

Qt 5.3

ПРОФЕССИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ НА C++



- Кроссплатформенная реализация приложений для Windows, Mac OS X и Linux
- Программирование графики, мультимедиа, веб-приложений, баз данных, сети, таймера, многопоточности, XML, QML и JavaScript
- 230 завершенных программ



Материалы
на www.bhv.ru

Наиболее
полное
руководство

В ПОДЛИННИКЕ®

УДК 004.438С++
ББК 32.973.26-018.1
Ш68

Шлее М.

Ш68 Qt 5.3. Профессиональное программирование на С++. — СПб.:
БХВ-Петербург, 2015. — 928 с.: ил. — (В подлиннике)

ISBN 978-5-9775-3346-1

Книга посвящена разработке приложений для Windows, Mac OS X и Linux с использованием библиотеки Qt версии 5.3. Подробно рассмотрены возможности, предоставляемые этой библиотекой, и описаны особенности, выгодно отличающие ее от других библиотек. Описана интегрированная среда разработки Qt Creator и работа с технологией Qt Quick. Книга содержит исчерпывающую информацию о классах Qt 5, и так же даны практические рекомендации их применения, проиллюстрированные на большом количестве подробно прокомментированных примеров. Проекты примеров из книги размещены на сайте издательства.

Для программистов

УДК 004.438С++
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	<i>Екатерина Кондукова</i>
Зам. главного редактора	<i>Евгений Рыбаков</i>
Зав. редакцией	<i>Екатерина Капальгина</i>
Редактор	<i>Григорий Добин</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 05.03.15.
Формат 70×100^{1/16}. Печать офсетная. Усл. печ. л. 74,82.
Тираж 1500 экз. Заказ №
"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20.
Первая Академическая типография "Наука"
199034, Санкт-Петербург, 9 линия, 12/28

ISBN 978-5-9775-3346-1

© Шлее М., 2015
© Оформление, издательство "БХВ-Петербург", 2015

Оглавление

Предисловие Маттиаса Эттриха	20
Благодарности.....	22
Предисловие	23
Структура книги.....	23
Введение	32
ЧАСТЬ I. ОСНОВЫ QT	43
Глава 1. Обзор иерархии классов Qt	45
Первая программа на Qt.....	45
Модули Qt	46
Пространство имен Qt	48
Модуль <i>QtCore</i>	48
Модуль <i>QtGui</i>	49
Модуль <i>QtWidgets</i>	49
Модули <i>QtQuick</i> и <i>QtQML</i>	50
Модуль <i>QtNetwork</i>	51
Модули <i>QtXml</i> и <i>QtXmlPatterns</i>	51
Модуль <i>QtSql</i>	51
Модуль <i>QtOpenGL</i>	51
Модули <i>QtWebKit</i> и <i>QtWebKitWidgets</i>	51
Модули <i>QtMultimedia</i> и <i>QtMultimediaWidgets</i>	51
Модули <i>QtScript</i> и <i>QtScriptTools</i>	51
Модуль <i>QtSvg</i>	52
Резюме	52
Глава 2. Философия объектной модели	53
Механизм сигналов и слотов	55
Сигналы	58
Слоты	60
Соединение объектов	61

Разъединение объектов	66
Переопределение сигналов	67
Организация объектных иерархий	68
Метаобъектная информация	70
Резюме	71
Глава 3. Работа с Qt	72
Интегрированная среда разработки	72
Qt Assistant	72
Работа с qmake	72
Рекомендации для проекта с Qt	76
Метаобъектный компилятор MOC	77
Компилятор ресурсов RCC	78
Структура Qt-проекта	79
Методы отладки	79
Отладчик GDB (GNU Debugger)	80
Прочие методы отладки	83
Глобальные определения Qt	86
Информация о библиотеке Qt	87
Резюме	89
Глава 4. Библиотека контейнеров	90
Контейнерные классы	91
Итераторы	92
Итераторы в стиле Java	93
Итераторы в стиле STL	94
Ключевое слово <i>foreach</i>	96
Последовательные контейнеры	96
Вектор <i>QVector<T></i>	97
Массив байтов <i>QByteArray</i>	98
Массив битов <i>QBitArray</i>	99
Списки <i>QList<T></i> и <i>QLinkedList<T></i>	99
Стек <i>QStack<T></i>	101
Очередь <i>QQueue<T></i>	101
Ассоциативные контейнеры	102
Словари <i>QMap<K,T></i> и <i>QMultiMap<K,T></i>	103
Хэши <i>QHash<K,T></i> и <i>QMultiHash<K,T></i>	104
Множество <i>QSet<T></i>	105
Алгоритмы	107
Сортировка	108
Поиск	109
Сравнение	109
Заполнение значениями	109
Строки	110
Регулярные выражения	111
Произвольный тип <i>QVariant</i>	113
Модель общего использования данных	114
Резюме	115

ЧАСТЬ II. ЭЛЕМЕНТЫ УПРАВЛЕНИЯ	117
Глава 5. С чего начинаются элементы управления?	119
Класс <i>QWidget</i>	119
Размеры и координаты виджета	122
Механизм закулисного хранения	123
Установка фона виджета	123
Изменение указателя мыши	124
Стек виджетов	127
Рамки	127
Виджет видовой прокрутки	128
Резюме	130
Глава 6. Управление автоматическим размещением элементов	131
Менеджеры компоновки (layout managers)	131
Горизонтальное и вертикальное размещение	133
Класс <i>QBoxLayout</i>	133
Горизонтальное размещение <i>QHBoxLayout</i>	135
Вертикальное размещение <i>QVBoxLayout</i>	136
Вложенные размещения	137
Табличное размещение <i>QGridLayout</i>	138
Порядок следования табулятора	144
Разделители <i>QSplitter</i>	144
Резюме	145
Глава 7. Элементы отображения	146
Надписи	146
Индикатор процесса	150
Электронный индикатор	153
Резюме	155
Глава 8. Кнопки, флажки и переключатели	156
С чего начинаются кнопки? Класс <i>QAbstractButton</i>	156
Установка текста и изображения	156
Взаимодействие с пользователем	156
Опрос состояния	157
Кнопки	157
Флажки	160
Переключатели	161
Группировка кнопок	162
Резюме	165
Глава 9. Элементы настройки	166
Класс <i>QAbstractSlider</i>	166
Изменение положения	166
Установка диапазона	166
Установка шага	167
Установка и получение значений	167

Ползунок.....	167
Полоса прокрутки.....	169
Установщик.....	170
Резюме.....	172
Глава 10. Элементы ввода.....	173
Однорочное текстовое поле.....	173
Редактор текста.....	175
Запись в файл.....	178
Расцветка синтаксиса (syntax highlighting).....	178
С чего начинаются виджеты счетчиков?.....	184
Счетчик.....	185
Элемент ввода даты и времени.....	186
Проверка ввода.....	187
Резюме.....	188
Глава 11. Элементы выбора.....	189
Простой список.....	189
Вставка элементов.....	189
Выбор элементов пользователем.....	191
Изменение элементов пользователем.....	191
Режим пиктограмм.....	191
Сортировка элементов.....	192
Иерархические списки.....	193
Сортировка элементов.....	196
Таблицы.....	196
Выпадающий список.....	198
Вкладки.....	199
Виджет панели инструментов.....	200
Резюме.....	201
Глава 12. Интервью, или модель-представление.....	202
Концепция.....	203
Модель.....	203
Представление.....	205
Выделение элемента.....	206
Делегат.....	208
Индексы модели.....	210
Иерархические данные.....	210
Роли элементов.....	214
Создание собственных моделей данных.....	215
Промежуточная модель данных (Proxy model).....	222
Модель элементарно-ориентированных классов.....	224
Резюме.....	226
Глава 13. Цветовая палитра элементов управления.....	227
Резюме.....	230

ЧАСТЬ III. СОБЫТИЯ И ВЗАИМОДЕЙСТВИЕ С ПОЛЬЗОВАТЕЛЕМ 231**Глава 14. События 233**

Переопределение специализированных методов обработки событий.....	235
События клавиатуры.....	235
Класс <i>QKeyEvent</i>	235
Класс <i>QFocusEvent</i>	238
Событие обновления контекста рисования. Класс <i>QPaintEvent</i>	238
События мыши.....	239
Класс <i>QMouseEvent</i>	239
Класс <i>QWheelEvent</i>	243
Методы <i>enterEvent()</i> и <i>leaveEvent()</i>	243
Событие таймера. Класс <i>QTimerEvent</i>	243
События перетаскивания (drag & drop).....	244
Класс <i>QDragEnterEvent</i>	244
Класс <i>QDragLeaveEvent</i>	244
Класс <i>QDragMoveEvent</i>	244
Класс <i>QDropEvent</i>	244
Остальные классы событий.....	244
Класс <i>QChildEvent</i>	244
Класс <i>QCloseEvent</i>	244
Класс <i>QHideEvent</i>	245
Класс <i>QMoveEvent</i>	245
Класс <i>QShowEvent</i>	245
Класс <i>QResizeEvent</i>	245
Реализация собственных классов событий.....	246
Переопределение метода <i>event()</i>	247
Сохранение работоспособности приложения.....	250
Резюме.....	251

Глава 15. Фильтры событий 252

Реализация фильтров событий.....	252
Резюме.....	255

Глава 16. Искусственное создание событий..... 256

Резюме.....	259
-------------	-----

ЧАСТЬ IV. ГРАФИКА И ЗВУК 261**Глава 17. Введение в компьютерную графику 263**

Классы геометрии.....	263
Точка.....	263
Двумерный размер.....	264
Прямоугольник.....	266
Прямая линия.....	266
Многоугольник.....	267
Цвет.....	267
Класс <i>QColor</i>	267
Цветовая модель RGB.....	268

Цветовая модель HSV	269
Цветовая модель CMYK.....	270
Палитра.....	271
Предопределенные цвета	272
Резюме	273
Глава 18. Легенда о короле Артуре и контекст рисования.....	274
Класс <i>QPainter</i>	275
Перья и кисти	277
Перо	277
Кисть	278
Градиенты.....	279
Техника сглаживания (Anti-aliasing)	280
Рисование	281
Рисование точек	281
Рисование линий.....	282
Рисование сплошных прямоугольников	283
Рисование заполненных фигур	283
Запись команд рисования.....	286
Трансформация систем координат.....	286
Перемещение.....	287
Масштабирование.....	288
Поворот.....	288
Скос.....	288
Трансформационные матрицы	288
Графическая траектория (painter path).....	289
Отсечения	290
Режим совмещения (composition mode).....	291
Графические эффекты	294
Резюме	296
Глава 19. Растровые изображения.....	297
Форматы графических файлов	297
Формат BMP	297
Формат GIF.....	298
Формат PNG	298
Формат JPEG	298
Формат XPM	298
Контекстно-независимое представление	300
Класс <i>QImage</i>	300
Класс <i>QImage</i> как контекст рисования	307
Контекстно-зависимое представление	308
Класс <i>QPixmap</i>	308
Класс <i>QPixmapCache</i>	310
Класс <i>QBitmap</i>	310
Использование масок для <i>QPixmap</i>	310
Создание нестандартного окна виджета	311
Резюме	314

Глава 20. Работа со шрифтами	316
Отображение строки.....	318
Резюме	321
Глава 21. Графическое представление.....	322
Сцена.....	324
Представление.....	324
Элемент	325
События	327
Виджеты в графическом представлении	333
Резюме	335
Глава 22. Анимация	336
Класс <i>QMovie</i>	336
SVG-графика	338
Анимационный движок и машина состояний	339
Смягчающие линии.....	342
Машина состояний и переходы	346
Резюме	349
Глава 23. Работа с OpenGL.....	350
Основные положения OpenGL	350
Классы Qt для работы с OpenGL	352
Реализация OpenGL-программы	352
Разворачивание OpenGL-программ во весь экран	355
Графические примитивы OpenGL	356
Трехмерная графика	359
Резюме	363
Глава 24. Вывод на печать	364
Класс <i>QPrinter</i>	364
Резюме	369
Глава 25. Разработка собственных элементов управления	370
Примеры создания виджетов.....	370
Резюме	375
Глава 26. Элементы со стилем.....	376
Встроенные стили.....	378
Создание собственных стилей	382
Метод рисования простых элементов управления.....	383
Метод рисования элементов управления	383
Метод рисования составных элементов управления	384
Реализация стиля простого элемента управления	384
Использование <i>QStyle</i> для рисования виджетов	388
Использование каскадных стилей документа	388
Основные положения	389
Изменение подэлементов	390
Управление состояниями	391
Пример.....	392
Резюме	396

Глава 27. Мультимедиа.....	397
Звук	397
Воспроизведение WAV-файлов: класс <i>QSound</i>	398
Более продвинутые возможности воспроизведения звуковых файлов: класс <i>QMediaPlayer</i>	399
Видео и класс <i>QMediaPlayer</i>	406
Резюме	408
 ЧАСТЬ V. СОЗДАНИЕ ПРИЛОЖЕНИЙ.....	409
 Глава 28. Сохранение настроек приложения.....	411
Управление сеансом	418
Резюме	420
 Глава 29. Буфер обмена и перетаскивание.....	421
Буфер обмена	421
Перетаскивание.....	422
Реализация drag	424
Реализация drop.....	426
Создание собственных типов перетаскивания	428
Резюме	433
 Глава 30. Интернационализация приложения	435
Подготовка приложения к интернационализации	435
Утилита <i>lupdate</i>	437
Программа <i>Qt Linguist</i>	438
Утилита <i>lrelease</i> . Пример программы, использующей перевод.....	439
Смена перевода в процессе работы программы	441
Завершающие размышления.....	443
Резюме	444
 Глава 31. Создание меню	445
«Анатомия» меню	445
Отрывные меню	449
Контекстные меню	450
Резюме	451
 Глава 32. Диалоговые окна	452
Правила создания диалоговых окон.....	452
Класс <i>QDialog</i>	453
Модальные диалоговые окна	453
Немодальные диалоговые окна	454
Создание собственного диалогового окна	454
Стандартные диалоговые окна	458
Диалоговое окно выбора файлов.....	458
Диалоговое окно настройки принтера	460
Диалоговое окно выбора цвета.....	461
Диалоговое окно выбора шрифта.....	462
Диалоговое окно ввода.....	463

Диалоговое окно процесса	464
Диалоговые окна мастера.....	465
Диалоговые окна сообщений.....	466
Окно информационного сообщения.....	468
Окно предупреждающего сообщения	469
Окно критического сообщения.....	469
Окно сообщения о программе	470
Окно сообщения <i>About Qt</i>	470
Окно сообщения об ошибке.....	471
Резюме	471
Глава 33. Предоставление помощи.....	473
Всплывающая подсказка.....	473
Подсказка «Что это».....	475
Система помощи (Online Help).....	476
Резюме	478
Глава 34. Главное окно, создание SDI- и MDI-приложений.....	480
Класс главного окна <i>QMainWindow</i>	480
Класс действия <i>QAction</i>	481
Панель инструментов	482
Доки	484
Строка состояния.....	485
Окно заставки.....	487
SDI- и MDI-приложения.....	489
SDI-приложение	489
MDI-приложение	493
Резюме	501
Глава 35. Рабочий стол (Desktop).....	502
Область уведомлений.....	502
Виджет экрана.....	507
Класс сервиса рабочего стола.....	511
Резюме	511
ЧАСТЬ VI. ОСОБЫЕ ВОЗМОЖНОСТИ QT	513
Глава 36. Работа с файлами, каталогами и потоками ввода/вывода	515
Ввод/вывод. Класс <i>QIODevice</i>	515
Работа с файлами. Класс <i>QFile</i>	517
Класс <i>QBuffer</i>	518
Класс <i>QTemporaryFile</i>	519
Работа с каталогами. Класс <i>QDir</i>	519
Просмотр содержимого каталога	520
Информация о файлах. Класс <i>QFileInfo</i>	523
Файл или каталог?	523
Путь и имя файла	524
Информация о дате и времени.....	524
Получение атрибутов файла	524
Определение размера файла	524

Наблюдение за файлами и каталогами	525
Потоки ввода/вывода.....	527
Класс <i>QTextStream</i>	527
Класс <i>QDataStream</i>	529
Резюме	529
Глава 37. Дата, время и таймер.....	531
Дата и время.....	531
Класс даты <i>QDate</i>	531
Класс времени <i>QTime</i>	533
Класс даты и времени <i>QDateTime</i>	534
Таймер	534
Событие таймера.....	535
Класс <i>QTimer</i>	537
Класс <i>QBasicTimer</i>	539
Резюме	539
Глава 38. Процессы и потоки.....	540
Процессы	540
Потоки	543
Приоритеты	545
Обмен сообщениями.....	545
Сигнально-слотовые соединения	546
Отправка событий.....	551
Синхронизация.....	554
Мьютексы	554
Семафоры	556
Ожидание условий.....	556
Возникновение тупиковых ситуаций	557
Фреймворк <i>QtConcurrent</i>	557
Резюме	559
Глава 39. Программирование поддержки сети	561
Сокетное соединение.....	561
Модель «клиент-сервер»	562
Реализация TCP-сервера	563
Реализация TCP-клиента.....	568
Реализация UDP-сервера и UDP-клиента.....	572
Управление доступом к сети	576
Блокирующий подход.....	583
Режим прокси.....	585
Резюме	586
Глава 40. Работа с XML	587
Основные понятия и структура XML-документа.....	587
XML и Qt.....	589
Работа с DOM	589
Чтение XML-документа	590
Создание и запись XML-документа	592

Работа с SAX.....	594
Чтение XML-документа	594
Класс <i>QXmlStreamReader</i> для чтения XML	597
Использование XQuery.....	599
Резюме	602
Глава 41. Программирование баз данных.....	603
Основные положения SQL.....	603
Создание таблицы.....	604
Операция вставки.....	604
Чтение данных	604
Изменение данных	605
Удаление	605
Использование языка SQL в библиотеке Qt	605
Соединение с базой данных (второй уровень)	607
Исполнение команд SQL (второй уровень)	608
Классы SQL-моделей для интервью (третий уровень)	611
Модель запроса.....	612
Табличная модель	613
Реляционная модель	615
Резюме	616
Глава 42. Динамические библиотеки и система расширений.....	617
Динамические библиотеки.....	617
Динамическая загрузка и выгрузка библиотеки.....	618
Расширения (plug-ins).....	621
Расширения для Qt.....	621
Поддержка собственных расширений в приложениях	623
Создание расширения для приложения	627
Резюме	629
Глава 43. Совместное использование Qt с платформозависимыми API	630
Совместное использование с Windows API.....	632
Совместное использование с Linux	635
Совместное использование с Mac OS X	635
Системная информация.....	639
Резюме	641
Глава 44. Qt Designer. Быстрая разработка прототипов.....	642
Создание новой формы в Qt Designer	642
Добавление виджетов.....	645
Компоновка (layout).....	646
Порядок следования табулятора.....	647
Сигналы и слоты	648
Использование в формах собственных виджетов	650
Использование форм в проектах	650
Компиляция.....	652
Динамическая загрузка формы.....	653
Резюме	655

Глава 45. Проведение тестов.....	657
Создание тестов	658
Тесты с передачей данных	661
Создание тестов графического интерфейса	663
Параметры для запуска тестов.....	664
Резюме	665
Глава 46. WebKit	666
Путешествие к истокам	667
А зачем?.....	668
Быстрый старт.....	668
Написание простого Web-браузера.....	670
Ввод адресов	670
Управление историей	670
Загрузка страниц и ресурсов.....	671
Пишем Web-браузер, попытка номер два.....	671
Резюме	676
Глава 47. Интегрированная среда разработки Qt Creator.....	677
Первый запуск.....	678
Создаем проект «Hello Qt Creator».....	679
Пользовательский интерфейс Qt Creator	684
Окна вывода	685
Окно проектного обозревателя.....	685
Секция компилирования и запуска.....	685
Редактирование текста	688
Как подсвечен ваш синтаксис?	688
Скрытие и отображение кода.....	688
Автоматическое дополнение кода	689
Поиск и замена	689
Комбинации клавиш для ускорения работы	694
Вертикальное выделение текста	694
Автоматическое форматирование текста	694
Комментирование блоков	694
Просмотр кода методов класса их определения и атрибутов	695
Помощь, которая всегда рядом	695
Использование стороннего редактора	696
Интерактивный отладчик и программный экзорцизм	696
Синтаксические ошибки.....	697
Ошибки компоновки.....	698
Ошибки времени исполнения	699
Логические ошибки	699
Трассировка.....	699
Команда <i>Step Over</i>	700
Команда <i>Step Into</i>	701
Команда <i>Step Out</i>	701
Контрольные точки.....	701
Окно переменных (Local and Watches)	702
Окно цепочки вызовов (Call Stack)	703
Резюме	703

Глава 48. Рекомендации по миграции программ из Qt 4 в Qt 5	705
Основные отличия Qt 5 от Qt 4	705
Подробности перевода на Qt 5	705
Виджеты	706
Контейнерные классы	706
Функция <i>qFindChildren<T*>()</i>	707
Сетевые классы	707
WebKit	707
Платформозависимый код	707
Система расширений Plug-ins	707
Принтер <i>QPrinter</i>	708
Мультимедиа	708
Модульное тестирование	708
Реализация обратной совместимости Qt 5 с Qt 4	708
Резюме	711
ЧАСТЬ VII. ЯЗЫК СЦЕНАРИЕВ QT SCRIPT	713
Глава 49. Основы поддержки сценариев	715
Принцип взаимодействия с языком сценариев	716
Первый шаг использования сценария	719
Привет, сценарий	720
Резюме	721
Глава 50. Синтаксис языка сценариев	723
Зарезервированные ключевые слова	723
Комментарии	724
Переменные	724
Предопределенные типы данных	725
Целый тип	725
Вещественный тип	725
Строковый тип	726
Логический тип	726
Преобразование типов	726
Константы	728
Операции	728
Операторы присваивания	728
Арифметические операции	728
Поразрядные операции	729
Операции сравнения	730
Приоритет выполнения операций	731
Управляющие структуры	732
Условные операторы	732
Оператор <i>if... else</i>	732
Оператор <i>switch</i>	733
Оператор условного выражения	733
Циклы	734
Операторы <i>break</i> и <i>continue</i>	734

Цикл <i>for</i>	734
Цикл <i>while</i>	734
Цикл <i>do...while</i>	735
Оператор <i>with</i>	735
Исключительные ситуации.....	735
Оператор <i>try...catch</i>	736
Оператор <i>throw</i>	736
Функции.....	737
Встроенные функции.....	738
Объектная ориентация.....	739
Статические классы.....	741
Наследование.....	742
Перегрузка методов.....	744
Сказание о «джейсоне».....	744
Резюме.....	745
Глава 51. Встроенные объекты Qt Script.....	746
Объект <i>Global</i>	746
Объект <i>Number</i>	746
Объект <i>Boolean</i>	746
Объект <i>String</i>	747
Преобразование строки к нижнему и верхнему регистрам.....	747
Замена.....	747
Получение символов.....	747
Получение подстроки.....	747
Объект <i>RegExp</i>	747
Проверка строки.....	748
Поиск совпадений.....	748
Объект <i>Array</i>	748
Дополнение массива элементами.....	749
Адресация элементов.....	749
Изменение порядка элементов массива.....	749
Преобразование массива в строку.....	750
Объединение массивов.....	750
Упорядочивание элементов.....	750
Многомерные массивы.....	750
Объект <i>Date</i>	751
Объект <i>Math</i>	752
Модуль числа.....	752
Округление.....	753
Определение максимума и минимума.....	753
Возведение в степень.....	753
Вычисление квадратного корня.....	753
Генератор случайных чисел.....	754
Тригонометрические методы.....	754
Вычисление натурального логарифма.....	754
Объект <i>Function</i>	755
Резюме.....	755

Глава 52. Классы поддержки Qt Script и практические примеры.....	756
Класс <i>QScriptValue</i>	756
Класс <i>QScriptContext</i>	756
Класс <i>QScriptEngine</i>	757
Практические примеры	759
«Черепашья» графика	760
Сигналы, слоты и функции	767
Отладчик Qt Script	770
Резюме	773
ЧАСТЬ VIII. ТЕХНОЛОГИЯ QT QUICK.....	775
Глава 53. Знакомство с Qt Quick.....	777
А зачем?.....	777
Введение в QML	779
Быстрый старт.....	779
Использование JavaScript в QML	783
Резюме	784
Глава 54. Элементы	786
Визуальные элементы	786
Свойства элементов.....	788
Собственные свойства	790
Создание собственных элементов.....	792
Готовые элементы пользовательского интерфейса	794
Диалоговые окна.....	797
Резюме	799
Глава 55. Управление размещением элементов	800
Фиксаторы	800
Традиционные размещения	807
Резюме	811
Глава 56. Элементы графики.....	812
Цвета	812
Растровые изображения	813
Элемент <i>Image</i>	813
Элемент <i>BorderImage</i>	816
Градиенты.....	817
Шрифты.....	818
Рисование на элементах холста	819
Резюме	821
Глава 57. Пользовательский ввод	822
Область мыши.....	822
Сигналы	825
Ввод с клавиатуры	829
Фокус	830
«Сырой» ввод	832
Резюме	834

Глава 58. Анимация	835
Анимация при изменении свойств	835
Анимация для изменения числовых значений	837
Анимация с изменением цвета	838
Анимация с поворотом	839
Анимации поведения	840
Параллельные и последовательные анимации	842
Состояния и переходы	845
Состояния	845
Переходы	848
Резюме	851
Глава 59. Модель/Представление	852
Модели	852
Модель списка	852
XML-модель	853
Представление данных моделей	855
Элемент <i>Flickable</i>	855
Элемент <i>ListView</i>	856
Элемент <i>GridView</i>	858
Элемент <i>PathView</i>	860
Резюме	863
Глава 60. Qt Quick и C++	864
Использование языка QML в C++	864
Использование компонентов языка C++ в QML	865
Резюме	877
Эпилог	878
ПРИЛОЖЕНИЯ	879
Приложение 1. Таблицы семибитной кодировки ASCII	881
Приложение 2. Таблица простых чисел	884
Приложение 3. Глоссарий	887
Приложение 4. Описание архива с примерами	891
Предметный указатель	901



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 такой пример не в состоянии продемонстрировать весь потенциал и возможности самой библиотеки, он дает представление о базовых понятиях. Этот пример позволяет оценить объем и сложность процесса реализации программ, использующих ту или иную библиотеку. Кроме того, на примере 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 <QtWidgets>
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 `QtWidgets`, представляющий собой файл модуля и включающий в себя заголовочные файлы для используемых в нашей программе классов: `QApplication` и `QLabel`. Конечно, мы могли бы обойтись и без

модуля `QtWidgets`, а непосредственно подключить заголовочные файлы для поддержки классов `QApplication` и `QLabel`, но при большем количестве включаемых классов, задействованных в программе, читаемость самой программы заметно бы ухудшилась. Кроме того, это ускоряет саму работу с кодом и, благодаря механизму предварительной компиляции заголовочных файлов (`Precompiled Headers`), не должно отразиться на скорости компиляции самой программы — конечно в том случае, если ваш компилятор ее поддерживает.

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, из 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 расходовало 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`, являющемся базовым для `QGuiApplication`, от которого унаследован класс `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

У программистов, начинающих изучение классов новой библиотеки, из-за большого объема информации, которую надо усвоить, зачастую создается ощущение перенасыщения. Но иерархия классов Qt имеет четкую внутреннюю структуру, которую важно сразу понять, чтобы потом уметь хорошо и интуитивно в этой библиотеке ориентироваться.

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 нашего примера из листинга 1.1 — это три модуля: `QtCore`, `QtGui` и `QtWidgets`, они присутствуют во всех программах с графическим интерфейсом и поэтому определены в программе создания `make-файлов` (см. главу 3) по умолчанию. Для использования других модулей в своих про-

Таблица 1.1. Некоторые модули Qt

Библиотека	Обозначение в проектном файле	Назначение
QtCore	core	Основополагающий модуль, состоящий из классов, не связанных с графическим интерфейсом
QtGui	gui	Модуль базовых классов для программирования графического интерфейса
QtWidgets	widgets	Модуль, дополняющий QtGui «строительным материалом» для графического интерфейса в виде виджетов на C++
QtQuick	quick	Модуль, содержащий описательный фреймворк для быстрого создания графического интерфейса
QtQML	qml	Модуль, содержащий движок для языка QML и JavaScript
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
QtXmlPatterns	xmlpatterns	Модуль поддержки XPath, XQuery, XSLT и XmlSchemaValidator
QtScript	script	Модуль поддержки языка сценариев
QtScriptTools	scripttools	Модуль дополнительных возможностей поддержки языка сценария. В настоящее время предоставляет отладчик
QtMultimedia	multimedia	Модуль мультимедиа
QtMultimediaWidgets	multimediawidgets	Модуль с виджетами для QtMultimedia
QtWebKit	webkit	Модуль для создания Web-приложений
QtWebKitWidgets	webkitwidgets	Модуль с виджетами для QtWebKit
QPrintSupport	printsupport	Модуль для работы с принтером
QtTest	test	Модуль, содержащий классы для тестирования кода

ектах необходимо перечислить их в проектном файле (см. главу 3). Например, чтобы добавить модули, нужно написать:

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

А чтобы исключить модуль из проекта:

```
QT -= gui
```

Наиболее значимый из перечисленных в табл. 1.1 модулей — это QtCore, так как он является базовым для всех остальных модулей. Далее идут модули, которые непосредственно зависят от QtCore, это: QtNetwork, QtGui, QSql и QtXml. И, наконец, модули, зависящие от только что упомянутых модулей: QtOpenGL и QtSvg.

Для каждого модуля Qt предоставляет отдельный заголовочный файл, содержащий заголовочные файлы всех классов этого модуля. Название такого заголовочного файла соответствует названию самого модуля. Например, для включения модуля QtWidgets нужно добавить в программу строку, как мы это уже сделали в листинге 1.1:

```
#include <QtWidgets>
```

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

- ◆ контейнерные классы: QList, QVector, QMap, QVariant, QString и т. д. (см. главу 4);
- ◆ классы для ввода и вывода: QIODevice, QTextStream, QFile (см. главу 36);
- ◆ классы процесса QProcess и для программирования многопоточности: QThread, QWaitCondition, QMutex (см. главу 38);
- ◆ классы для работы с таймером: QBasicTimer и QTimer (см. главу 37);
- ◆ классы для работы с датой и временем: QDate и QTime (см. главу 37);
- ◆ 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 28);
- ◆ класс приложения QCoreApplication, из объекта которого, если требуется, можно запустить цикл событий;
- ◆ классы поддержки анимации: QAbstractAnimation, QVariantAnimation и т. д. (см. главу 22);
- ◆ классы для машины состояний: QStateMachine, QState и т. д. (см. главу 22);
- ◆ классы моделей интервью: QAbstractItemModel, QStringListModel, QAbstractProxyModel (см. главу 12).

Модуль содержит так же механизмы поддержки файлов ресурсов (см. главу 3).

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, 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 предоставляет классы интеграции с оконной системой, с OpenGL и OpenGL ES. Он содержит класс `QWindow`, который является элементарной областью с возможностью получения событий пользовательского ввода, изменения фокуса и размеров, а так же позволяющей производить графические операции и рисование на своей поверхности.

Класс приложения этого модуля `QGuiApplication`. Этот класс содержит механизм цикла событий и обладает так же возможностями:

- ◆ получения доступа к буферу обмена (см. главу 29);
- ◆ инициализации необходимых настроек приложения — например, палитры для расцветки элементов управления (см. главу 13);
- ◆ управления формой курсора мыши.

Модуль `QtWidgets`

Этот модуль содержит в себе классы виджетов, представляющие собой «строительный материал» 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 он не что иное, как заполненный четырехугольник, но за этой внешней простотой скрывается большой потенциал простых функциональных возможностей. Этот класс насчитывает 254 метода и 53 свойства. В главе 5 ему 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 31 и 34);
- ◆ классы окон сообщений и диалоговых окон: `QMessageBox`, `QDialog` (см. главу 32);
- ◆ классы для рисования: `QPainter`, `QBrush`, `QPen`, `QColor` (см. главу 18);
- ◆ классы для растровых изображений: `QImage`, `QPixmap` (см. главу 19);

- ◆ классы стилей (см. главу 26) — 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 подробнее последний класс — класс `QApplication`, с которым мы встречались в самом первом примере. Все, что было сказано ранее о классе `QCoreApplication`, относится также и к этому классу, поскольку он является его наследником. Объект класса `QApplication` представляет собой центральный контрольный пункт Qt-приложений, имеющих пользовательский интерфейс на базе виджетов. Этот 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.

Класс `QApplication` напрямую унаследован от `QGuiApplication` и дополняет его следующими возможностями:

- ◆ установка стиля приложения. Таким способом можно устанавливать *виды и поведения* (Look & Feel) приложения, включая и свои собственные (см. главу 26);
- ◆ получение указателя на объект *рабочего стола* (desktop);
- ◆ управление глобальными манипуляциями с мышью (например, установка интервала двойного щелчка кнопкой мыши) и регистрация движения мыши в пределах и за пределами окна приложения;
- ◆ обеспечение правильного завершения работающего приложения при завершении работы операционной системы (см. главу 28).

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 подсакиванию значка приложения на док-панели в ОС Mac OS X (рис. 1.2) и его пульсации на панели задач в ОС Windows 7 (рис. 1.3).

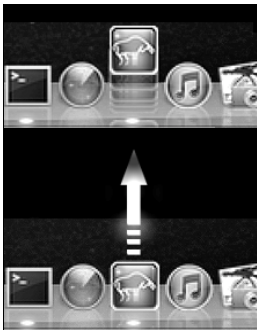


Рис. 1.2. Подсакивание значка приложения на док-панели в ОС Mac OS X

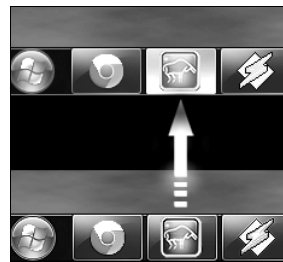


Рис. 1.3. Пульсация значка приложения на панели задач в Windows 7

Модули `QtQuick` и `QtQML`

Это альтернатива виджетам — модули, представляющие собой набор технологий для быстрой разработки графических интерфейсов нового поколения на базе описательного языка QML, языка программирования JavaScript и всех остальных возможностей библиотеки Qt (см. главу 53).

Модуль *QtNetwork*

Сетевой модуль *QtNetwork* предоставляет инструментарий для программирования TCP- и UDP-сокетов (классы *QTcpSocket* и *QUdpSocket*), а также для реализации программ-клиентов, использующих HTTP- и FTP-протоколы (класс *QNetworkAccessManager*). Этот модуль описывается в *главе 39*.

Модули *QtXml* и *QtXmlPatterns*

Модуль *QtXml* предназначен для работы с базовыми возможностями XML посредством SAX2- и DOM-интерфейсов, которые определяют классы Qt (см. *главу 40*). А модуль *QtXmlPatterns* идет дальше и предоставляет поддержку для дополнительных технологий XML — таких как: XPath, XQuery, XSLT и *XmlSchemaValidator*.

Модуль *QtSql*

Этот модуль предназначен для работы с базами данных. В него входят классы, предоставляющие возможность для манипулирования значениями баз данных (см. *главу 41*).

Модуль *QtOpenGL*

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

Модули *QtWebKit* и *QtWebKitWidgets*

Модуль *QtWebKit* позволяет очень просто интегрировать в приложение возможности Web. А модуль *QtWebKitWidgets* предоставляет готовые к интеграции в приложение элементы в виде виджетов с возможностью также расширять элементы Web своими собственными виджетами. Более подробно с этими модулями можно ознакомиться в *главе 46*.

Модули *QtMultimedia* и *QtMultimediaWidgets*

Модуль *QtMultimedia* обладает всем необходимым для создания приложений с поддержкой мультимедиа. Он поддерживает как низкий уровень, необходимый для более детальной специализированной реализации, так и высокий уровень, делающий возможным проигрывать видео- и звуковые файлы при помощи всего нескольких строк программного кода. Модуль *QtMultimediaWidgets* содержит полезные элементы в виде виджетов, которые позволяют экономить время для реализации. Более подробно с этими модулями можно ознакомиться в *главе 27*.

Модули *QtScript* и *QtScriptTools*

Модуль *QtScript* предоставляет возможности расширения и изменения уже написанных приложений при помощи языка сценариев JavaScript. Модуль *QtScriptTools* обеспечивает средства отладки для программ сценариев. Эти модули подробно описывает *часть VII*.

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).

Резюме

Библиотека Qt не является монолитной библиотекой, она разбита на отдельные модули: QtCore, QtGui, QtWidgets, QtQuick, QtQML, QtScript, QtMultimedia, QtWebKit, QtNetwork, QtOpenGL, QtSql, 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 QApplication должен быть создан в приложении только один раз.

Для реализации приложений с графическим интерфейсом пользователя необходимы модули QtWidgets или QtQuick. Классы QApplication и QApplication являются стержнем для Qt-приложений с графическим интерфейсом. Объект одного из этих классов не должен создаваться в приложении больше одного раза.



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 его правильно распознать. 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.

Таймер 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 37*.

Механизм объединения объектов в иерархические структуры позволяет резко сократить временные затраты при разработке приложений, не заботясь об освобождении памяти создаваемых объектов, поскольку 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.

Приведение типов. 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 задействованы в визуальной среде разработки пользовательского интерфейса Qt Designer (см. *главу 44*), механизм которой реализован в Qt при помощи директив препроцессора. Задается свойство использованием 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, 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, то есть тем, будет ли свойство запоминаться во время сохранения объекта.

Итак, теперь, когда вы познакомились с понятием «свойство» (хотя в ближайшее время этот механизм нам и не понадобится), давайте все равно 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 у них наверняка появятся трудности.

Существуют, впрочем, специальные библиотеки классов языка C++, облегчающие программирование для ОС Windows. Одной из самых первых таких библиотек (и до сих пор находящихся в применении у целого ряда индивидуальных разработчиков и компаний) является Microsoft Foundation Classes (MFC). Назвать ее объектно-ориентированной можно лишь с большой натяжкой, поскольку она создавалась людьми, не подозревающими о существовании самых элементарных принципов объектно-ориентированного подхода. Одна из главных 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 применяются 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)
END_MESSAGE_MAP()
```