

# Qt 4

## НА ПРИМЕРАХ + ДИСТРИБУТИВ

ПЛАТФОРМОНЕЗАВИСИМОЕ  
ПРОГРАММИРОВАНИЕ НА C++  
ДЛЯ WINDOWS, LINUX И MAC OS X

145 ПРОГРАММ: ОТ КОНСОЛЬНОГО  
ПРИЛОЖЕНИЯ ДО ПОЧТОВОГО  
КЛИЕНТА

БАЗЫ ДАННЫХ, ПОТОКИ, СЕТЕВЫЕ  
ПРИЛОЖЕНИЯ И ИНТЕРНЕТ, XML,  
XQUERY, ПОЛЬЗОВАТЕЛЬСКИЙ  
ИНТЕРФЕЙС, СТИЛИ, ГРАФИКА,  
СРЕДСТВА МУЛЬТИМЕДИА,  
Qt SCRIPT, PYTHON

НОВАЯ ВЕРСИЯ Qt 4.4

**Юрий Земсков**

**Qt 4**

**НА ПРИМЕРАХ**

Санкт-Петербург

«БХВ-Петербург»

2008

УДК 681.3.068+800.92Qt4  
ББК 32.973.26-018.1  
3-55

**Земсков Ю. В.**

3-55 Qt 4 на примерах. — СПб.: БХВ-Петербург, 2008. — 608 с.:  
ил. + Дистрибутив (на CD-ROM)

ISBN 978-5-9775-0256-6

Рассмотрена разработка приложений на языке C++ для Windows и/или Linux с применением библиотеки Qt 4, которая широко используется как в мире коммерческого, так и свободного программного обеспечения.

Приведены примеры решения многих задач, с которыми сталкивается разработчик при проектировании пользовательского интерфейса, сетевых взаимодействий, применении средств мультимедиа, языков XML и XQuery, многопоточном программировании. Особое внимание уделено созданию приложений баз данных и программ, которые могут расширяться конечным пользователем с помощью встроенных языков Qt Script или Python. Компакт-диск содержит исходные тексты описанных в книге примеров и библиотеку Qt 4.4 для Windows, Linux и Mac OS X.

*Для программистов*

УДК 681.3.068+800.92Qt4  
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>Леонид Кочин</i>
Компьютерная верстка	<i>Ольги Сергиенко</i>
Корректор	<i>Зинаида Дмитриева</i>
Дизайн серии	<i>Игоря Цырульниковца</i>
Оформление обложки	<i>Елены Беляевой</i>
Зав. производством	<i>Николай Тверских</i>

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 15.07.08.

Формат 70×100<sup>1/16</sup>. Печать офсетная. Усл. печ. л. 49,02.

Тираж 2500 экз. Заказ №

"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Отпечатано с готовых диапозитивов

в ГУП "Типография "Наука"

199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-9775-0256-6

© Земсков Ю. В., 2008

© Оформление, издательство "БХВ-Петербург", 2008

# Оглавление

<b>Введение</b> .....	<b>9</b>
Коммерческие и свободные версии Qt 4 .....	12
Благодарности .....	13
<b>Глава 1. Инструменты программиста</b> .....	<b>15</b>
1.1. Microsoft Visual C++ .....	16
1.2. Intel C++ .....	17
1.3. Borland C++ .....	17
1.4. GNU C++ и MinGW .....	17
1.5. QDevelop .....	17
1.6. Рекомендации по выбору компилятора .....	19
<b>Глава 2. "Обычный" C++</b> .....	<b>21</b>
2.1. Консольная программа на языке C++ .....	21
2.2. Компиляция и выполнение программы с помощью командной строки Microsoft Visual C++ .....	23
2.3. Компиляция и выполнение программы с помощью командной строки в системах Linux/FreeBSD/Solaris/HP-UX/Mac OS X .....	25
2.4. Создание консольного приложения в Microsoft Visual Studio 2005 .....	26
2.5. Шаблоны в C++ .....	30
<b>Глава 3. Основы Qt 4</b> .....	<b>35</b>
3.1. Обзор классов библиотеки Qt 4 .....	35
3.2. Компиляция библиотеки Qt .....	40
3.2.1. Компиляция в системе Microsoft Windows .....	41
3.2.2. Компиляция в системе Linux/FreeBSD .....	43
3.3. Простейшее приложение Qt 4 .....	44
3.4. Компиляция приложений Qt 4 из командной строки .....	45

3.5. Компиляция приложений Qt 4 с помощью интегрированной среды Microsoft Visual Studio.....	47
3.6. Использование кириллицы.....	48
3.7. Консольное приложение Qt.....	52
3.8. Создание новых классов.....	54
3.9. Правила именования объектов в библиотеке Qt.....	57
<b>Глава 4. Текстовые надписи .....</b>	<b>59</b>
4.1. Форматированный текст.....	60
4.2. Ссылки Интернета.....	61
4.3. Редактируемая надпись .....	62
4.4. Надписи с рисунками.....	63
4.5. Несколько элементов на одной форме .....	64
4.6. Диалог с текстовым сообщением .....	66
4.7. Форматирование строк .....	67
4.8. Ввод текста с клавиатуры.....	68
<b>Глава 5. Обработка событий.....</b>	<b>71</b>
5.1. Сигналы и слоты .....	71
5.2. События .....	79
5.3. Обработка событий с помощью виртуальных методов .....	80
5.4. Фильтры для событий.....	84
5.5. Сопоставление сигналов.....	88
<b>Глава 6. Кнопки и диалоги.....</b>	<b>91</b>
6.1. Кнопки нажатия <i>QPushButton</i> .....	91
6.2. Программная эмуляция нажатия кнопок .....	97
6.3. Группы переключателей.....	99
6.4. Группа кнопок <i>QButtonGroup</i> .....	101
6.5. Кнопки диалогов .....	103
6.6. Модальные диалоги .....	104
6.7. Стандартные диалоги.....	108
6.8. Окно внутри диалога.....	110
<b>Глава 7. Меню, панели инструментов и строка состояния .....</b>	<b>113</b>
7.1. Строка состояния .....	113
7.2. Действия .....	115
7.3. Меню.....	115
7.4. Виджеты в строке состояния.....	116
7.5. Примеры приложений .....	118
<b>Глава 8. Размещение элементов на форме .....</b>	<b>127</b>
8.1. "Ручное" размещение элементов формы .....	127
8.2. Менеджеры размещения.....	134

<b>Глава 9. Автоматизация создания диалогов, отладка программ и предоставление помощи .....</b>	<b>139</b>
9.1. Создание диалога с помощью Qt Designer .....	139
9.2. Использование диалога, созданного в Qt Designer .....	145
9.3. Отладка программ .....	150
9.4. Система помощи .....	153
<b>Глава 10. Работа с данными .....</b>	<b>159</b>
10.1. Числа и строки .....	159
10.2. Массивы и списки .....	160
10.3. Контейнеры и итераторы .....	163
10.4. Класс <i>QObject</i> .....	165
10.5. Неявное совместное использование данных .....	167
10.6. Явное совместное использование данных .....	170
<b>Глава 11. Каталоги, файлы, потоки ввода-вывода, ресурсы .....</b>	<b>173</b>
11.1. Чтение и запись текстового файла .....	173
11.2. Работа с двоичными данными .....	176
11.3. Каталоги и свойства файлов .....	180
11.4. Временные файлы .....	180
11.5. Каталог приложения .....	181
11.6. Копирование файлов .....	181
11.7. Хранение ресурсов в программе .....	183
11.8. Хранение скомпилированных ресурсов в отдельном двоичном файле .....	184
11.9. Значок приложения .....	185
11.10. Диалог выбора файла .....	185
11.11. Сжатие информации .....	186
<b>Глава 12. Приложения SDI и MDI .....</b>	<b>189</b>
12.1. Пример приложения SDI: текстовый редактор .....	189
12.2. Открытие и сохранение файлов .....	194
12.3. Команды редактирования .....	201
12.4. Пример приложения MDI .....	203
12.5. Плавающие окна .....	211
12.6. Хранение настроек приложения .....	214
12.7. Разное .....	219
<b>Глава 13. Графика и печать .....</b>	<b>221</b>
13.1. Класс <i>QPainter</i> .....	222
13.2. Пользовательский стиль линий .....	227
13.3. Форматированный текст и HTML .....	228
13.4. Отображение форматированного текста .....	232
13.5. Часы со стрелками .....	232

13.6. Класс <i>QPixmap</i> .....	237
13.7. Работа с принтером .....	239
13.8. Печать на нескольких страницах .....	250
13.9. Предварительный просмотр перед печатью .....	252
13.10. Графические сцены .....	254
<b>Глава 14. Списки, таблицы и деревья .....</b>	<b>263</b>
14.1. Список <i>QListWidget</i> .....	263
14.2. Реакция на действия пользователя .....	264
14.3. Редактирование элементов списка .....	267
14.4. Таблица <i>QTableWidget</i> .....	267
14.5. Дерево <i>QTreeWidget</i> .....	278
<b>Глава 15. Технология "Модель — представление" .....</b>	<b>287</b>
15.1. Табличная модель и ее представления .....	287
15.2. Выравнивание элементов списка .....	299
15.3. Галочки для элементов таблицы .....	300
15.4. Дерево каталогов и файлов .....	303
15.5. Иерархическая модель .....	306
15.6. Дерево групп и таблица элементов .....	313
15.7. Разное .....	323
<b>Глава 16. Процессы, потоки, синхронизация .....</b>	<b>333</b>
16.1. Процессы .....	333
16.2. Потоки .....	335
16.3. Реентерабельность и потокобезопасность .....	336
16.4. Методы синхронизации процессов и потоков .....	338
16.4.1. Мьютексы .....	338
16.4.2. Читатели и писатели .....	341
16.4.3. Семафоры .....	341
16.4.4. Условие ожидания .....	342
16.5. Производитель — потребитель .....	343
16.6. Мультипликация .....	346
16.7. <i>QtConcurrent</i> : высокоуровневое API для параллельного программирования ...	350
<b>Глава 17. Работа с базами данных .....</b>	<b>359</b>
17.1. Компиляция SQL-драйверов .....	359
17.2. Подключение к базе данных и выполнение SQL-запросов .....	362
17.3. Работа с таблицами баз данных .....	374
<b>Глава 18. Модели таблиц баз данных и их представления .....</b>	<b>381</b>
18.1. Разработка модели и представления таблицы БД .....	381
18.2. Делегаты для ячеек таблицы .....	389

18.3. Связывание элементов управления форм с моделью данных .....	399
18.4. Консоль запросов .....	408
18.5. Связывание таблиц .....	420
18.6. Выполнение запросов в отдельных потоках .....	422
<b>Глава 19. Средства XML.....</b>	<b>431</b>
19.1. DOM API.....	432
19.2. SAX API .....	437
19.3. Класс <i>QStreamReader</i> .....	440
19.4. Модуль <i>QtXmlPatterns</i> .....	443
<b>Глава 20. Локальные сети и Интернет .....</b>	<b>447</b>
20.1. Эхо-сервер в блокирующем режиме .....	449
20.2. Клиент для эхо-сервера .....	453
20.3. Асинхронный однопоточный эхо-сервер .....	456
20.4. Многопоточный эхо-сервер .....	460
20.5. Работа с протоколом FTP .....	466
20.6. Отправка электронной почты по протоколу SMTP .....	467
20.7. Использование модуля WebKit: простой Web-браузер .....	475
<b>Глава 21. Использование модуля QtScript.....</b>	<b>479</b>
<b>Глава 22. Технология COM. Интеграция с MS Office. Средства мультимедиа.....</b>	<b>495</b>
22.1. Импорт и экспорт таблиц MS Excel .....	495
22.2. Работа со звуком .....	501
22.3. Анимация .....	502
22.4. Библиотека Phonon.....	503
22.5. Воспроизведение видео с помощью Media Player.....	507
<b>Глава 23. Внешний вид элементов управления .....</b>	<b>509</b>
23.1. Палитры .....	509
23.2. Стили.....	511
<b>Глава 24. Интернационализация приложений.....</b>	<b>519</b>
24.1. Подготовка исходного текста программы .....	519
24.2. Подготовка файлов описания проекта и ресурсов .....	522
24.3. Утилиты <i>lupdate</i> и <i>lrelease</i> .....	526
24.4. Программа Qt Linguist .....	526
24.5. Переключение языка во время выполнения программы .....	528
<b>Глава 25. Использование интерпретатора Python.....</b>	<b>531</b>
25.1. Зачем это нужно .....	531
25.2. Основные сведения о языке Python .....	531



25.3. Встраивание интерпретатора Python в приложения, написанные на C/C++ .....	534
25.4. Библиотека PythonQt.....	545
25.5. Библиотека PyQt4.....	547
<b>Глава 26. Создание библиотек и плагинов для Qt Designer .....</b>	<b>549</b>
26.1. Создание библиотеки.....	549
26.2. Статическая компоновка .....	551
26.3. Динамическая загрузка .....	553
26.4. Расширение библиотеки виджетов Qt Designer .....	555
<b>Глава 27. Сравнение библиотеки Qt с другими средствами межплатформенной разработки .....</b>	<b>565</b>
27.1. Qt и Java .....	565
27.2. Qt и .NET .....	566
27.3. Qt и wxWidgets .....	567
<b>Глава 28. Разное.....</b>	<b>573</b>
28.1. Борьба с "зависанием" интерфейса пользователя .....	573
28.2. Перемещение виджетов с помощью мыши .....	573
28.3. Заголовок окна .....	575
28.4. Область прокрутки.....	582
28.5. Сведения о текущей платформе.....	582
28.6. Бегущая строка .....	585
28.7. Работа с переменными окружения .....	587
28.8. Типы MIME .....	588
28.9. Буфер обмена.....	589
28.10. Перетаскивание файлов в окно текстового редактора.....	591
<b>Заключение.....</b>	<b>595</b>
<b>Приложение. Описание содержимого компакт-диска .....</b>	<b>597</b>
<b>Список литературы .....</b>	<b>603</b>
<b>Предметный указатель .....</b>	<b>605</b>

# Введение

Если вы возьмете любую работу, любое искусство, любую отрасль знания, любое умение и продвинете их так далеко вперед, как только возможно, — так далеко вперед, как они еще никогда не бывали, до самой границы границ неизведанного, то вы заставите их перейти в царство магии и волшебства.

*Том Роббинс, писатель*

Мы должны быть благодарны Господу за то, что Он создал мир таким, что все простое в нем истинно, а все сложное — ложно.

*Григорий Сковорода,  
украинский философ (1722–1794)*

Когда мне хочется прочесть роман, я сам пишу его.

*Б. Дизраэли, лорд Биконсфилд,  
британский государственный деятель и писатель*

Библиотека Qt норвежской фирмы Trolltech представляет собой набор классов C++ и инструментов разработки программ для Microsoft Windows, Linux, FreeBSD, Solaris, HP-UX, Mac OS X и встраиваемых систем (*Embedded Linux*). На всех платформах библиотека Qt использует свой собственный набор визуальных элементов, в результате приложения, созданные на ее основе, во всех системах выглядят и работают одинаково.

Поскольку Qt — удобный и мощный инструмент, многие разработчики применяют ее не только для построения платформонезависимых приложений, но также и при создании программ, которые должны работать в конкретной операционной системе (Linux или Windows). Являясь образцом объектно-ориентированной библиотеки, Qt включает в себя классы для работы со структурами данных, файлами и каталогами, сетевыми протоколами, потоками и процессами, 2D- и 3D-графикой, OpenGL, базами данных, форматами HTML и XML, обширной коллекцией виджетов для построения графического интерфейса пользователя и многие другие возможности, облегчающие процесс создания профессиональных приложений на всех его этапах: от про-

граммирования заставки, показываемой при запуске, до разработки системы помощи.

Вместе с библиотекой поставляются средства для быстрой разработки программ: редактор форм Qt Designer, модуль интеграции с Microsoft Visual Studio 6.0, .NET 2003 и .NET 2005, программа для интернационализации приложений Qt Linguist, система помощи Qt Assistant с подробной документацией и примерами программ. Подобно программным продуктам Microsoft Office для Windows, функциональные возможности которых можно расширять с помощью встроенного языка Visual Basic, в приложения Qt тоже может быть встроен свой скриптовый язык Qt Script.

По сравнению с предыдущей версией библиотеки (Qt 3) структура классов Qt 4 существенно изменилась, поэтому для старых приложений Qt 3 необходима переработка исходного текста. Хотя процедура конвертирования в достаточной степени автоматизирована (имеется утилита `qt3to4`), но в серьезных проектах без "ручной" работы не обойтись.

Библиотека Qt — безусловный лидер среди имеющихся средств разработки межплатформенных программ на языке C++. Широко известная и часто используемая в мире Linux, она, благодаря распространению графической оболочки KDE, стала де-факто стандартом проектирования программного обеспечения на этой платформе. Для разработчиков Windows-приложений библиотека Qt долгое время не выходила на передний план, поскольку для Windows существовали более доступные и удобные средства быстрой разработки программ. Но в последнее время расстановка сил в корне изменилась. Новая, 4-я версия Qt, наконец, "дотянулась" по своим возможностям до тех вершин, на которых долгое время господствовали Microsoft и Borland/Inprise (последняя, к слову сказать, когда речь зашла о межплатформенном инструменте, не нашла ничего лучше, как обратиться к библиотеке Qt, но результат, к сожалению, оказался хуже первоисточника). К тому же версия Qt для Windows наконец-то стала свободной, а не только коммерческой, как это было раньше.

Исходные тексты библиотеки открыты, но лицензия GPL требует, чтобы программы, которые разрабатываются на основе Qt, распространялись с открытым исходным кодом. Поэтому если вы не желаете открывать исходный код своей программы, то должны приобрести коммерческую версию Qt.

Библиотека Qt используется многими программистами во всем мире. Среди корпоративных клиентов Trolltech такие известные компании, как Synopsys (автоматизированные системы проектирования электронных схем), Walt Disney Feature Animation (компьютерная анимация), Skype (общение голосом через Интернет), Volvo (применение Qtopia Core для разработки человеко-машинного интерфейса между водителем и бортовым компьютером, рабо-

тающим под управлением Linux), Adobe (программа Adobe Photoshop Album), Google (электронная карта земной поверхности Google Earth), Wolfram (система компьютерной математики Mathematica), Perforce (мощная система управления конфигурированием программного обеспечения — Software Configuration Management, SCM), Samsung и NXP Semiconductors (программное обеспечение смартфонов), Siemens (программное обеспечение для контроля производственной линии), NASA (моделирование условий полета космических кораблей) и еще более 5000 фирм — зарегистрированных пользователей Qt.

Данная книга представляет собой сборник готовых решений многих распространенных проблем, с которыми сталкивается разработчик, использующий библиотеку Qt. Автор надеется, что приведенные рецепты помогут вам освоить большинство возможностей этой замечательной библиотеки и пополнить обширный список программ на сайтах <http://sourceforge.net>, [www.qt-apps.org](http://www.qt-apps.org) и <http://trolltech.com/customers/coolapps>.

Книга состоит из 28 глав. В *главе 1* рассмотрены основные инструменты программиста, *глава 2* поможет читателю вспомнить основы языка C++. *Глава 3* начинает наше знакомство с библиотекой Qt. В *главе 4* обсуждаются основы построения интерфейса, *глава 5* посвящена методам обработки событий, в *главе 6* рассмотрены кнопки и диалоги, а в *главе 7* — меню, панели инструментов и строка состояния. В *главе 8* рассмотрены менеджеры размещения, в *главе 9* — визуальное проектирование интерфейса с помощью программы Qt Designer и разработка системы помощи. В *главе 10* обсуждаются методы работы с данными различного типа, в *главе 11* — каталоги и файлы. В *главе 12* описаны интерфейсы SDI и MDI, в *главе 13* обсуждаются графические инструменты и работа с принтером. *Глава 14* посвящена составным элементам интерфейса: спискам, таблицам и деревьям. В *главе 15* вводится понятие технологии "Модель — представление". В *главе 16* обсуждаются методы использования процессов и потоков. *Главы 17 и 18* посвящены разработке приложений баз данных. В *главе 19* описаны средства XML, а в *главе 20* — работа с сокетами и модулем WebKit. В *главе 21* мы знакомимся со скриптовым языком Qt, в *главе 22* применяем технологию COM для работы с приложениями Microsoft Office, а также обсуждаем мультимедийные возможности библиотеки Qt. В *главе 23* описана работа с палитрами и стилями, определяющими внешний вид визуальных элементов. *Глава 24* посвящена технологии локализации интерфейса приложения. В *главе 25* мы научимся встраивать в Qt-приложения интерпретатор языка Python, а в *главе 26* — создавать динамические библиотеки и плагины для Qt Designer. В *главе 27* обсуждаются и сравниваются известные технологии разработки межплатформенных приложений. В *главе 28* собраны частные вопросы, которые не попали в другие

разделы книги. В приложении приведено описание компакт-диска. Прилагаемый к книге компакт-диск содержит исходные тексты всех программ.

Особое внимание в книге уделено вопросам разработки приложений баз данных, программированию сетевых взаимодействий и многопоточных задач. Рассмотрены новые возможности библиотеки Qt, появившиеся в релизе 4.4 (май 2008 г.): мультимедийная библиотека Phonon, модули WebKit и QtConcurrent.

Книга рассчитана на читателей, имеющих хотя бы минимальный опыт разработки программ на языке C++ для Windows или Linux, также будет полезна студентам-программистам и всем, кого интересуют вопросы межплатформенной разработки приложений.

Приводимые решения, если это не оговорено особо, тестировались в системах Microsoft Windows XP и Vista, Mandriva Linux 2008, Ubuntu 7.10 и Knoppix 5.

## Коммерческие и свободные версии Qt 4

Коммерческие версии Qt 4: *Qt Console Edition*, *Qt Desktop Light Edition* и *Qt Desktop Edition* различаются составом инструментальных средств разработки, поддерживаемыми функциями и компонентами API, а также, разумеется, ценой. Самый полный набор функциональных возможностей имеет *Qt Desktop Edition*.

Свободная (т. е. некоммерческая, бесплатная) версия *Qt Open Source Edition*, предназначенная только для создания программ с открытым исходным кодом, доступна для платформ Microsoft Windows, UNIX/X11 и Mac OS X и включает все средства разработки и компоненты API, имеющиеся в коммерческой *Qt Desktop Edition*, кроме:

- драйверов для доступа к коммерческим СУБД Oracle, DB2 и Sybase;
- модуля ActiveQt для работы с COM/ActiveX в Windows;
- поддержки коммерческих компиляторов (например, отсутствует интеграция с Microsoft Visual Studio).

Некоммерческую редакцию библиотеки Qt для Linux и Windows можно скачать с сайта <http://trolltech.com>. Там же имеется подробная документация по последним релизам библиотеки Qt и другим продуктам фирмы Trolltech (к сожалению, пока только на английском, китайском и японском языках).

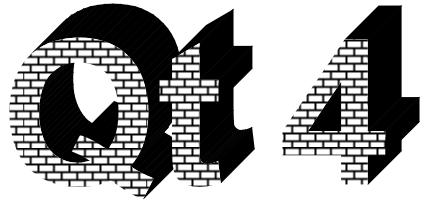
Заметим, что исходные тексты на языке C++ предоставляются как для коммерческих, так и для свободной версии библиотеки Qt. С помощью коммерческих версий Qt могут разрабатываться программы как с открытым, так и с закрытым исходным кодом.

## Благодарности

Автор выражает признательность Игорю Владимировичу Шишигину и Максиму Шлее за предложение написать эту книгу, а также Леониду Борисовичу Кочину — за помощь при подготовке книги к печати. Благодарю своих научных руководителей Эдуарда Ивановича Чаплыгина, Валерия Петровича Шевчука и Анатолия Павловича Желтоногова. Большое спасибо моей жене Ирине и дочке Женьке за терпение и понимание.



# ГЛАВА 1



## Инструменты программиста

Самое важное в политике — следовать своей цели: средства ничего не значат.

*Наполеон I*

Для редактирования исходного текста программ прежде всего необходим какой-нибудь текстовый редактор. Хорошо, если он выполняет подсветку синтаксиса, умеет форматировать листинги, дает подсказку по текущему объекту и выводит список членов нужного класса. Кроме того, не обойтись без компилятора и отладчика. Все эти инструменты могут быть включены в состав единой, так называемой *интегрированной среды разработки приложений* (IDE, Integrated Development Environment).

Для каждой операционной системы, в которой (или для которой) вы пишете приложение, применяются свои инструменты. Но разработчики библиотеки Qt побеспокоились о том, чтобы их продукт был совместим с большинством из них.

Если вы держите эту книгу в руках, то, вероятно, знаете, что для языка C++ существует множество компиляторов. Самые распространенные для системы Microsoft Windows — это компиляторы Microsoft Visual C/C++, Borland C/C++, Intel C/C++, Watcom C/C++, Digital Mars C/C++ и MinGW. В системах Linux/FreeBSD чаще всего используется G++, входящий в состав коллекции компиляторов GCC (GNU Compiler Collection), но также имеется и компилятор от Intel.

Более подробную информацию о совместимости конкретных релизов библиотеки Qt с различными версиями компиляторов можно найти в документации, поставляемой вместе с библиотекой, в разделах *Compiler Notes* (Замечания по отношению компиляторов) и *Platform-Specific Notes* (Замечания по используемым платформам), а также на сайте фирмы Trolltech по адресу <http://trolltech.com/developer/notes/>.



## 1.1. Microsoft Visual C++

Коммерческие версии Qt работают с компиляторами Microsoft Visual C++ 6.0, .NET 2003, .NET 2005 и свободно распространяемым Visual Studio 2005 Express Edition (в последнем случае необходимо отдельно установить Platform SDK). Для Microsoft Visual C++ 6.0 требуется Service Pack 5. Сообщается, что поддержка Microsoft Visual C++ 6.0 в скором времени, вероятно, будет прекращена.

В коммерческой версии Qt для Windows предусмотрена интеграция с Microsoft Visual Studio (рис. 1.1).

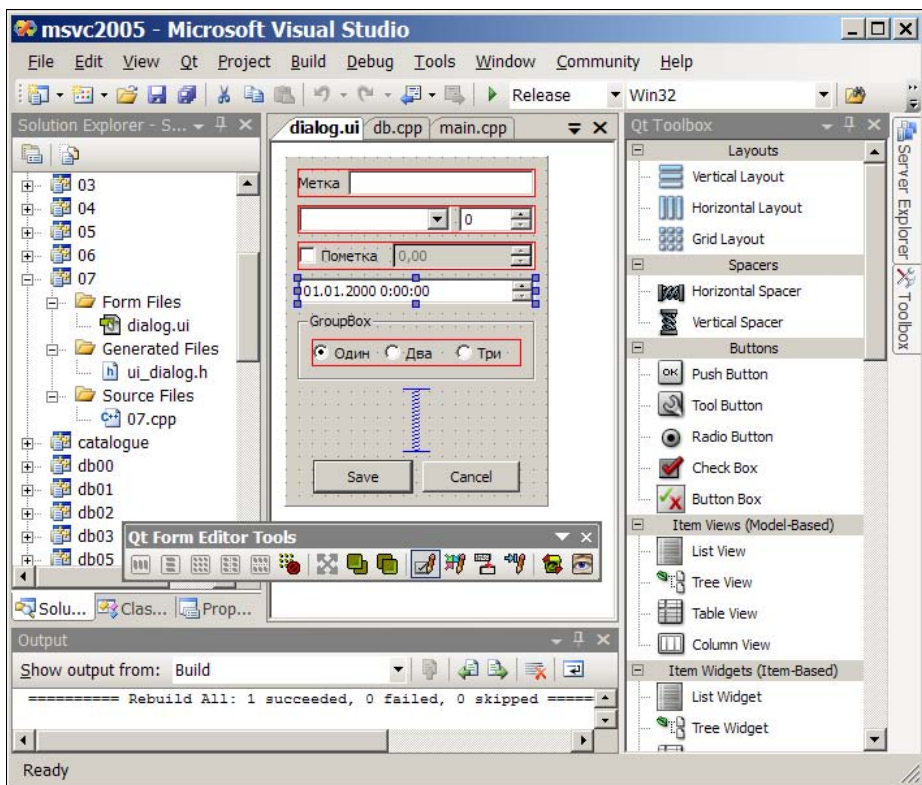


Рис. 1.1. Проект Qt в Microsoft Visual Studio 2005

Начиная с релиза Qt 4.3, компилятор Microsoft Visual C++ наконец-то может использоваться и со свободной версией *Qt Open Source Edition* (но интеграция, по-прежнему, доступен только для коммерческих версий Qt).

## 1.2. Intel C++

Коммерческие версии библиотеки Qt 4 поддерживают 32- и 64-разрядный Intel C++ 10.0 и 10.1 для Linux и 8.1 для Windows. Однако в Intel C++ 10.0 недопустимы предкомпилированные заголовочные файлы. Также известны проблемы при компиляции в режиме Release с помощью Intel C++ 10.1.

## 1.3. Borland C++

К сожалению, Qt 4 (в отличие от Qt 3) официально не поддерживает компилятор Borland C/C++, но соответствующие файлы конфигурации в составе библиотеки Qt 4 еще остались, их можно найти в каталоге `mkspcs/win32-borland`. Хотя утилита `configure` по-прежнему воспринимает ключ `-platform win32-borland`, но требуются дополнительные ухищрения, чтобы собрать библиотеку и свои программы с помощью компилятора Borland.

## 1.4. GNU C++ и MinGW

Для некоммерческой библиотеки *Qt Open Source Edition* в системе Windows до недавнего времени существовал только свободно распространяемый компилятор MinGW, а в Linux — GCC 3.2 и более поздние версии. Разумеется, GCC и MinGW подходят и для коммерческих версий Qt.

Для начинающих автор рекомендует использовать MinGW Developer Studio — свободно распространяемую интегрированную среду разработки программ на C++ для Windows. Она доступна в Интернете по адресу [www.paritysoft.com](http://www.paritysoft.com) (вместе с компилятором MinGW32).

## 1.5. QDevelop

Конечно, с исходным текстом программ можно работать в любом текстовом редакторе, а компиляцию проводить с помощью командной строки. Но удобнее (во всяком случае, так считают те, кто мало знаком с миром UNIX/Linux-систем) разрабатывать программы в какой-нибудь интегрированной среде (IDE). В состав библиотеки Qt входит утилита Qt Designer, с помощью которой можно в диалоговом режиме проектировать графический интерфейс приложений. Но она не является интегрированной средой разработки, т. к. в ней нет текстового редактора и отладчика (как было объявлено, в 4-й версии программисты Trolltech отказались от идеи превратить Qt Designer в полноценную среду разработки, а сосредоточились на вопросах интеграции его со стандартными инструментами).

В Linux, где Qt стала де-факто стандартом разработки программ, с этим особых проблем не возникает, поскольку практически любая IDE (например, KDevelop) поддерживает работу с Qt.

С системой Windows все, как обычно, сложнее. В коммерческой версии Qt для Windows предусмотрена интеграция с Microsoft Visual Studio, но в *Open Source Edition* данная возможность отсутствует. Поэтому в рамках направления Open Source в настоящее время разрабатываются несколько похожих проектов, способных претендовать на звание "IDE для Qt". Среди них стоит отметить QDevelop <http://qdevelop.org> (рис. 1.2).

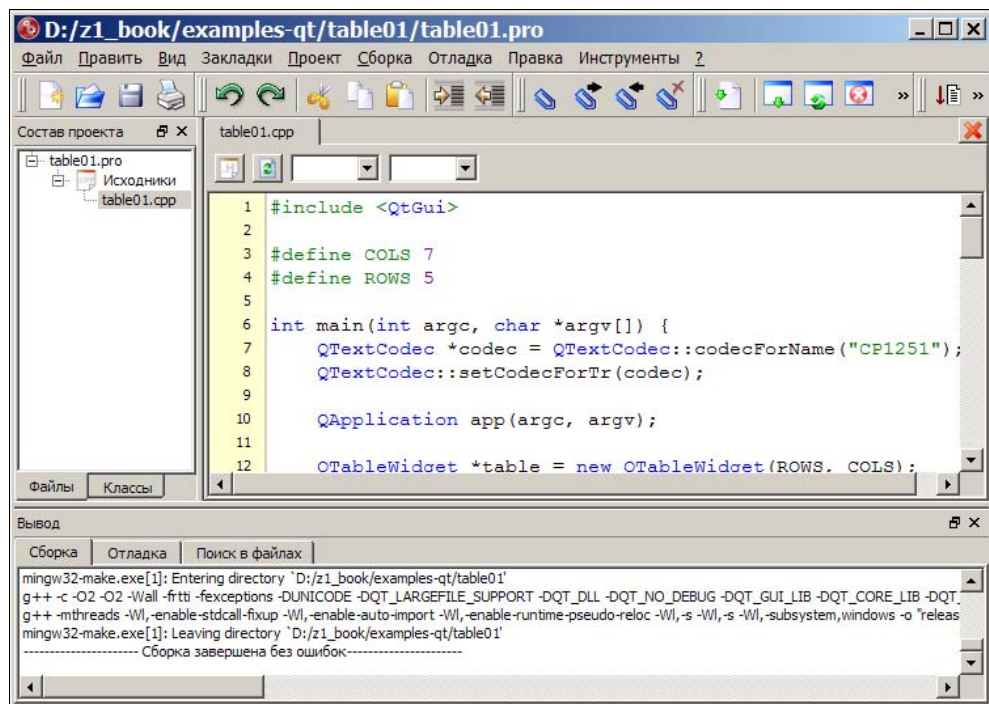


Рис. 1.2. Интегрированная среда разработки QDevelop

Сама программа QDevelop написана с использованием Qt, поэтому ее исходные тексты можно скомпилировать на любой платформе, которая поддерживает библиотеку Qt. Хотя к январю 2008 г. был выпущен только предварительный релиз 0.25, но программа уже вполне работоспособна: имеется русский перевод интерфейса, система контекстной помощи, работает отладчик, подсветка синтаксиса, а также автодополнение кода (для последнего требуется утилита `ctags`, ее можно скачать с сайта <http://ctags.sourceforge.net>).

После компиляции программы QDevelop (или после ее инсталляции, если вы не захотели возиться с исходными текстами) нужно в меню **Инструменты** выбрать пункт **Инструменты** и указать пути к инструментам библиотеки Qt, компилятору и отладчику (рис. 1.3).

В остальном интерфейс программы стандартен и достаточно удобен, поэтому работа с ней не составит трудностей для любого, кто знает какую-либо другую интегрированную среду разработки.

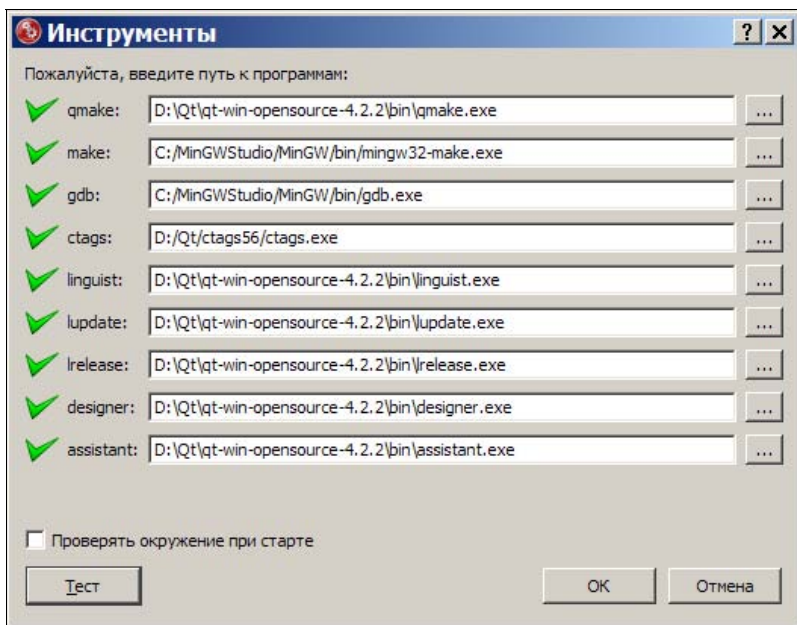


Рис. 1.3. Настройка параметров QDevelop

## 1.6. Рекомендации по выбору компилятора

Совет — это теория жизни, а практика жизни обычно совсем другая.

*Пауло Козльо*

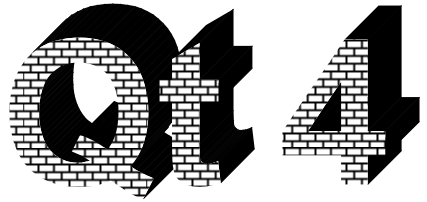
Если читатель собирается компилировать программы только для системы Windows, то, несомненно, лучшим выбором будет Microsoft Visual C++. При отсутствии коммерческого компилятора подойдет компилятор MinGW вместе с отладчиком GDB и интегрированной средой QDevelop.

Для разработки приложений в системе Linux однозначно следует выбрать GNU C++.

Если приложение, разрабатываемое в Windows, предполагается затем компилировать в Linux, то целесообразнее сразу обратиться к MinGW, чтобы затем безболезненно перейти на GNU C++. И наоборот, приложение, разработанное в Linux с помощью GNU C++, в систему Windows лучше всего переносить с помощью MinGW.

Если у вас уже имеется какой-нибудь любимый компилятор, поддерживаемый библиотекой Qt, то можете игнорировать приведенные рассуждения и смело использовать привычные инструменты.

# ГЛАВА 2



## "Обычный" C++

Всегда можно начать сначала. Не потому ли столько людей ходят по кругу.

*Андрей Сток*

Прежде чем приступить к обсуждению библиотеки Qt, рассмотрим некоторые концепции C++, которые пригодятся нам в дальнейшем.

### 2.1. Консольная программа на языке C++

**ЗАДАЧА.** Написать на языке C++ простую программу, которая запрашивает исходные данные (два целых числа) и выводит результат их сложения.

**Решение.** В листинге 2.1 приведен текст *консольной* (не имеющей графического интерфейса) программы, которая решает поставленную задачу.

#### Листинг 2.1. Простейшее консольное приложение

```
1 // Сложение двух целых чисел (консольное приложение)
2
3 #include <iostream>
4
5 int main() {
6
7     using namespace std;
8     int a, b;
9
10    cout << endl << "a = ";
11    cin >> a;
12
13    cout << "b = ";
14    cin >> b;
```

```

15
16     int c = a + b;
17     cout << a << " + " << b << " = " << c << endl;
18
19     return 0;
20 }

```

### Пояснения к программе\*:

- (1) — строка, начинающаяся двойным слэшем, содержит комментарий;
- (3) — с помощью директивы `#include` подключается так называемый заголовочный файл с определениями внешних классов и функций. В данном случае это библиотека ввода-вывода `iostream` (в старых книгах вы можете встретить `iostream.h`);
- (5) — главная функция, с которой начинается выполнение программы, всегда называется `main`. Она должна возвращать результат типа `int` (целое число), которое обычно считается кодом ошибки. Если при выполнении программы ошибок не возникло, то возвращается число 0. Вообще говоря, могут присутствовать также входные параметры командной строки, но в нашем простом проекте они не нужны. Открывающая фигурная скобка начинает программный блок, относящийся к главной функции приложения;
- (7) — директива `using namespace` указывает, какое *пространство имен* используется по умолчанию. Если ее опустить, то во всей программе вместо идентификаторов `cin` (поток ввода данных), `cout` (поток вывода) и `endl` (конец строки) пришлось бы писать `std::cin`, `std::cout` и `std::endl`;
- (8) — объявили две переменные целого типа;
- (10) — в стандартный поток вывода `std::cout` (как правило, это экран монитора) вывели сначала код перехода на новую строку `endl`, а затем приглашение ввести значение переменной `a`;
- (11) — прочитали значение переменной `a` из стандартного потока ввода `std::cin` (обычно это клавиатура);
- (13, 14) — вывели приглашение на ввод второй переменной `b` и прочитали введенное значение;
- (16) — объявили новую переменную `c` типа `int`, в которую сразу записали результат сложения двух введенных чисел;

---

\* Здесь и далее в круглых скобках указаны номера строк листинга. — *Ред.*

- (17) — вывели значение переменной `a`, символ сложения, значение переменной `b`, символ равенства, результат сложения и выполнили переход на новую строку;
- (19) — вышли из функции `main()`, указав нулевой результат — признак того, что программа завершилась без ошибок;
- (20) — закрывающая фигурная скобка оканчивает программный блок, относящийся к функции `main()`.

Это наша первая программа, поэтому здесь мы не отвлекаемся на анализ корректности вводимых пользователем данных (попробуйте ввести буквенный символ или вещественное число с дробной частью), а также возможное переполнение при сложении очень больших чисел.

## 2.2. Компиляция и выполнение программы с помощью командной строки Microsoft Visual C++

**ЗАДАЧА.** С помощью Microsoft Visual C++ выполнить компиляцию и сборку консольной программы, текст которой приведен в листинге 2.1, а также проверить правильность ее выполнения.

### Решение.

1. Наберем в любом текстовом редакторе исходный текст программы (см. листинг 2.1). Сохраним его в отдельном каталоге, в файле под именем `console1.cpp`.
2. Откроем командную строку Visual Studio 2005 Command Prompt, для чего выполним команды меню **Пуск | Программы | Microsoft Visual Studio 2005 | Visual Studio Tools | Visual Studio 2005 Command Prompt**. В результате на экране появится окно MS-DOS, но предварительно будут настроены переменные окружения, необходимые для работы компилятора и компоновщика.
3. Для удобства запустим в открывшемся консольном окне какой-нибудь текстовый файловый менеджер, например, `Far`:  

```
"C:\Program Files\Far\far.exe"
```

Кавычки необходимы, поскольку полное имя исполняемого файла содержит пробел. Если на вашем компьютере не установлен `Far`, то данный пункт можно пропустить.
4. Перейдем в каталог, где находится файл с исходным текстом нашей программы. Если вы не используете файловый менеджер, то для этого придется выполнить команду смены текущего каталога `cd`.

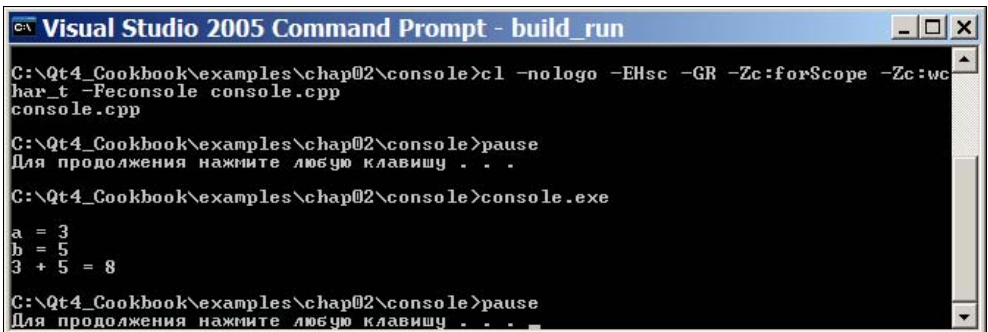


5. Выполним компиляцию и сборку программы, для чего введем команду

```
cl -nologo -EHsc -GR -Zc:forScope -Zc:wchar_t -Feconsole1
  ↵ console1.cpp
```

Символ ↵ в самой команде не вводится, тут он обозначает перенос строки. Здесь `cl` — это имя исполняемого файла компилятора. Последним параметром указано имя файла с исходным текстом программы. Ключ `-Fe` задает имя файла-результата сборки программы. Ключ `-nologo` позволяет избежать вывода на экран лишних сообщений (при первом запуске не указывайте его, чтобы узнать название и номер версии установленного у вас компилятора). Остальные ключи пока не обязательны, т. к. соответствующие возможности пока отсутствуют в нашей простой программе, но в дальнейшем без них не обойтись: ключ `-EHsc` разрешает использование исключений; ключ `-GR` разрешает компилятору учитывать информацию о типах во время выполнения программы (RTTI — RunTime Type Information); ключ `-Zc:forScope` устанавливает новое правило видимости параметров оператора цикла `for`; ключ `-Zc:wchar_t` разрешает применение типа `wchar_t` (символов Unicode). Чтобы рассмотреть сообщения, которые будут выведены на экран в результате выполнения указанной команды, при работе в `Far` нужно нажать комбинацию клавиш `<Ctrl>+<O>`.

6. Если компиляция и компоновка программы прошла без ошибок, то в текущем каталоге будет создан файл `console1.exe`. Запустим его на выполнение. Введем два целых числа и получим результат их сложения (рис. 2.1).



```

C:\Qt4_Cookbook\examples\chap02\console>cl -nologo -EHsc -GR -Zc:forScope -Zc:wchar_t -Feconsole console.cpp
console.cpp

C:\Qt4_Cookbook\examples\chap02\console>pause
Для продолжения нажмите любую клавишу . . .

C:\Qt4_Cookbook\examples\chap02\console>console.exe
a = 3
b = 5
3 + 5 = 8

C:\Qt4_Cookbook\examples\chap02\console>pause
Для продолжения нажмите любую клавишу . . .

```

Рис. 2.1. Сборка и выполнение консольной программы

Еще несколько замечаний.

Поскольку при разработке реальных приложений приходится неоднократно вводить в текст программы изменения и снова выполнять компиляцию, то данный процесс лучше автоматизировать с помощью командного файла. Для этого в любом текстовом редакторе нужно набрать текст, приведенный в листинге 2.2, и сохранить его в файле с расширением `bat` или `cmd`.

**Листинг 2.2. Пример командного файла**

```
cl -nologo -EHsc -GR -Zc:forScope -Zc:wchar_t -Feconsole1
↳ console1.cpp
   pause
   console1.exe
   pause
```

Разумеется, запускать данный файл на выполнение необходимо из командной строки Visual Studio.

Следует иметь в виду, что разные компиляторы требуют указания различных ключей. Если вместо Microsoft Visual C++ у вас установлен, например, Borland C++, то команда для сборки программы должна выглядеть следующим образом:

```
bcc32 -q -econsole1 console1.cpp
```

Здесь ключ `q` подавляет приветствие компилятора, а ключ `e` позволяет задать имя выходного файла.

Реальные проекты обычно содержат несколько файлов с исходными текстами, к тому же часто используют предварительно скомпилированные библиотеки. Поэтому этапы *компиляции* (генерации объектных файлов) и *компоновки* (получения исполняемого файла программы) приходится выполнять по отдельности. Вызов компилятора (без компоновщика) в Microsoft Visual C++, Borland C++ и GNU C++ осуществляется с указанием одинакового ключа `c`. Например, для Microsoft Visual C++ компиляция с последующей компоновкой выполняется так:

```
cl -c -nologo -EHsc -GR -Zc:forScope -Zc:wchar_t -Foconsole1
↳ console1.cpp
link -nologo -out:console1.exe console1.obj
```

## 2.3. Компиляция и выполнение программы с помощью командной строки в системах Linux/FreeBSD/Solaris/HP-UX/Mac OS X

**ЗАДАЧА.** С помощью GNU C++ выполнить компиляцию и сборку консольной программы, текст которой приведен в листинге 2.1, и проверить правильность ее выполнения.

**Решение.**

1. Наберем в любом текстовом редакторе исходный текст программы (см. листинг 2.1). Сохраним его в отдельном каталоге, в файле под именем `console1.cpp`.

2. Для удобства запустим какой-нибудь текстовый файловый менеджер, например, Midnight Commander (команда `mc`).
3. Перейдем в тот каталог, где находится файл с исходным текстом нашей программы. Если не используется файловый менеджер, то для этого придется выполнить команду смены текущего каталога `cd`.
4. Выполним компиляцию и сборку программы, для чего введем команду вызова компилятора и компоновщика GNU C++ (ключ `-o` позволяет задать имя выходного файла):

```
g++ -o console1 console1.cpp
```

### Замечание

Для вызова только компилятора (без компоновщика) GNU C++ следует указать ключ `-c`. Для последующей компоновки объектных файлов придется выполнить команду `g++` еще раз:

```
g++ -c -o console1.o console1.cpp
g++ -o console1 console1.o
```

## 2.4. Создание консольного приложения в Microsoft Visual Studio 2005

**ЗАДАЧА.** Создать группу проектов (*solution* — "решение") и проект консольного приложения Win32, собрать и выполнить программу из листинга 2.1.

### Решение.

1. Откроем интегрированную среду разработки программ Microsoft Visual Studio 2005. Закроем окно Qt Visual Studio Integration, если оно появилось на экране.
2. Создадим группу проектов (*solution*), для этого выберем из главного меню пункты **File** (Файл) | **New** (Новый) | **Project** (Проект). Слева, в разделе **Project Types** (Типы проектов) раскроем группу **Other Project Types** (Другие типы проектов) и щелчком мыши укажем пункт **Visual Studio Solutions**, затем справа, в окне **Templates** (шаблоны) щелкнем левой кнопкой мыши по пункту **Blank Solution** (Пустое решение). Внизу в поле **Name** (Имя) введем название, например, `Qt4_Cookbook_Examples`, а в графе **Location** (Расположение) укажем полный путь к каталогу, в котором будут располагаться наши проекты (рис. 2.2). Нажмем кнопку **OK**.
3. Теперь создадим новый консольный проект: **File** (Файл) | **New** (Новый) | **Project** (Проект). В разделе **Project types** (Типы проектов) раскроем груп-

пу **Visual C++** и щелчком мыши укажем пункт **Win32**. В окне **Templates** (Шаблоны) выберем пункт **Win32 Console Application** (Консольное приложение Win32). В поле **Name** введем название проекта (и одновременно имя каталога, в котором будут располагаться все файлы проекта), например, `console1`. В поле **Location** (Расположение) укажем полный путь к каталогу, в котором будет создан новый каталог с указанным именем. В графе **Solution** (Группа проектов) выберем из раскрывающегося списка режим **Add to Solution** (Добавить к имеющейся группе проектов, рис. 2.3). Нажмем кнопку **OK**.

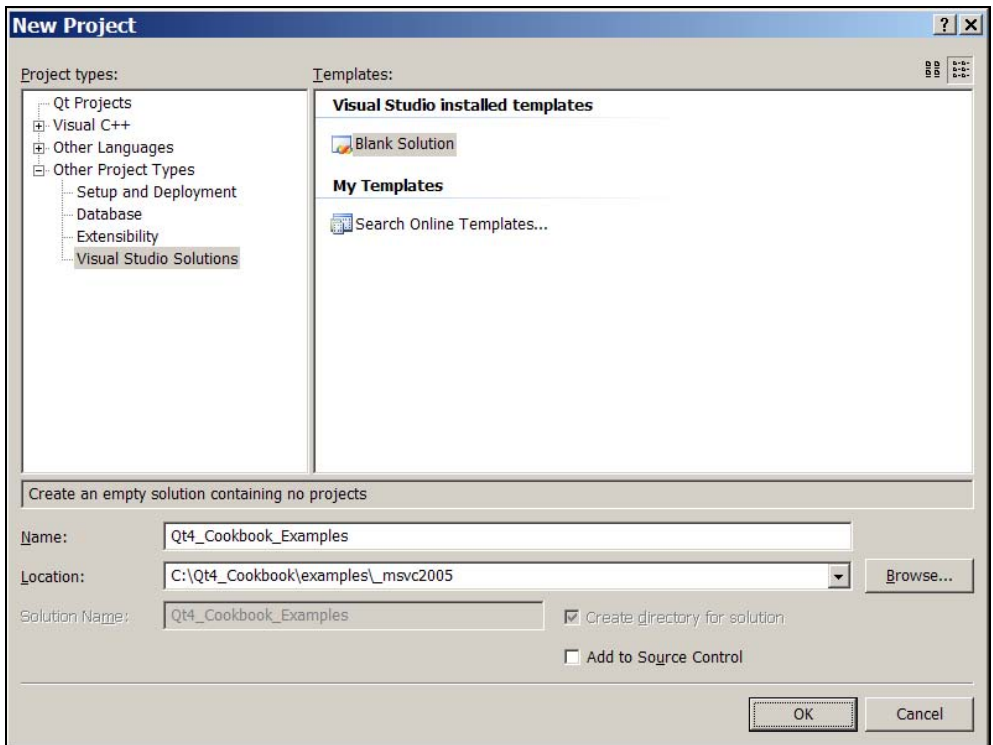


Рис. 2.2. Создание группы проектов в Microsoft Visual Studio 2005

4. На экране появится диалог **Win32 Application Wizard** (Мастер приложения Win32). Перейдем к разделу **Application Settings** (Настройки приложения), выбрав пункт с соответствующим названием слева или просто нажав кнопку **Next** (Далее).
5. Убедимся, что в разделе **Application type** (Тип приложения) выбрана опция **Console application** (Консольное приложение). В разделе **Additional options** (Дополнительные параметры) поставим галочку напротив пункта

**Empty project** (Пустой проект), т. к. мы не хотим, чтобы мастер добавлял к проекту образец кода (рис. 2.4). Нажмем кнопку **Finish** (Конец).

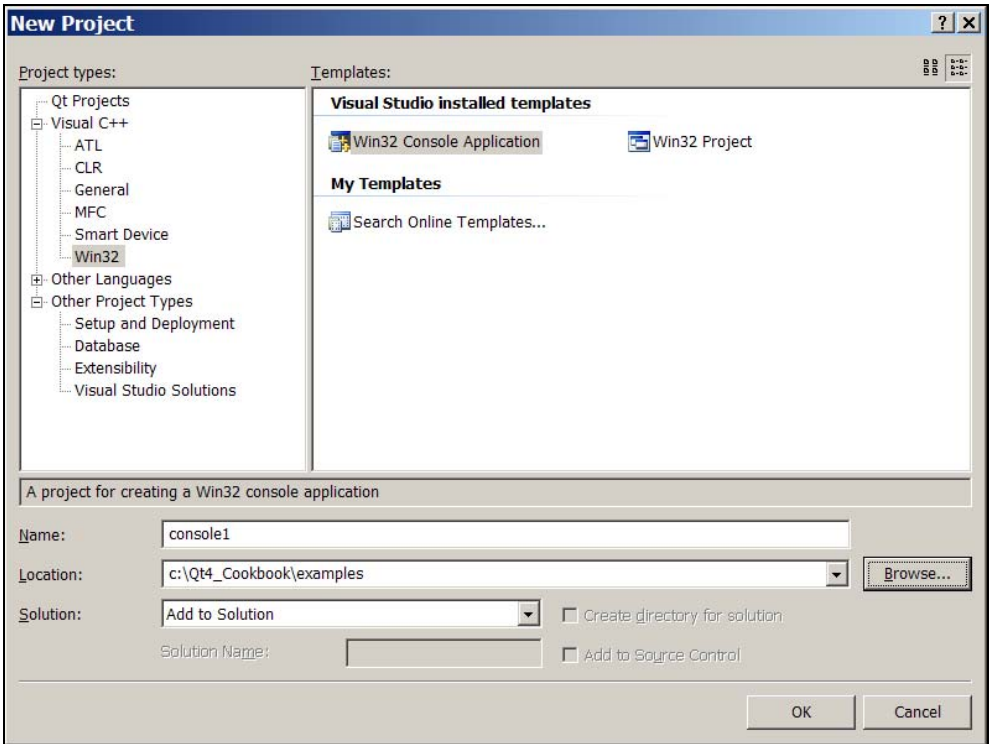


Рис. 2.3. Создание проекта консольного приложения в Microsoft Visual Studio 2005

6. Создадим новый файл для хранения исходного текста нашей программы и сразу добавим его к проекту: **Project** (Проект) | **Add New Item** (Добавить новый элемент). В разделе **Categories** (виды файлов) слева выберем группу **Visual C++**, а справа, в разделе **Templates** (Шаблоны)— пункт **C++ File (.cpp)**. В поле **Name** (Имя) введем название нового файла (т. к. в проекте предполагается всего один файл с исходным текстом, то его лучше назвать так же, как и сам проект), а в поле **Location** (Расположение) укажем каталог, в котором будет сохранен этот файл. Нажмем кнопку **Add** (Добавить).
7. Теперь можно ввести текст программы (см. листинг 2.1).
8. Нажмем кнопку **Save** (Сохранить) на панели инструментов, или воспользуемся главным меню: **File** (Файл) | **Save** (Сохранить), или комбинацией клавиш <Ctrl>+<S>.

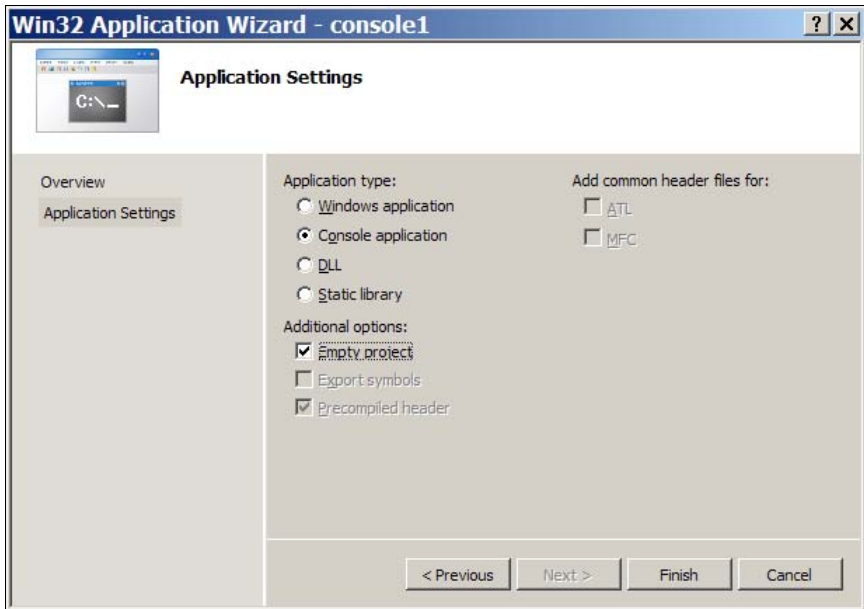


Рис. 2.4. Установка параметров нового проекта

9. Зададим режим компиляции: **Build** (Собрать) | **Configuration Manager** (Менеджер конфигураций). В раскрывающемся списке **Active Solution Configuration** (Текущая конфигурация) выберем пункт **Release**. Это значит, что результат компиляции не будет содержать информацию для отладчика.
10. Выполним компиляцию и сборку программы: **Build** (Собрать) | **Build Solution** (Собрать все проекты) или просто нажмем клавишу <F7>. Если не будет выведено каких-либо сообщений об ошибках, то мы увидим строку:
 

```
==== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped =====
```

 Это означает, что один проект собран без ошибок (*succeeded* — выполненный успешно).
11. Теперь можно запустить нашу программу на выполнение: **Debug** (Отладка) | **Start Without Debugging** (Выполнить без отладки). На экране появится консольное окно, в котором нас попросят ввести два числа, а затем будет выведен результат их сложения (рис. 2.5). После окончания выполнения программы на экран автоматически выводится приглашение нажать любую клавишу (оно появляется только в том случае, если программа была запущена из интегрированной среды) — иначе бы окно сразу закрылось и мы не успели рассмотреть результат.