), 45
- Enable Eventpoint Enable Log
  - Point, 45
  - Enable Script Point, 45
  - Enable Skip Point, 45
  - Enable Sound Point, 45
  - Enable Trace Collection off, 45
  - Enable Trace Collection on, 45
- Enable/Disable Watchpoint, 45
- Hide Breakpoints, 45
- Kill (Shift+F5), 44
- Restart (Ctrl+Shift+F5), 44
- Run to Cursor, 44
- Set Breakpoint (F9), 44
- Set Eventpoint Set Log Point, 44
- Set Script Point, 44
- Set Skip Point, 44
- Set Sound Point, 44
- Set Trace Collection off, 45
- Set Trace Collection on, 45
- Set/Clear Breakpoint, 45
- Set/Clear Watchpoint, 45
- Step Into (F11), 44
- StepOut(Shift+F11),44
- Step Over (F10), 44
- Symbian Device, 45
- Target Server, 45
- меню Edit
  - Balance (Ctrl+B), 41

- Code Completion, 41
- Commands and Key Bindings, 41
- Copy (Ctrl+C), 40
- Cut (Ctrl+X), 40
- Delete (Delete), 41
- Get Next Completion (Alt+/,), 41
- Get Previous
  - Completion (Alt+Shift+/,), 41
- Paste (Ctrl+V), 40
- Preference, 41
- Redo (Ctrl+Shift+Z), 40
- Select All (Ctrl+A), 41
- Shift Left (Ctrl+[,), 41
- Shift Right (Ctrl+]), 41
- Symbian Environments, 41
- Undo (Ctrl+Z), 40
- Version Control Settings, 41
- WINSW UDEB Settings (Alt+F7), 41
- меню File
  - Close (Ctrl+W), 40
  - Close Workspace, 40
  - Exit, 40
  - Export Project, 40
  - Find and Open (Ctrl+D), 40
  - Import Project, 40
  - Import Project From .mmp File, 40
  - New (Ctrl+Shift+N), 40
  - Open (Ctrl+O), 40
  - Open Recent, 40
  - Open Workspace, 40
  - Page Setup, 40
  - Print (Ctrl+P), 40
  - Revert, 40
  - Save (Ctrl+S), 40
  - Save A Copy As, 40
  - Save All (Ctrl+Shift+S), 40
  - Save As, 40
  - Save Workspace As, 40
- меню Help
  - About Metrowerks CodeWarrior, 46
  - Code Warrior Help, 46
  - Index, 46
  - Licensee Authorization, 46
  - Metrowerks Website, 46
  - Search, 46
  - Symbian Release Note, 46
- меню Project
  - Check Syntax, 43
  - Add, 43
  - Add Files, 43
  - Compile (Ctrl+F7), 43
  - Create Design, 43
  - Create Group, 43
  - Create Target, 43
  - Debug (F5), 43
  - Disassemble (Ctrl+Shift+F7), 43
  - Make (F7), 43
  - Precompile, 43
  - Re-search for Files, 43
  - Reset Project Entry Patch, 43
  - Rmove Object Code (Ctrl+-), 43
  - Run (Ctrl+F5), 44
  - Set Default Project, 44
  - Set Default Target
    - ARMIUDEB, 44
    - ARMI UREL, 44
    - Build All, 44
    - THUMB UDEB, 44
    - THUMB UREL, 44
    - WINSW UDEB, 44
    - WINSW UREL, 44
  - Stop Build (Ctrl+Break), 43
  - Synchronize Modification Date, 43
  - меню Search
    - Apply Difference, 43
    - Compare Files, 42
    - Enter Find String (Ctrl+E), 42
    - Find (Ctrl+F), 42
    - Find Definition (Ctrl+'), 42
    - Find in Files (Ctrl+Shift+M), 42
    - Find in Next File (Ctrl+T), 42
    - Find Next (F3), 42
    - Find Selection (Ctrl+F3), 42
    - Go Back (Ctrl+Shift+B), 42
    - Go Forward (Ctrl+Shift+F), 42
    - Go to Line (Ctrl+G), 42
    - Replace (Ctrl+H), 42
    - Replace All, 42
    - Replace and Find Next (Ctrl+L), 42
    - Replace Selection (Ctrl+=), 42
    - Unapply Difference, 43
  - меню View
    - Breakpoints, 42
    - Browser Contents, 42
    - Build Progress, 42
    - Class Browser (Alt+F12), 42
    - Class Hierarchy, 42
    - Command Windows, 42
    - Errors and Warnings (Ctrl+I), 42
    - Expressions, 42
    - Global Variables, 42
    - Object Inspector, 42

- Processes, 42 Project
- Inspector, 42
- Registers, 42
- Symbolics, 42
- Toolbar
  - Clear Main Toolbar, 42
  - Clear Window Toolbar, 42
  - Hide Main Toolbar, 42
  - Hide Window Toolbar, 42
  - Reset Main Toolbar, 42
  - Reset Window Toolbar, 42
- меню Window Cascade, 46
- Close (Ctrl+W), 45 Close All
- Editors Document
  - (Ctrl+Shift+W), 46 Tile
- Horizontally, 46 Tile Vertically, 46
- настройки дополнительные, 53
- компиляции, 87 компоновки проекта, 52 появления окон, 62 представления переменных, 61 свойств текстового редактора, 58 цвета текста, 60
  - шрифта и позиции табуляции, 59
- неисправности в дополнительно подключаемых модулях, 54
- окно IDE Preferences кнопки
  - Apply, 52
  - Cancel, 52
  - Export Panel, 52
  - Factory Settings, 51
  - Import Panel, 51
  - ОК, 52
- Revert, 51 окно
- Workspace
- вкладки
  - Files, 47
  - Link Orders, 47
  - Target, 47
- кнопки
  - Debug, 49
  - Make, 49
  - Project Inspector, 49
  - Run, 49
  - Synchronize Modification Date, 48
  - WINSW UDEB Settings, 48
- контекстное меню
  - Add Files, 49 Check Syntax, 49
  - Compile, 49 Compile If Dirty, 49
  - Create Group, 49 Disassemble, 49
  - Open in Windows Explorer, 49
  - Preprocess, 49 Remove, 49
- окно текстового редактора
  - кнопки
    - Document Settings, 49
    - Functions, 49 Header
    - Files, 49 Markers, 49
    - Version Control, 49
  - контекстное меню
    - Compile, 50 Disassemble, 50
    - Find and Open File, 50
    - Paste, 50 Preprocess, 50
    - Redo, 50
    - Set Breakpoint, 50 Set Eventpoint Set Log Point, 50
    - Set Script Point, 50 Set Skip Point, 50
    - Set Sound Point, 50
    - Set Trace Collection off, 51
    - Set Trace Collection on, 51
    - Set Software Breakpoint, 51
    - Undo, 50
  - цвет ключевых слов, 49
  - платформа, 66
- стиль написания исходного кода, 117
- установочный пакет, 69 **Среда C++**
- BuilderX** вкладки
  - Class Browser, 82 File Browser, 82 Project Content, 82 запуск, 75
  - импорт проекта, 86
  - кнопки Back, 82 Close, 81 Copy, 81 Cut, 81
  - Debug Project, 82
  - Find Classes, 81

- Find, 81
- Forward, 82
- Go To, 82
- Help Topics, 82
- Make Project, 81
- Messages, 81
- New, 81
- Open File, 81
- Paste, 81
- Print, 81
- Redo, 81
- Reopen, 81
- Replace, 81
- Run Project, 82
- Save All, 81
- Save, 81
- Search Again, 81
- Undo, 81
- Выбор версии, 81
- Выбор платформы, 81
- 81 компиляция, 87
- меню Edit
  - Copy (Ctrl-C), 77
  - Cut (Ctrl-X), 77
  - Delete, 77
  - Format All, 77
  - Paste (Ctrl-V), 77
  - Redo (Ctrl+Sift-Z), 77
  - Select All (Ctrl-A), 77
  - Sync Edit (Ctrl+Shift-J), 77
  - Undo (Ctrl-Z), 77
- меню File
  - Close (Ctrl+Shift-F4), 77
  - Close All (Ctrl+Alt-F4), 77
  - Close All Except, 77
  - Close File (Ctrl-F4), 77
  - Close Project, 77
  - Compare Files, 77
  - Exit, 77
  - New (Ctrl-N), 76
  - New File, 76
  - Open File (Ctrl-O), 76
  - Open Project, 76
  - Page Layout, 77
  - Print, 77
  - Rename, 77
  - Reopen, 77
  - Revert, 77
  - Save (Ctrl-S), 77
  - Save All (Ctrl+Shift-A), 77
  - Save As, 77
  - Save Copy, 77
  - Save Project, 77
  - Save Project As, 77 меню
- Help
  - About C++ BuilderX, 81
  - C++ BuilderX Home Page, 81
  - Help Topics, 81
  - Release Notes, 81
- 81 меню Project
  - Add Files, 79
  - Default Project Properties, 79
  - Link Project, 79
  - Link Project Group, 79
  - Make (Ctrl+Shift-F9), 79
  - Make Project (Ctrl-F9), 79
  - Make Project Group, 79
  - New Directory View, 79
  - New Folders, 79
  - Project Group Properties, 79
  - Project Properties, 79
  - Rebuild, 79
  - Rebuild Project, 79
  - Rebuild Project Group, 79
  - Refresh, 79
  - Remove from Project, 79
  - Rename, 79
- меню Run
  - Add Breakpoint, 79
  - Configurations, 79
  - Debug Project (Shift-F9), 79
  - Inspect (Alt-F5), 79
  - Pause Program, 79
  - Reset Program (Ctrl-F2), 79
  - Resume Program, 79
  - Run Project (F9), 79
  - Run to Cursor (F4), 79
- меню Search
  - Find (Ctrl-F), 78
  - Find Classes (Ctrl-Minus), 78
  - Find in Patch (Ctrl-P), 78
  - Go to Adress (Ctrl+Shift-G), 78
  - Go to Line (Ctrl-G), 78
  - Incremental Saerch (Ctrl-E), 78
  - Replace (Ctrl-R), 78
  - Replace in Patch, 78
  - Search Again (F3), 78
- меню Team
  - Add, 80
  - Commit, 80
  - Commit Browser, 80
  - Configure Version Control, 80



- Create Local Repository, 80
- CVS Administration, 80
- File Status, 80
- Place Project into CVS, 80
- Remove, 80
- Select Project VCS, 80
- Sync Project Settings, 80
- Update, 80
- Update Project, 80
- меню
- Tools Build Tools, 80
  - Configure File Associations, 80
  - Configure Tools, 80 -
  - Editor Options, 80
  - Exported Settings Group, 80
  - IDE Options, 80
  - Reload Toolset, 80
  - Reset GDB, 80
  - Symbian SDK Configuration, 80
  - меню View
    - Back (Ctrl+Alt-Left), 78
    - Breakpoints (Ctrl+Alt-B), 79
    - Content (Ctrl+Alt-C), 78
    - Context Menu (Ctrl-F10), 78
    - CPU View (Ctrl+Alt-U), 79
    - Debug Event Log, 79
    - Forward (Ctrl+Alt-Right), 79
    - Hide All (Ctrl+Alt-Z), 78
    - History, 78
    - Messages (Ctrl+Alt-M), 78
    - Project (Ctrl+Alt-P), 78
    - Remove All Message Tabs, 78
    - Status Bar, 78
    - Structure (Ctrl+Alt-S), 78
    - Switch Viewer to History, 78
    - Toolbars
      - Build, 78
      - Configuration, 78
      - Editing, 78
      - File, 78
      - Help, 78
      - Hide All, 78
      - Navigation, 78
      - Platform, 78
      - Run/Debug, 78
      - Search, 78
      - Show All, 78
      - меню
- Window Cascade
- Browsers, 80
- Minimize
- Browsers, 80
- New
- Browser, 80
  - Restore Browsers, 80
  - Select Browser, 81
  - Select Message, 81
  - Tile Browsers Horizontally, 81
  - Tile Browsers Vertically, 81
- меню Wizards
  - Build Options, 80
  - New Build Configuration, 80
  - Target Settings, 80
- тестирование программы, 87
- установочный пакет, 88
- Сервисы системы базовые**
  - Fileserver, 109
  - Low Level Libraries, 110
  - графические, 110
  - мультимедиа, **111**
  - передачи данных, 111
  - пользовательские
    - PIM, 112
    - передача сообщений, 112
    - синхронизации данных, 112
  - связи, 118
  - связи с компьютером, **НО**
- Структура**
- RESOURCE CAPTION\_DATA **147**
- Список**, 187
- Grid
  - Markable List, 192
  - Menu List, 192
  - Multiselection List, 192
  - Selection List, 192
  - создание, 192
- Setting
  - Binary switch, 194
  - Date editor, 194
  - Enumerated text, 195
  - IP editor, 198
  - Password editor
    - alphabetic, 199
    - numeric, 199
  - Slider control, 195
  - Text editor, 196
  - Time editor, 198
  - Volume control, 195
  - создание, 195
- Vertical List
  - Markable, 189
  - Menu, 188
  - Multiselection, 189

Selection, 188  
создание, 19(Б)

### **Тип**

TTestIds, 146  
EAknCtNumericSecretEditor,  
213 EEikCtSecretEd, 213 LBUF,  
190  
num\_code\_chars, 213  
num\_letters, 213

### **Установка программ**

через Диспетчер, 27 через  
Приложения, 26 через Файловый  
менеджер, 29 **Установка SDK**  
Sony Ericsson, 15  
серии 80,15  
серии 90,15 обновление  
установленных  
компонентов, 94 **Уникальный  
идентификатор** UID1,150  
UID2,150 UID3,150 платформы  
(Platform UID), 152

## **Ф**

### **Файловая система, 82**

#### **Функции**

не уходящего вида, 120  
уходящего вида, 119  
**Функция** ActivateL(), 143  
AddToStackLO, 139  
AppDllUid(), 156 Clear(), 143  
ClientRect(), 139  
ConstructL(), 139  
CreateAppUiL(), 134  
CreateDocumentLQ, 132

CreateWindowL(), 143  
DrawRect(), 144  
E32D11(), 144  
Exit(), 120  
HandleCommandL(), 145  
NewApplication(), 144  
NewL(), 120  
NewLC(), 120  
Panic(), 139  
RestoreL(), 136  
Shrink(), 143  
StoreL(), 136  
SystemGc(), 143  
CreateListBoxL(), 219  
CreateSettingItemL(), 220  
DisplayNextOutlineL(), 219  
DisplayPreviousOutlineL(), 219  
DoActivateL(), 219  
DoDeactivate(), 219  
HandleApplicationSpecificEventL(),  
186  
HandleForegroundEventL(), 186  
HandleKeyEventL(), 186  
HandleSystemEventL(), 186  
OfferKeyEventL(), 186  
SwapContainerL(), 219  
TUidId(),219  
CreateBitmapL(), 255  
DiscardFont(), 244  
DrawBitmap(), 255  
DrawEllipse(), 237  
DrawLine(), 228  
DrawPie(), 239  
DrawText(), 245  
Height(), 251  
Load(), 245  
NormalFont(), 245  
Reset(), 252  
SetPenColor(), 252  
SetStrikethroughStyle(), 252  
SetUnderlineStylG(), 252  
SetXY(),  
230 Width(),  
251

## **Я**

**Ядро системы, 109**

Книги издательства «ДМК Пресс» можно заказать в торгово-издательском холдинге «АЛЬЯНС-КНИГА» наложенным платежом, выслав открытку или письмо по почтовому адресу: **123242, Москва, а/я 20** или по электронному адресу: **post@abook.ru**.

При оформлении заказа следует указать адрес (полностью), по которому должны быть высланы книги; фамилию, имя и отчество получателя. Желательно также указать свой телефон и электронный адрес.

Эти книги вы можете заказать и в Internet-магазине: **www.abook.ru**.

Оптовые закупки: тел. **(095) 258-91-94, 258-91-95**; электронный адрес **abook@abook.ru**.

Горнаков Станислав Геннадьевич

Symbian OS Программирование  
мобильных телефонов на C++ и Java 2 ME

Главный редактор *Мовчан Д. А.*  
**dm@dmkpress.ru**

Литературный редактор *Стукалова О.*  
Верстка *Чаянова А. А.* Дизайн  
обложки *Мовчан А. Г.*

Подписано в печать 26.06.2005. Формат 70x100  $\frac{1}{16}$ .

Гарнитура «Петербург». Печать офсетная.

Усл. печ. л. 35,75. Тираж 2000 экз.

Электронный адрес издательства: [www.dmkpress.ru](http://www.dmkpress.ru)



Автор книги - Горнаков Станислав Геннадьевич – программист с многолетним стажем, автор книг: «Программирование мобильных телефонов на Java 2 Micro Edition», «DirectX 9. Уроки программирования на C++», «Инструментальные средства для отладки и программирования шейдеров в DirectX и OpenGL».

## SYMBIAN OS

# Программирование мобильных телефонов на C++ и Java 2 ME

Горнаков С. Г.

УРОВЕНЬ ПОЛЬЗОВАТЕЛЯ



- ✓ начинающий
- ✓ средний
- опытный
- профессиональный

Создание мобильных приложений для операционной системы Symbian - очень сложная и трудоемкая задача. Эта книга познакомит вас с основами программирования для Symbian OS на языке программирования C++, а одна из глав посвящена программированию Java 2 ME приложений. Темы, рассматриваемые в книге весьма разносторонние – это интегрированные среды программирования CodeWarrior for Symbian, C++ BuilderX Mobile Studio, инструментальные средства разработчика SDK от Symbian, Sony Ericsson и Nokia для платформ UIQ, серии 60, серии 80 и серии 90. Большой объем информации освещает вопросы, связанные с программной архитектурой операционной системы, основными идиомами программирования в Symbian OS, структурой и созданием GUI приложения, локализацией, работой с меню, элементами пользовательского интерфейса, графикой, изображениями, созданием инсталляционного пакета. Книга будет интересна широкому кругу читателей, желающим самостоятельно изучить программирование для операционной системы Symbian на языке C++.



Компакт-диск содержит все исходные коды к демонстрационным примерам, рассмотренным в книге и набор SDK для платформ UIQ, серии 60, серии 80 и серии 90 от компаний Symbian, SonyEricsson и Nokia.

**АМК**  
Альянс-Книга

Оптовая продажа

“Альянс-книга”  
тел./факс: (095) 258-9195

Internet-магазин

[www.abook.ru](http://www.abook.ru)

Книга – почтой

Россия, 123242, Москва, а/я 20  
e-mail: [post@abook.ru](mailto:post@abook.ru)

ISBN 5-94074-030-8



9 785940 740308