

# Qt 4.8

## ПРОФЕССИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ НА C++



- Кроссплатформенная реализация приложений для Windows, Linux и Mac OS X
- Программирование графики, мультимедиа, веб-приложений, баз данных, сети, таймера, многопоточности, XML
- 222 завершенные программы
- Создание пользовательских интерфейсов с помощью Qt Quick и QML



Материалы  
на [www.bhy.ru](http://www.bhy.ru)

**Наиболее  
полное  
руководство**

**В ПОДЛИННИКЕ®**

УДК 681.3.068+800.92С++  
ББК 32.973.26-018.1  
Ш68

### **Шлее М.**

Ш68 Qt 4.8. Профессиональное программирование на С++. — СПб.: БХВ-Петербург, 2012. — 912 с.: ил. — (В подлиннике)

ISBN 978-5-9775-0736-3

Книга посвящена разработке приложений для Windows, Linux и Mac OS X с использованием библиотеки Qt версии 4.8. Подробно рассмотрены возможности, предоставляемые этой библиотекой, и описаны особенности, выгодно отличающие ее от других библиотек. Описана интегрированная среда разработки Qt Creator. Показано создание пользовательских интерфейсов с помощью Qt Quick и QML. Книга содержит исчерпывающую информацию о классах Qt 4, а также даны практические рекомендации их применения, проиллюстрированные на большом количестве подробно прокомментированных примеров. Проекты примеров из книги размещены на сайте издательства.

*Для программистов*

УДК 681.3.068+800.92С++  
ББК 32.973.26-018.1

#### **Группа подготовки издания:**

Главный редактор	<i>Екатерина Кондукова</i>
Зам. главного редактора	<i>Игорь Шишигин</i>
Зав. редакцией	<i>Григорий Добин</i>
Редактор	<i>Юрий Рожко</i>
Компьютерная верстка	<i>Ольги Сергиенко</i>
Корректор	<i>Зинаида Дмитриева</i>
Дизайн серии	<i>Инны Тачиной</i>
Оформление обложки	<i>Марины Дамбиевой</i>

Подписано в печать 31.03.12.

Формат 70×100<sup>1/16</sup>. Печать офсетная. Усл. печ. л. 73,53.

Тираж 2000 экз. Заказ №

"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Первая Академическая типография "Наука"  
199034, Санкт-Петербург, 9 линия, 12/28

ISBN 978-5-9775-0736-3

© Шлее М., 2012  
© Оформление, издательство "БХВ-Петербург", 2012

# Оглавление

<b>Предисловие Маттиаса Эттриха</b> .....	<b>3</b>
<b>Благодарности</b> .....	<b>5</b>
<b>Предисловие</b> .....	<b>6</b>
Структура книги.....	6
<b>Введение</b> .....	<b>16</b>
<b>ЧАСТЬ I. ОСНОВЫ QT</b> .....	<b>25</b>
<b>Глава 1. Обзор иерархии классов Qt</b> .....	<b>27</b>
Первая программа на Qt.....	27
Модули Qt.....	28
Пространство имен Qt .....	30
Модуль QtCore .....	30
Модуль QtGui .....	31
Модуль QtNetwork .....	32
Модуль QtXml .....	32
Модуль QSql.....	33
Модуль QtOpenGL .....	33
Модуль QtWebKit.....	33
Модуль QtSvg .....	33
Модуль Qt3Support.....	33
Резюме .....	33
<b>Глава 2. Философия объектной модели</b> .....	<b>34</b>
Механизм сигналов и слотов .....	36
Сигналы .....	39
Слоты .....	41
Соединение объектов.....	42
Разъединение объектов.....	46
Организация объектных иерархий .....	47
Метаобъектная информация .....	49
Резюме .....	50

<b>Глава 3. Работа с Qt</b> .....	<b>51</b>
Интегрированная среда разработки IDE .....	51
Qt Assistant .....	51
Работа с cmake .....	51
Рекомендации для проекта с Qt .....	55
Метаобъектный компилятор MOC .....	56
Компилятор ресурсов RCC .....	57
Структура Qt-проекта .....	57
Методы отладки .....	58
Отладчик GDB (GNU Debugger) .....	59
Прочие методы отладки .....	62
Глобальные определения Qt .....	63
Информация о библиотеке Qt .....	65
Резюме .....	66
<b>Глава 4. Библиотека контейнеров</b> .....	<b>67</b>
Контейнерные классы .....	68
Итераторы .....	70
Итераторы в стиле Java .....	70
Итераторы в стиле STL .....	71
Ключевое слово <i>foreach</i> .....	73
Последовательные контейнеры .....	73
Вектор <i>QVector&lt;T&gt;</i> .....	74
Массив байтов <i>QByteArray</i> .....	75
Массив битов <i>QBitArray</i> .....	76
Списки <i>QList&lt;T&gt;</i> , <i>QLinkedList&lt;T&gt;</i> .....	76
Стек <i>QStack&lt;T&gt;</i> .....	78
Очередь <i>QQueue&lt;T&gt;</i> .....	79
Ассоциативные контейнеры .....	79
Словари <i>QMap&lt;K,T&gt;</i> , <i>QMultiMap&lt;K,T&gt;</i> .....	80
Хэши <i>QHash&lt;K,T&gt;</i> и <i>QMultiHash&lt;K,T&gt;</i> .....	82
Множество <i>QSet&lt;T&gt;</i> .....	82
Алгоритмы .....	84
Сортировка .....	85
Поиск .....	85
Сравнение .....	86
Заполнение значениями .....	86
Строки .....	86
Регулярные выражения .....	88
Произвольный тип <i>QVariant</i> .....	90
Модель общего использования данных .....	91
Резюме .....	92
<b>ЧАСТЬ II. ЭЛЕМЕНТЫ УПРАВЛЕНИЯ</b> .....	<b>93</b>
<b>Глава 5. С чего начинаются элементы управления</b> .....	<b>95</b>
Класс <i>QWidget</i> .....	95
Размеры и координаты виджета .....	98
Механизм закулисного хранения .....	99
Установка фона виджета .....	99
Изменение указателя мыши .....	100

Стек виджетов .....	103
Рамки.....	103
Виджет видовой прокрутки.....	104
Резюме .....	106
<b>Глава 6. Управление автоматическим размещением элементов.....</b>	<b>107</b>
Менеджеры компоновки (layout managers).....	107
Горизонтальное и вертикальное размещения .....	109
Класс <i>QBoxLayout</i> .....	109
Горизонтальное размещение <i>QHBoxLayout</i> .....	111
Вертикальное размещение <i>QVBoxLayout</i> .....	112
Вложенные размещения .....	113
Табличное размещение <i>QGridLayout</i> .....	114
Порядок следования табулятора .....	120
Разделители <i>QSplitter</i> .....	120
Резюме .....	121
<b>Глава 7. Элементы отображения .....</b>	<b>122</b>
Надписи .....	122
Индикатор прогресса .....	126
Электронный индикатор.....	129
Резюме .....	131
<b>Глава 8. Кнопки, флажки и переключатели.....</b>	<b>132</b>
С чего начинаются кнопки. Класс <i>QAbstractButton</i> .....	132
Установка текста и изображения .....	132
Взаимодействие с пользователем .....	132
Опрос состояния.....	133
Кнопки .....	133
Флажки.....	136
Переключатели.....	137
Группировка кнопок .....	138
Резюме .....	141
<b>Глава 9. Элементы настройки.....</b>	<b>143</b>
Класс <i>QAbstractSlider</i> .....	143
Изменение положения .....	143
Установка диапазона .....	143
Установка шага .....	144
Установка и получение значений .....	144
Ползунок.....	144
Полоса прокрутки .....	146
Установщик .....	147
Резюме .....	149
<b>Глава 10. Элементы ввода.....</b>	<b>150</b>
Однорочное текстовое поле.....	150
Редактор текста .....	152
Запись в файл.....	155
Расцветка синтаксиса (syntax highlighting).....	155

С чего начинаются виджеты счетчиков .....	162
Счетчик .....	162
Элемент ввода даты и времени .....	163
Проверка ввода.....	164
Резюме .....	165
<b>Глава 11. Элементы выбора .....</b>	<b>167</b>
Простой список .....	167
Вставка элементов .....	167
Выбор элементов пользователем.....	169
Изменение элементов пользователем.....	169
Режим пиктограмм.....	169
Сортировка элементов.....	170
Иерархические списки.....	171
Сортировка элементов.....	174
Таблицы .....	174
Выпадающий список .....	176
Вкладки.....	177
Виджет панели инструментов.....	178
Резюме .....	179
<b>Глава 12. Интервью или модель-представление.....</b>	<b>180</b>
Концепция .....	181
Модель .....	181
Представление .....	183
Выделение элемента .....	184
Делегат .....	186
Индексы модели.....	188
Иерархические данные .....	188
Роли элементов .....	192
Создание собственных моделей данных .....	194
Промежуточная модель данных (Proxy model) .....	200
Модель элементарно-базированных классов .....	202
Резюме .....	204
<b>Глава 13. Цветовая палитра элементов управления.....</b>	<b>205</b>
Резюме .....	208
<b>ЧАСТЬ III. СОБЫТИЯ И ВЗАИМОДЕЙСТВИЕ С ПОЛЬЗОВАТЕЛЕМ.....</b>	<b>209</b>
<b>Глава 14. События .....</b>	<b>211</b>
Переопределение специализированных методов обработки событий .....	213
События клавиатуры.....	213
Класс <i>QKeyEvent</i> .....	213
Класс <i>QFocusEvent</i> .....	216
Событие обновления контекста рисования. Класс <i>QPaintEvent</i> .....	216
События мыши .....	217
Класс <i>QMouseEvent</i> .....	217
Класс <i>QWheelEvent</i> .....	221
Методы <i>enterEvent()</i> и <i>leaveEvent()</i> .....	221

Событие таймера. Класс <i>QTimerEvent</i> .....	221
События перетаскивания (drag & drop).....	222
Класс <i>QDragEnterEvent</i> .....	222
Класс <i>QDragLeaveEvent</i> .....	222
Класс <i>QDragMoveEvent</i> .....	222
Класс <i>QDropEvent</i> .....	222
Остальные классы событий.....	222
Класс <i>QChildEvent</i> .....	222
Класс <i>QCloseEvent</i> .....	222
Класс <i>QHideEvent</i> .....	223
Класс <i>QMoveEvent</i> .....	223
Класс <i>QShowEvent</i> .....	223
Класс <i>QResizeEvent</i> .....	223
Реализация собственных классов событий.....	224
Переопределение метода <i>event()</i> .....	225
Сохранение работоспособности приложения.....	228
Резюме .....	229
<b>Глава 15. Фильтры событий.....</b>	<b>230</b>
Реализация фильтров событий.....	230
Резюме .....	233
<b>Глава 16. Искусственное создание событий .....</b>	<b>234</b>
Резюме .....	237
<b>ЧАСТЬ IV. ГРАФИКА И ЗВУК .....</b>	<b>239</b>
<b>Глава 17. Введение в компьютерную графику .....</b>	<b>241</b>
Классы геометрии .....	241
Точка .....	241
Двумерный размер .....	242
Прямоугольник.....	244
Прямая линия .....	244
Полигон.....	245
Цвет.....	245
Класс <i>QColor</i> .....	245
Цветовая модель RGB .....	246
Цветовая модель HSV.....	247
Цветовая модель CMYK.....	248
Палитра .....	249
Предопределенные цвета.....	250
Резюме .....	251
<b>Глава 18. Легенда о короле Артуре и контексте рисования .....</b>	<b>252</b>
Класс <i>QPainter</i> .....	253
Перья и кисти .....	255
Перо.....	255
Кисть .....	256
Градиенты.....	257
Техника сглаживания (Anti-aliasing) .....	258

Рисование .....	259
Рисование точек .....	259
Рисование линий .....	260
Рисование сплошных прямоугольников .....	261
Рисование заполненных фигур .....	261
Запись команд рисования .....	264
Трансформация систем координат .....	264
Перемещение .....	265
Масштабирование .....	266
Поворот .....	266
Скос .....	266
Трансформационные матрицы .....	266
Графическая траектория (painter path) .....	267
Отсечения .....	268
Режим совмещения (composition mode) .....	269
Графические эффекты .....	272
Резюме .....	274
<b>Глава 19. Растровые изображения.....</b>	<b>275</b>
Форматы графических файлов.....	275
Формат BMP .....	275
Формат GIF .....	276
Формат PNG .....	276
Формат JPEG .....	276
Формат XPM.....	276
Контекстно-независимое представление .....	278
Класс <i>QImage</i> .....	278
Класс <i>QImage</i> как контекст рисования .....	285
Контекстно-зависимое представление .....	286
Класс <i>QPixmap</i> .....	286
Класс <i>QPixmapCache</i> .....	287
Класс <i>QBitmap</i> .....	288
Использование масок для <i>QPixmap</i> .....	288
Создание нестандартного окна виджета .....	290
Резюме .....	292
<b>Глава 20. Работа со шрифтами.....</b>	<b>294</b>
Отображение строки .....	296
Резюме .....	299
<b>Глава 21. Графическое представление .....</b>	<b>300</b>
Сцена.....	301
Представление.....	301
Элемент.....	302
События .....	305
Виджеты в графическом представлении.....	311
Резюме .....	313
<b>Глава 22. Анимация.....</b>	<b>314</b>
Класс <i>QMovie</i> .....	314
SVG-графика .....	316



Анимационный движок и машина состояний .....	317
Смягчающие линии .....	320
Машина состояний и переходы .....	324
Резюме .....	327
<b>Глава 23. Работа с OpenGL .....</b>	<b>328</b>
Основные положения OpenGL .....	328
Классы Qt для работы с OpenGL .....	330
Реализация OpenGL-программы .....	330
Разворачивание OpenGL-программ во весь экран .....	333
Графические примитивы OpenGL .....	334
Трехмерная графика .....	337
Резюме .....	341
<b>Глава 24. Вывод на печать .....</b>	<b>342</b>
Класс <i>QPrinter</i> .....	342
Резюме .....	347
<b>Глава 25. Разработка собственных элементов управления .....</b>	<b>348</b>
Примеры создания виджетов .....	348
Резюме .....	353
<b>Глава 26. Элементы со стилем .....</b>	<b>354</b>
Встроенные стили .....	356
Создание собственных стилей .....	360
Метод рисования простых элементов управления .....	361
Метод рисования элементов управления .....	361
Метод рисования составных элементов управления .....	362
Реализация стиля простого элемента управления .....	362
Использование <i>QStyle</i> для рисования виджетов .....	366
Использование каскадных стилей документа .....	366
Основные положения .....	367
Изменение подэлементов .....	368
Управление состояниями .....	369
Пример .....	370
Резюме .....	374
<b>Глава 27. Звук .....</b>	<b>375</b>
Воспроизведение звука .....	375
Проверка возможности воспроизведения .....	376
Пример программы, воспроизводящей звук .....	376
Резюме .....	379
<b>Глава 28. Мультимедиа .....</b>	<b>380</b>
Путешествие к истокам Phonon .....	380
Архитектура Phonon .....	381
Быстрый старт .....	384
Создаем простой медиаплеер .....	386
Резюме .....	389

<b>ЧАСТЬ V. СОЗДАНИЕ ПРИЛОЖЕНИЙ .....</b>	<b>391</b>
<b>Глава 29. Сохранение настроек приложения .....</b>	<b>393</b>
Управление сеансом .....	400
Резюме .....	402
<b>Глава 30. Буфер обмена и перетаскивание .....</b>	<b>403</b>
Буфер обмена .....	403
Перетаскивание .....	404
Реализация drag .....	406
Реализация drop .....	408
Создание собственных типов перетаскивания .....	410
Резюме .....	415
<b>Глава 31. Интернационализация приложения.....</b>	<b>417</b>
Подготовка приложения к интернационализации.....	417
Утилита lupdate .....	419
Программа Qt Linguist .....	420
Утилита lrelease. Пример программы, использующей перевод .....	421
Смена перевода в процессе работы программы.....	423
Завершающие размышления.....	425
Резюме .....	426
<b>Глава 32. Создание меню.....</b>	<b>427</b>
Анатомия меню .....	427
Отрывные меню .....	431
Контекстные меню.....	432
Резюме .....	433
<b>Глава 33. Диалоговые окна.....</b>	<b>434</b>
Правила создания диалоговых окон .....	434
Класс <i>QDialog</i> .....	435
Модальные диалоговые окна .....	435
Немодальные диалоговые окна.....	436
Создание собственного диалогового окна .....	436
Стандартные диалоговые окна .....	440
Диалоговое окно выбора файлов .....	440
Диалоговое окно настройки принтера.....	442
Диалоговое окно выбора цвета .....	443
Диалоговое окно выбора шрифта .....	444
Диалоговое окно ввода .....	445
Диалоговое окно прогресса.....	446
Диалоговые окна мастера.....	447
Диалоговые окна сообщений .....	448
Окно информационного сообщения .....	450
Окно предупреждающего сообщения .....	450
Окно критического сообщения .....	451
Окно сообщения о программе.....	452
Окно сообщения <i>About Qt</i> .....	452
Окно сообщения об ошибке .....	453
Резюме .....	453

<b>Глава 34. Предоставление помощи .....</b>	<b>455</b>
Всплывающая подсказка .....	455
Подсказка "Что это" .....	457
Система помощи (Online Help) .....	458
Резюме .....	461
<b>Глава 35. Создание SDI- и MDI-приложений .....</b>	<b>462</b>
Класс главного окна <i>QMainWindow</i> .....	462
Класс действия <i>QAction</i> .....	463
Панель инструментов .....	464
Доки .....	466
Строка состояния .....	467
Окно заставки .....	469
SDI- и MDI-приложения .....	471
SDI-приложение .....	471
MDI-приложение .....	475
Резюме .....	483
<b>Глава 36. Рабочий стол (Desktop).....</b>	<b>484</b>
Область уведомлений .....	484
Виджет экрана .....	489
Класс сервиса рабочего стола .....	493
Резюме .....	493
<b>ЧАСТЬ VI. ОСОБЫЕ ВОЗМОЖНОСТИ QT .....</b>	<b>495</b>
<b>Глава 37. Работа с файлами, каталогами и потоками ввода-вывода.....</b>	<b>497</b>
Ввод-вывод. Класс <i>QIODevice</i> .....	497
Работа с файлами. Класс <i>QFile</i> .....	499
Класс <i>QBuffer</i> .....	500
Класс <i>QTemporaryFile</i> .....	501
Работа с каталогами. Класс <i>QDir</i> .....	501
Просмотр содержимого каталога .....	502
Информация о файлах. Класс <i>QFileInfo</i> .....	505
Файл или каталог? .....	505
Путь и имя файла .....	506
Информация о дате и времени .....	506
Получение атрибутов файла .....	506
Определение размера файла .....	506
Наблюдение за файлами и каталогами .....	507
Потоки ввода-вывода .....	509
Класс <i>QTextStream</i> .....	509
Класс <i>QDataStream</i> .....	511
Резюме .....	511
<b>Глава 38. Дата, время и таймер .....</b>	<b>513</b>
Дата и время .....	513
Класс даты <i>QDate</i> .....	513
Класс времени <i>QTime</i> .....	515
Класс даты и времени <i>QDateTime</i> .....	516

Таймер.....	516
Событие таймера.....	517
Класс <i>QTimer</i> .....	519
Класс <i>QBasicTimer</i> .....	521
Резюме.....	521
<b>Глава 39. Процессы и потоки.....</b>	<b>522</b>
Процессы.....	522
Потоки.....	525
Приоритеты.....	527
Обмен сообщениями.....	527
Сигнально-слотовые соединения.....	528
Отправка событий.....	532
Синхронизация.....	535
Мьютексы.....	536
Семафоры.....	537
Ожидание условий.....	538
Возникновение тупиковых ситуаций.....	538
Фреймворк <i>QtConcurrent</i> .....	539
Резюме.....	541
<b>Глава 40. Программирование поддержки сети.....</b>	<b>542</b>
Сокетное соединение.....	542
Модель "клиент-сервер".....	543
Реализация TCP-сервера.....	544
Реализация TCP-клиента.....	549
Реализация UDP-сервера и UDP-клиента.....	553
Высокоуровневые классы.....	557
Класс <i>QFtp</i> .....	557
Класс <i>QHttp</i> .....	558
Управляющий доступом к сети.....	559
Блокирующий подход.....	566
Режим прокси.....	569
Резюме.....	569
<b>Глава 41. Работа с XML.....</b>	<b>571</b>
Основные понятия и структура XML-документа.....	571
XML и Qt.....	573
Работа с DOM.....	573
Чтение XML-документа.....	574
Создание и запись XML-документа.....	576
Работа с SAX.....	578
Чтение XML-документа.....	578
Класс <i>QXmlStreamReader</i> для чтения XML.....	581
Использование XQuery.....	583
Резюме.....	586
<b>Глава 42. Программирование баз данных.....</b>	<b>587</b>
Основные положения SQL.....	587
Создание таблицы.....	588
Операция вставки.....	588

Чтение данных.....	588
Изменение данных .....	589
Удаление .....	589
Использование языка SQL в библиотеке Qt .....	589
Соединение с базой данных (второй уровень) .....	591
Исполнение команд SQL (второй уровень) .....	592
Классы SQL-моделей для интервью (третий уровень) .....	595
Модель запроса .....	596
Табличная модель .....	597
Реляционная модель .....	599
Резюме .....	600
<b>Глава 43. Динамические библиотеки и система расширений.....</b>	<b>601</b>
Динамические библиотеки .....	601
Динамическая загрузка и выгрузка библиотеки .....	602
Расширения (plug-ins).....	605
Расширения для Qt.....	605
Поддержка собственных расширений в приложениях .....	607
Создание расширения для приложения.....	610
Резюме .....	613
<b>Глава 44. Совместное использование Qt с платформозависимыми API.....</b>	<b>614</b>
Совместное использование с Windows API.....	616
Совместное использование с Linux .....	618
Совместное использование с Mac OS X .....	618
Системная информация.....	622
Резюме .....	623
<b>Глава 45. Qt Designer. Быстрая разработка прототипов.....</b>	<b>624</b>
Создание новой формы в Qt Designer .....	624
Добавление виджетов .....	627
Компоновка (layout).....	628
Порядок следования табулятора.....	629
Сигналы и слоты .....	630
Использование в формах собственных виджетов .....	632
Использование форм в проектах .....	632
Компиляция .....	634
Динамическая загрузка формы .....	635
Резюме .....	637
<b>Глава 46. Проведение тестов .....</b>	<b>639</b>
Создание тестов .....	640
Тесты с передачей данных .....	643
Создание тестов графического интерфейса.....	645
Параметры для запуска тестов.....	647
Резюме .....	647
<b>Глава 47. WebKit.....</b>	<b>648</b>
Путешествие к истокам .....	649
А зачем?.....	650

Быстрый старт .....	650
Написание простого Web-браузера .....	652
Ввод адресов.....	652
Управление историей.....	652
Загрузка страниц и ресурсов .....	653
Пишем Web-браузер, попытка номер два .....	653
Резюме .....	657
<b>Глава 48. Интегрированная среда разработки Qt Creator .....</b>	<b>658</b>
Первый запуск.....	659
Создаем проект "Hello Qt Creator" .....	659
Пользовательский интерфейс Qt Creator.....	665
Окна вывода.....	666
Окно проектного обозревателя .....	666
Секция компилирования и запуска .....	666
Редактирование текста .....	669
Как подсвечен ваш синтаксис? .....	669
Скрытие и отображение кода .....	670
Автоматическое дополнение кода .....	670
Поиск и замена .....	670
Комбинации клавиш для ускорения работы .....	675
Вертикальное выделение текста .....	675
Автоматическое форматирование текста.....	675
Комментирование блоков.....	675
Просмотр кода методов класса их определения и атрибутов .....	676
Помощь, которая всегда рядом.....	676
Использование стороннего редактора.....	677
Интерактивный отладчик и программный экзорцизм .....	677
Синтаксические ошибки.....	678
Ошибки компоновки.....	679
Ошибки времени исполнения .....	680
Логические ошибки .....	680
Трассировка .....	680
Команда <i>Step Over</i> .....	681
Команда <i>Step Into</i> .....	681
Команда <i>Step Out</i> .....	682
Контрольные точки .....	682
Окно переменных (Local and Watches).....	683
Окно цепочки вызовов (Call Stack).....	684
Резюме .....	684
<b>Глава 49. Рекомендации по миграции программ из Qt3 в Qt4.....</b>	<b>686</b>
Основные отличия Qt4 от Qt3 .....	686
Классы графического интерфейса .....	687
Контейнерные классы .....	688
Классы программирования сети .....	689
Классы для программирования баз данных .....	689
Qt Designer .....	689

Начало перевода на Qt4.....	689
Модуль совместимости Qt3Support.....	690
Завершение перевода на Qt4.....	691
Резюме.....	692
<b>ЧАСТЬ VII. ЯЗЫК СЦЕНАРИЕВ QT SCRIPT.....</b>	<b>693</b>
<b>Глава 50. Основы поддержки сценариев.....</b>	<b>695</b>
Принцип взаимодействия с языком сценариев.....	696
Первый шаг использования сценария.....	699
Привет, сценарий.....	700
Резюме.....	702
<b>Глава 51. Синтаксис языка сценариев.....</b>	<b>703</b>
Зарезервированные ключевые слова.....	703
Комментарии.....	704
Переменные.....	704
Предопределенные типы данных.....	705
Целый тип.....	705
Вещественный тип.....	705
Строковый тип.....	706
Логический тип.....	706
Преобразование типов.....	706
Константы.....	708
Операции.....	708
Операторы присваивания.....	708
Арифметические операции.....	708
Поразрядные операции.....	709
Операции сравнения.....	710
Приоритет выполнения операций.....	711
Управляющие структуры.....	711
Условные операторы.....	711
Оператор <i>if ... else</i> .....	713
Оператор <i>switch</i> .....	713
Оператор условного выражения.....	714
Циклы.....	715
Операторы <i>break</i> и <i>continue</i> .....	715
Цикл <i>for</i> .....	715
Цикл <i>while</i> .....	715
Цикл <i>do...while</i> .....	716
Оператор <i>with</i> .....	716
Исключительные ситуации.....	716
Оператор <i>try...catch</i> .....	716
Оператор <i>throw</i> .....	717
Функции.....	717
Встроенные функции.....	719
Объектная ориентация.....	719
Статические классы.....	722
Наследование.....	722

Перегрузка методов .....	724
Сказание о "джейсоне" .....	725
Резюме .....	726
<b>Глава 52. Встроенные объекты Qt Script .....</b>	<b>727</b>
Объект <i>Global</i> .....	727
Объект <i>Number</i> .....	727
Объект <i>Boolean</i> .....	727
Объект <i>String</i> .....	728
Преобразование строки к нижнему и верхнему регистрам .....	728
Замена .....	728
Получение символов .....	728
Получение подстроки .....	728
Объект <i>RegExp</i> .....	728
Проверка строки .....	729
Поиск совпадений .....	729
Объект <i>Array</i> .....	729
Дополнение массива элементами .....	730
Адресация элементов .....	730
Изменение порядка элементов массива .....	730
Преобразование массива в строку .....	731
Объединение массивов .....	731
Упорядочивание элементов .....	731
Многомерные массивы .....	731
Объект <i>Date</i> .....	732
Объект <i>Math</i> .....	733
Модуль числа .....	733
Округление .....	734
Определение максимума и минимума .....	734
Возведение в степень .....	734
Вычисление квадратного корня .....	734
Генератор случайных чисел .....	735
Тригонометрические методы .....	735
Вычисление натурального логарифма .....	735
Объект <i>Function</i> .....	736
Резюме .....	736
<b>Глава 53. Классы поддержки Qt Script и практические примеры.....</b>	<b>737</b>
Класс <i>QScriptValue</i> .....	737
Класс <i>QScriptContext</i> .....	737
Класс <i>QScriptEngine</i> .....	738
Практические примеры .....	740
"Черепашья" графика .....	741
Сигналы, слоты и функции .....	748
Отладчик Qt Script .....	751
Резюме .....	754
<b>ЧАСТЬ VIII. ТЕХНОЛОГИЯ QT QUICK.....</b>	<b>755</b>
<b>Глава 54. Знакомство с Qt Quick .....</b>	<b>757</b>
А зачем? .....	757



Введение в QML.....	759
Быстрый старт .....	759
Резюме .....	763
<b>Глава 55. Элементы.....</b>	<b>765</b>
Визуальные элементы.....	765
Свойства элементов .....	767
Собственные свойства .....	769
Создание собственных элементов .....	771
Использование JavaScript в QML .....	773
Резюме .....	774
<b>Глава 56. Управление размещением элементов .....</b>	<b>775</b>
Фиксаторы .....	775
Традиционные размещения.....	782
Резюме .....	784
<b>Глава 57. Элементы графики .....</b>	<b>785</b>
Цвета .....	785
Растровые изображения .....	786
Элемент <i>Image</i> .....	786
Элемент <i>BorderImage</i> .....	789
Градиенты.....	790
Шрифты .....	791
Резюме .....	792
<b>Глава 58. Пользовательский ввод .....</b>	<b>793</b>
Область мыши.....	793
Сигналы .....	796
Ввод с клавиатуры .....	800
Фокус.....	801
"Сырой" ввод.....	803
Резюме .....	804
<b>Глава 59. Анимация.....</b>	<b>805</b>
Анимация при изменении свойств.....	805
Анимация для изменения числовых значений.....	807
Анимация с изменением цвета.....	808
Анимация с поворотом .....	809
Анимации поведения .....	810
Параллельные и последовательные анимации .....	812
Состояния и переходы .....	815
Состояния .....	815
Переходы .....	818
Резюме .....	820
<b>Глава 60. Модель/Представление .....</b>	<b>822</b>
Модели.....	822
Модель списка.....	822
XML-модель .....	823

---

Представления данных моделей .....	825
Элемент <i>Flickable</i> .....	825
Элемент <i>ListView</i> .....	826
Элемент <i>GridView</i> .....	828
Элемент <i>PathView</i> .....	830
Резюме .....	833
<b>Глава 61. Qt Quick и C++ .....</b>	<b>834</b>
Использование языка QML в C++ .....	834
Использование компонентов языка C++ в QML .....	835
Резюме .....	838
<b>Эпилог .....</b>	<b>839</b>
<b>ПРИЛОЖЕНИЯ .....</b>	<b>841</b>
<b>Приложение 1. Таблицы семибитной кодировки ASCII.....</b>	<b>843</b>
<b>Приложение 2. Таблица простых чисел .....</b>	<b>846</b>
<b>Приложение 3. Глоссарий .....</b>	<b>849</b>
<b>Приложение 4. Описание архива с примерами.....</b>	<b>853</b>
<b>Предметный указатель .....</b>	<b>863</b>



# ГЛАВА 1

## Обзор иерархии классов Qt

Если вы хотите знать территорию — нужно сначала изучить карту.

*Тони Бьюзен*

### Первая программа на Qt

Как заведено, в самом начале знакомства нужно поздороваться, и, чтобы никого не оставить без внимания, лучше всего обратиться сразу ко всему миру. Давайте для этого напишем короткую программу "Hello World" ("Здравствуй, Мир"), результат выполнения которой показан на рис. 1.1.

Написание подобного рода программ стало уже традицией при знакомстве с новым языком или библиотекой. Хотя подобный пример не в состоянии продемонстрировать весь потенциал и возможности самой библиотеки, он дает представление о базовых понятиях. Данный пример позволяет оценить объем и сложность процесса реализации программ, использующих эту библиотеку. Кроме того, при помощи примера можно убедиться, что все необходимое для компиляции и компоновки установлено правильно.

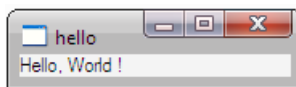


Рис. 1.1. Окно программы "Hello World"

#### Листинг 1.1. Программа "Hello World". Файл hello.cpp

```
#include <QtGui>
int main(int argc, char** argv)
{
    QApplication app(argc, argv);
    QLabel lbl("Hello, World !");
    lbl.show();
    return app.exec();
}
```

В первой строке листинга 1.1 подключается заголовочный файл `QtGui`, который представляет собой файл модуля, включающий в себя заголовочные файлы для используемых в нашей программе классов: `QApplication` и `QLabel`. Конечно, мы могли бы обойтись и без модуля

QtGui, а непосредственно подключить заголовочные файлы для поддержки классов QApplication и QLabel, но при большем количестве классов разных модулей, задействованных в программе, читаемость самой программы заметно бы ухудшилась. Кроме того, подключение модулей дает возможность ускорить компиляцию самой программы за счет предварительно откомпилированных заголовочных файлов (Precompiled Headers) в том случае, если ваш компилятор позволяет это делать.

Теперь давайте разберем наш пример. Сначала создается объект класса QApplication, который осуществляет контроль и управление приложением. Для его создания в конструктор этого класса необходимо передать два аргумента. Первый аргумент представляет собой информацию о количестве аргументов в командной строке, с которой происходит обращение к программе, а второй — это указатель на массив символьных строк, содержащих аргументы, по одному в строке. Любая использующая Qt-программа с графическим интерфейсом должна создавать только один объект этого класса, и он должен быть создан до использования операций, связанных с пользовательским интерфейсом.

Затем создается объект класса QLabel. После создания элементы управления Qt по умолчанию невидимы, и для их отображения необходимо вызвать метод show(). Объект класса QLabel является *основным управляющим элементом приложения*, что позволяет завершить работу приложения при закрытии окна элемента. Если вдруг окажется, что в созданном приложении имеется сразу несколько независимых друг от друга элементов управления, то при закрытии окна последнего такого элемента управления завершится и само приложение. Это правильно, иначе приложение осталось бы в памяти компьютера и использовало бы его ресурсы.

Наконец, в последней строке программы приложение запускается вызовом QApplication::exec(). С его запуском приводится в действие цикл обработки событий, определенный в классе QCoreApplication, являющимся базовым для QApplication. Этот цикл передает получаемые от системы события на обработку соответствующим объектам. Он продолжается до тех пор, пока либо не будет вызван статический метод QCoreApplication::exit(), либо не закроется окно последнего элемента управления. По завершению работы приложения метод QApplication::exec() возвращает значение целого типа, содержащее код, информирующий о его завершении.

## Модули Qt

В начале изучения классов новой библиотеки создается ощущение перенасыщения из-за большого объема информации. Иерархия классов Qt имеет четкую внутреннюю структуру, которую важно понять, чтобы уметь хорошо и интуитивно ориентироваться в этой библиотеке.

Библиотека Qt — это множество классов (более 500), которые охватывают большую часть функциональных возможностей операционных систем, предоставляя разработчику мощные механизмы, расширяющие и, вместе с тем, упрощающие разработку приложений. При этом не нарушается идеология операционной системы. Qt не является единым целым, она разбита на модули (табл. 1.1).

Любая Qt-программа так или иначе должна использовать хотя бы один из модулей, в большинстве случаев это QtCore и QtGui, поэтому эти два модуля определены в программе создания make-файлов (см. главу 3) по умолчанию. Для использования других модулей в своих проектах необходимо перечислить их в проектном файле (см. главу 3). Например, чтобы добавить модули, нужно написать

```
QT += opengl network sql
```

Таблица 1.1. Модули Qt

Библиотека	Обозначение в проектном файле	Назначение
QtCore	core	Основополагающий модуль, состоящий из классов, не связанных с графическим интерфейсом
QtGui	gui	Модуль для программирования графического интерфейса
QtNetwork	network	Модуль для программирования сети
QtOpenGL	opengl	Модуль для программирования графики OpenGL
QtSql	sql	Модуль для программирования баз данных
QtSvg	svg	Модуль для работы с SVG (Scalable Vector Graphics, масштабируемая векторная графика)
QtXml	xml	Модуль поддержки XML, классы, относящиеся к SAX и DOM
Qt3Support	qt3support	Модуль, содержащий классы для совместимости с предыдущей библиотекой Qt
QtScript	script	Модуль поддержки языка сценариев
Phonon	phonon	Модуль мультимедиа
QtWebKit	webkit	Модуль для создания Web-приложений
QtScriptTools	scripttools	Модуль дополнительных возможностей поддержки языка сценария. На настоящий момент предоставляет отладчик
QtTest	test	Модуль, содержащий классы для тестирования кода

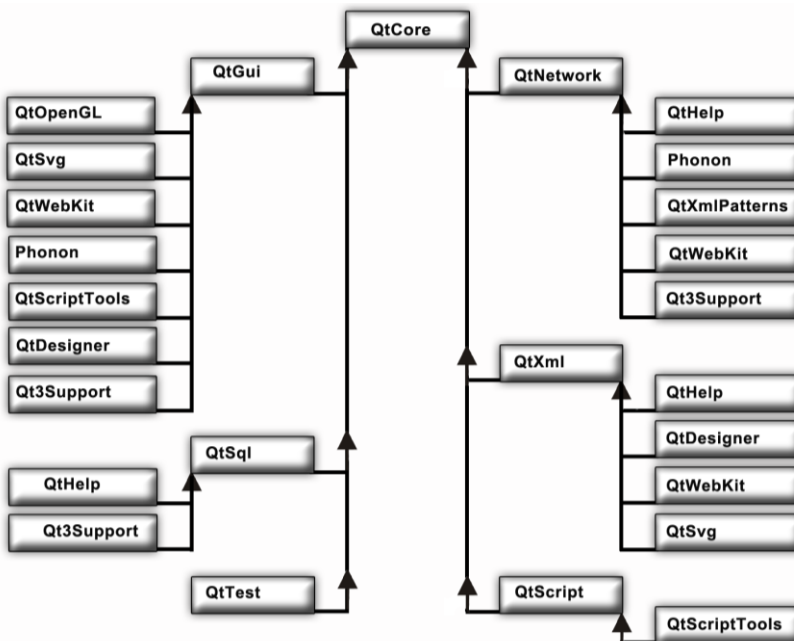


Рис. 1.2. Диаграмма некоторых модульных зависимостей

А чтобы исключить модуль из проекта:

```
QT -= gui
```

Наиболее значимый из перечисленных в табл. 1.1 модулей — это QtCore, т. к. он является базовым для всех остальных модулей (рис. 1.2). Далее идут модули, которые непосредственно зависят от QtCore, это — QtNetwork, QtGui, QSql и QtXml. И, наконец, модули, зависящие от только что упомянутых модулей — Qt3Support, QtOpenGL и QtSvg.

Для каждого модуля Qt предоставляет отдельный заголовочный файл, содержащий заголовочные файлы всех классов этого модуля. Название этого заголовочного файла соответствует названию самого модуля. Например, для включения QtGui модуля нужно добавить в программу строку

```
#include <QtGui>
```

## Пространство имен Qt

Пространство имен Qt содержит ряд типов перечислений и констант, которые часто применяются при программировании. Если вам необходимо получить доступ к какой-либо константе этого пространства имен, то вы должны указать префикс Qt (например, не `red`, а `Qt::red`). Если вы все-таки хотите опускать префикс Qt, то необходимо в начале файла с исходным кодом добавить следующую директиву:

```
using namespace Qt;
```

## Модуль QtCore

Как уже было сказано ранее, базовым является модуль QtCore. Этот модуль является базовым для приложений и не содержит классов, относящихся к интерфейсу пользователя. Если вы собираетесь реализовать консольное приложение, то, вполне возможно, вы можете ограничиться одним этим модулем. В модуль QtCore входят более 200 классов, вот некоторые из них:

- ◆ контейнерные классы `QList`, `QVector`, `QMap` (см. главу 4);
- ◆ классы для ввода и вывода `QIODevice`, `QTextStream`, `QFile` (см. главу 37);
- ◆ классы процесса `QProcess` и для программирования многопоточности `QThread`, `QWaitCondition`, `QMutex` (см. главу 39);
- ◆ классы для работы с таймером `QBasicTimer` и `QTimer` (см. главу 38);
- ◆ классы для работы с датой и временем `QDate` и `QTime` (см. главу 38);
- ◆ класс `QObject`, являющийся *краеугольным камнем* объектной модели Qt (см. главу 2);
- ◆ базовый класс событий `QEvent` (см. главу 14);
- ◆ класс для сохранения настроек приложения `QSettings` (см. главу 29);
- ◆ класс приложения `QCoreApplication`, из объекта которого, если требуется, можно запустить цикл событий.

Давайте немного остановимся на классе `QCoreApplication`. Объект класса приложения `QCoreApplication` можно образно сравнить с сосудом, содержащим объекты, подсоединенные к контексту операционной системы. Срок жизни объекта класса `QCoreApplication` соответствует продолжительности работы всего приложения, и он остается доступным в любой

момент работы программы. Объект класса `QCoreApplication` должен создаваться в приложении только один раз. К задачам этого объекта можно отнести:

- ◆ управление событиями между приложением и операционной системой;
- ◆ передачу и предоставление аргументов командной строки.

Кроме того, `QCoreApplication` можно унаследовать, для того чтобы перезаписать некоторые методы, а также задействовать сам объект для дополнительных глобальных данных, используемых внутри приложения. Такой подход может избавить вас от нежелательного использования шаблона проектирования Singleton.

## Модуль QtGui

Этот модуль содержит в себе классы, необходимые для программирования графического интерфейса пользователя. В этот модуль входят около 300 классов. Вот некоторые из них:

- ◆ класс `QWidget` — это базовый класс для всех элементов управления библиотеки Qt. По своему внешнему виду это не что иное, как заполненный четырехугольник, но за этой внешней простотой скрывается большой потенциал непростых функциональных возможностей. Этот класс насчитывает 254 метода и 53 свойства. В *главе 5* этому элементу уделено особое внимание;
- ◆ классы для автоматического размещения элементов `QVBoxLayout`, `QHBoxLayout` (см. *главу 6*);
- ◆ классы элементов отображения `QLabel`, `QLCDNumber` (см. *главу 7*);
- ◆ классы кнопок `QPushButton`, `QCheckBox`, `QRadioButton` (см. *главу 8*);
- ◆ классы элементов установок `QSlider`, `QScrollBar` (см. *главу 9*);
- ◆ классы элементов ввода `QLineEdit`, `QSpinBox` (см. *главу 10*);
- ◆ классы элементов выбора `QComboBox`, `QToolBox` (см. *главу 11*);
- ◆ классы меню `QMainWindow` и `QMenu` (см. *главы 32* и *35*);
- ◆ классы окон сообщений и диалоговых окон `QMessageBox`, `QDialog` (см. *главу 33*);
- ◆ классы для рисования `QPainter`, `QBrush`, `QPen`, `QColor` (см. *главу 18*);
- ◆ классы для растровых изображений `QImage`, `QPixmap` (см. *главу 19*);
- ◆ классы стилей `QMotifStyle`, `QWindowsStyle` и др. (см. *главу 26*). Как отдельному элементу, так и всему приложению может быть присвоен определенный стиль, изменяющий их внешний облик;
- ◆ класс приложения `QApplication`, который предоставляет цикл событий.

Давайте рассмотрим немного поподробнее последний класс, класс `QApplication`, с которым мы встречались в самом первом примере. Все, что было сказано выше о классе `QCoreApplication`, относится также и к этому классу, т. к. он является прямым его наследником. Объект класса `QApplication` представляет собой центральный контрольный пункт всех Qt-приложений, имеющих пользовательский интерфейс. Данный объект используется для получения событий клавиатуры, мыши, таймера и других событий, на которые приложение должно реагировать соответствующим образом. Например, окно даже самого простого приложения может быть изменено по величине или быть перекрыто окном другого приложения, и на все подобные события необходима правильная реакция.

В назначение `QApplication` входит:

- ◆ установка стиля приложения. Таким образом можно устанавливать стиль `Motif`, `Windows`, а также многие другие *виды и поведения* (Look & Feel) приложения, включая и свои собственные (см. главу 26);
- ◆ получение указателя на объект *рабочего стола* (desktop);
- ◆ получение доступа к буферу обмена (см. главу 30);
- ◆ управление глобальными манипуляциями с мышью (например, установка интервала двойного щелчка кнопкой мыши) и регистрация движения мыши в пределах и за пределами окна приложения;
- ◆ выдача предупреждающего звукового сигнала (см. главу 27);
- ◆ обеспечение правильного завершения работающего приложения при завершении работы операционной системы (см. главу 29);
- ◆ инициализация необходимых настроек приложения, например палитры для расцветки элементов управления (см. главу 13).

Бывает так, что приложение может быть неактивным, а есть необходимость обратить на себя внимание пользователя. Для этой цели класс `QApplication` предоставляет статический метод `alert()`. Его вызов приведет к подскакиванию значка приложения на док-панели в ОС Mac OS X, а в ОС Windows на панели задач произойдет его пульсация, как это показано на рис. 1.3 и 1.4.



Рис. 1.3. Подскакивание значка приложения на док-панели в ОС Mac OS X

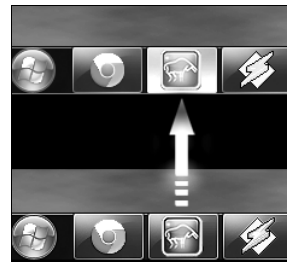


Рис. 1.4. Пульсация значка приложения на панели задач в Windows 7

## Модуль QtNetwork

Сетевой модуль предоставляет инструментарий для программирования TCP- и UDP-сокетов (классы `QTcpSocket` и `QUdpSocket`), а также для реализации программ-клиентов, использующих HTTP- и FTP-протоколы (классы `QHttp` и `QFtp`). Этот модуль описывается в главе 40.

## Модуль QtXml

Этот модуль предназначен для работы с XML посредством SAX2- и DOM-интерфейсов, которые определяют классы Qt (см. главу 41).



## Модуль QSql

Этот модуль предназначен для работы с базами данных. В него входят классы, предоставляющие возможность для манипулирования значениями баз данных (см. главу 42).

## Модуль QtOpenGL

Модуль QtOpenGL делает возможным использование OpenGL в Qt-программах для двух- и трехмерной графики. Основным классом этого модуля является `QOpenGLWidget`, который унаследован от `QWidget` (см. главу 23).

## Модуль QtWebKit

Этот модуль позволяет очень просто интегрировать в приложение содержание из Web. Возможно также расширять элементы Web своими собственными виджетами. Более подробно с этим модулем можно ознакомиться в главе 47.

## Модуль QtSvg

Модуль поддержки графического векторного формата SVG, базирующегося на XML. Этот формат предоставляет возможность не только для вывода одного кадра векторного изображения, но может быть использован и для векторной анимации (см. главу 22).

## Модуль Qt3Support

Если сравнивать третью версию Qt с четвертой, то можно заметить, что библиотека подверглась очень большим изменениям. Некоторые классы были удалены из новой версии библиотеки, а другие претерпели такие изменения, которые сделали их несовместимыми со старым программным кодом. Основное назначение этого модуля состоит в облегчении адаптации старых Qt-программ к Qt4. Разумеется, этот модуль не нужно использовать в ваших новых программах, написанных на базе Qt4. Некоторые рекомендации по адаптации старых Qt-программ к Qt4 можно найти в главе 49.

## Резюме

Библиотека Qt не является монолитной библиотекой, она разбита на отдельные модули: QtCore, QtGui, QtNetwork, QtOpenGL, QSql, QtXml и QtSvg. Каждый модуль имеет свое назначение, например программирование интерфейса пользователя, графики, баз данных и др. Классы модулей предоставляют разработчику механизмы, расширяющие возможности программистов и, вместе с тем, упрощающие создание приложений. Вершиной модульной иерархии является модуль QtCore, который позволяет реализовывать приложения без графического интерфейса пользователя (консольные приложения). Объект класса `QCoreApplication` должен быть создан в приложении только один раз.

Для реализации приложений с графическим интерфейсом пользователя необходим модуль QtGui. Класс `QApplication` является стержнем любого Qt-приложения с графическим интерфейсом. Объект этого класса также не должен создаваться в приложении больше одного раза. Данный объект используется для получения событий клавиатуры, мыши и др.



## ГЛАВА 2

# Философия объектной модели

Те, кого первое знакомство с квантовой теорией не повергло в шок, скорее всего, вовсе ее не поняли.

*Макс Борн*

Объектная модель Qt подразумевает, что все построено на объектах. Фактически, класс `QObject` — основной, базовый класс. Подавляющее большинство классов Qt являются его наследниками. Классы, имеющие сигналы и слоты, должны быть унаследованы от этого класса.

### **ПРИМЕЧАНИЕ**

При множественном наследовании важно помнить, что при определении класса имя класса `QObject` (или унаследованного от него) должно стоять первым, чтобы МОС (Meta Object Compiler, метаобъектный компилятор) мог правильно распознать его. Другой порядок приведет к ошибке при компиляции. В листинге 2.1 приведен правильный порядок для множественного наследования.

### **Листинг 2.1. Порядок наследования**

```
class MyClass : public QObject, public AnotherClass {  
    ...  
};
```

### **ПРИМЕЧАНИЕ**

При множественном наследовании также важно учитывать, что от класса `QObject` должен быть унаследован только один из базовых классов. Другими словами, нельзя производить наследование сразу от нескольких классов, наследующих класс `QObject`.

Класс `QObject` содержит в себе поддержку:

- ◆ сигналов и слотов (signal/slot);
- ◆ таймера;
- ◆ механизма объединения объектов в иерархии;
- ◆ событий и механизма их фильтрации;
- ◆ организации объектных иерархий;
- ◆ метаобъектной информации;

- ◆ приведения типов;
- ◆ свойств.

Сигналы и слоты — это средства, позволяющие эффективно производить обмен информацией о событиях, вырабатываемых объектами. О них мы подробно поговорим позже в этой главе.

Поддержка таймера дает возможность каждому из классов, унаследованных от класса `QObject`, не создавать дополнительно объект таймера. Тем самым экономится время на разработку. Подробнее о таймерах говорится в *главе 38*.

Механизм объединения объектов в иерархические структуры позволяет резко сократить временные затраты при разработке приложений, не заботясь об освобождении памяти создаваемых объектов, т. к. объекты-предки сами отвечают за уничтожение своих потомков.

Механизм фильтрации событий позволяет осуществить их перехват. Фильтр событий может быть установлен в любом классе, унаследованном от класса `QObject`, благодаря чему можно изменять реакцию объектов на происходящие события без изменения исходного кода класса (см. *главу 15*).

Метаобъектная информация включает в себя информацию о наследовании классов, что позволяет определять, являются ли классы непосредственными наследниками, а также узнать имя класса.

Для приведения типов Qt предоставляет шаблонную функцию `qobject_cast<T>()`, базирующуюся на метаинформации, создаваемой метаобъектным компилятором МОС (см. *главу 3*), для классов, унаследованных от `QObject`.

*Свойства* — это поля, для которых обязательно должны существовать методы чтения. С их помощью можно получать доступ к атрибутам объектов извне, например из языка сценариев Qt Script (см. *часть VII*). Свойства также широко используются в визуальной среде разработки пользовательского интерфейса Qt Designer (см. *главу 45*). Этот механизм реализован в Qt при помощи директив препроцессора. Задается свойство при помощи макроса `Q_PROPERTY`. Определение свойства в общем виде выглядит следующим образом:

```
Q_PROPERTY(type name
           READ getFunction
           [WRITE setFunction]
           [RESET resetFunction]
           [DESIGNABLE bool]
           [SCRIPTABLE bool]
           [STORED bool]
           )
```

Первыми задаются тип и имя свойства, вторым — имя метода чтения (`READ`). Определение остальных параметров не является обязательным. Третий параметр задает имя метода записи (`WRITE`), четвертый — имя метода сброса значения (`RESET`), пятый (`DESIGNABLE`) является логическим (булевым) значением, говорящим о том, должно ли свойство появляться в инспекторе свойств Qt Designer. Шестой параметр (`SCRIPTABLE`) — также логическое значение, которое управляет тем, будет ли свойство доступно для языка сценариев Qt Script. Последний, седьмой параметр (`STORED`) управляет сериализацией, т. е. тем, будет ли свойство запонинаться во время сохранения объекта.

Итак, теперь, когда вы познакомились с понятием "свойство", давайте в качестве простого примера определим в классе свойство для управления режимом только чтения (листинг 2.2).

**Листинг 2.2. Определение свойства для управления режимом только чтения**

```

class MyClass : public QObject {
    Q_OBJECT
    Q_PROPERTY(bool readOnly READ isReadOnly WRITE setReadOnly)

private:
    bool m_bReadOnly;

public:
    MyClass(QObject* pObj = 0) : QObject(pObj)
        , m_bReadOnly(false)
    {
    }

public:
    void setReadOnly(bool bReadOnly)
    {
        m_bReadOnly = bReadOnly;
    }

    bool isReadOnly() const
    {
        return m_bReadOnly;
    }
}

```

Класс `MyClass`, показанный в листинге 2.2, наследуется от класса `QObject`. Мы определяем атрибут `m_bReadOnly`, в котором будут запоминаться значения состояния. Этот атрибут инициализируется в конструкторе значением `false`. Для получения и изменения значения атрибута в классе `MyClass` определены методы `isReadOnly()` и `setReadOnly()`. Эти методы регистрируются в макросе `Q_PROPERTY`. Метод `isReadOnly()` служит для получения значения, поэтому указывается в секции `READ`, а метод `setReadOnly()` — для изменения значения, поэтому пишется в секции `WRITE`.

Из программы мы можем изменить значение нашего свойства следующим образом:

```
pObj->setProperty("readOnly", true);
```

А так можно получить текущее значение:

```
bool bReadOnly = pObj->property("readOnly").toBool();
```

## Механизм сигналов и слотов

Элементы графического интерфейса определенным образом реагируют на действия пользователя и посылают сообщения. Существует несколько вариантов такого решения.

Старая концепция *функций обратного вызова* (callback functions), лежащая в основе X Window System, основана на использовании обычных функций, которые должны вызываться в результате действий пользователя. Применение такой концепции значительно

усложняет исходный код программы, делая его менее понятным. Кроме того, отсутствует возможность производить проверку типов возвращаемых значений, потому что во всех случаях возвращается указатель на пустой тип `void`. Например, для того чтобы сопоставить код с кнопкой, необходимо передать в функцию указатель на кнопку. Если пользователь нажимает на кнопку, функция будет вызвана. Сами библиотеки не проверяют, были ли аргументы, переданные в функцию, требуемого типа, а это часто является причиной сбоев. Другой недостаток функций обратного вызова заключается в том, что элементы графического интерфейса пользователя тесно связаны с функциональными частями программы и это, в свою очередь, заметно усложняет разработку классов независимо друг от друга. Одним из ярких представителей этой концепции является библиотека `Motif`.

Важно помнить, что `Motif` и `Windows API` предназначены для процедурного программирования, и с реализацией объектно-ориентированных проектов, наверняка, появятся трудности.

Для программирования в ОС `Windows` существуют специальные библиотеки классов языка `C++`, облегчающие программирование для этой операционной системы. Самой популярной библиотекой является `Microsoft Foundation Classes (MFC)`. Ее можно, с большой натяжкой, назвать объектно-ориентированной, т. к. она создавалась людьми, не подозревающими о существовании самых элементарных принципов объектно-ориентированного подхода. Одна из самых фундаментальных заповедей объектно-ориентированного подхода — это *инкапсуляция*, которая запрещает оставлять атрибуты классов незащищенными (ведь тогда объекты могут читать и изменять данные без ведома объекта-владельца), но, несмотря на это, во многих `MFC`-классах это требование не соблюдено. Сама библиотека `MFC` является надстройкой, предоставляющей доступ к функциям `Windows`, реализованным на языке `C`, что заставляет разработчиков время от времени использовать устаревшие структуры, не вписывающиеся в рамки концепции объектно-ориентированного подхода. Интересно также отметить, что сама `Microsoft` для реализации широко известной программы `Microsoft Word` не использует `MFC` вообще.

При использовании `MFC` для обеспечения связей сообщения и методов обработки используются специальные макросы — так называемые карты сообщений (листинг 2.3). Они очень сильно загромождают исходный код программы, заметно снижая ее читаемость.

### Листинг 2.3. Отрывок программы, реализованной с помощью `MFC`

```
class CPhotoStylerApp : public CWinApp {
public:
    CPhotoStylerApp();
public:
    virtual BOOL InitInstance();

    afx_msg void OnAppAbout();
    afx_msg void OnFileNew();

    DECLARE_MESSAGE_MAP()
};
BEGIN_MESSAGE_MAP(CPhotoStylerApp, CWinApp)
    ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
    ON_COMMAND(ID_FILE_NEW, OnFileNew)
    ON_COMMAND(ID_FILE_NEW, CWinApp::OnFileNew)
```

```

ON_COMMAND(ID_FILE_OPEN, CWinApp::OnFileOpen)
ON_COMMAND(ID_FILE_PRINT_SETUP, CWinApp::OnFilePrintSetup)
END_MESSAGE_MAP()

```

Конструкции, подобные показанной в листинге 2.3, очень неудобны для человеческого восприятия и приводят в замешательство при проведении анализа кода программы. Пусть многие рассказывают об удобстве использования средств для автоматического создания подобного кода, но созданы они были не от хорошей жизни. Так, непродуманность самой библиотеки вынуждает разработчика при внесении незначительных изменений модифицировать код самой программы сразу в нескольких местах. Например, для того чтобы добавить в диалоговое окно текстовое поле, необходимо провести целый ряд операций. Во-первых, нужно создать в классе диалога атрибут, предназначенный для хранения значений, вводимых в текстовом поле. Во-вторых, надо задать идентификатор ресурса текстового поля. В-третьих, поставить идентификатор ресурса и атрибут в методе `DoDataExchange()` в соответствие друг с другом при помощи метода `DDX_Text()`, после чего будет осуществляться обмен данными между текстовым полем и атрибутом. В-четвертых, этим обменом необходимо управлять, передавая в методе `UpdateData()` значения булевого типа `true` или `false`. И лишь с помощью средств автоматического создания кода можно частично избавиться от этой проблемы, заставив выполнить эти изменения за вас и получив взамен другие недостатки, например дополнительное засорение кода программы ненужной информацией и возможное несовпадение созданного кода с утвержденными для проекта требованиями для форматирования и нотации (если не используется венгерская нотация). Я не противник обоснованного применения подобного рода средств, но, по моему мнению, они не должны создаваться как средство устранения изъянов плохого дизайна самой библиотеки.

В данной ситуации часть вины скрыта в самом языке C++. Дело в том, что C++ не создавался как средство для написания пользовательского интерфейса, и поэтому не предоставляет соответствующей поддержки, делающей программирование в этой области более удобным. Например, если бы работа по передаче событий реализовывалась средствами самого языка, тогда отпадала бы необходимость в использовании подобного рода макросов. До настоящего времени не удавалось сделать ничего подобного, именно поэтому библиотека Qt явилась "как гром среди ясного неба". В отличие от большинства других библиотек программирования, Qt расширяет язык C++ дополнительными ключевыми словами.

Проблема расширения языка C++ решена в Qt с помощью специального препроцессора МОС (Meta Object Compiler, метаобъектный компилятор). Он анализирует классы на наличие специального макроса `Q_OBJECT` в их определении и внедряет в отдельный файл всю необходимую дополнительную информацию. Это происходит автоматически, без непосредственного участия разработчика. Подобная операция автоматического создания кода не противоречит привычному процессу программирования на C++, ведь стандартный препроцессор перед компиляцией самой программы тоже создает промежуточный код, содержащий исполненные команды препроцессора. Подобным образом действует и МОС, записывая всю необходимую дополнительную информацию в отдельный файл, содержимое которого может быть проигнорировано. Макрос `Q_OBJECT` должен располагаться сразу на следующей строке после ключевого слова `class` с определением имени класса. Очень важно помнить, что после макроса не должно стоять точки с запятой. Внедрять макрос в определение класса имеет смысл в тех случаях, когда созданный класс использует механизм сигналов и слотов или если ему необходима информация о свойствах.

Механизм сигналов и слотов полностью замещает старую модель функций обратного вызова, он очень гибок и полностью объектно-ориентирован. Сигналы и слоты — это краеугольный концепт программирования с использованием Qt, позволяющий соединить вместе