ГАЛИНА ДОВБУШ
АНАТОЛИЙ ХОМОНЕНКО

Visual C++

НА ПРИМЕРАХ

СОЗДАНИЕ ПРИЛОЖЕНИЙ В СРЕДЕ VISUAL C++

ОСНОВЫ ПРОГРАММИРОВАНИЯ НА C++

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

ОРГАНИЗАЦИЯ ВВОДА-ВЫВОДА И ОБРАБОТКА ИСКЛЮЧЕНИЙ

СОЗДАНИЕ ПРИЛОЖЕНИЙ API WINDOWS И MFC

+CD

bhv®

Галина Довбуш

Анатолий Хомоненко

Visual C++

НА ПРИМЕРАХ

Под редакцией профессора Хомоненко А. Д.

Санкт-Петербург

«БХВ-Петербург»

2007

УДК 681.3.068+800.92VisualC++
ББК 32.973.26-018.1
Д58

Довбуш, Г. Ф.

Д58 Visual C++ на примерах / Г. Ф. Довбуш, А. Д. Хомоненко / Под ред. проф. А. Д. Хомоненко. — СПб.: БХВ-Петербург, 2007. — 528 с.: ил.

ISBN 978-5-94157-918-1

Рассмотрены интерфейс системы программирования Visual C++, техника создания и отладки проектов приложений в среде Visual Studio 2005. Описаны основы языка C++: типы данных и операции, приемы программирования разветвлений и циклов, техника работы со статическими и динамическими массивами, использование функций. Рассмотрены классы и объекты, механизм множественного и одиночного наследования, перегрузка операторов и шаблоны классов, понятия ввода-вывода данных и классификация, принципы работы с потоками и файлами, стандартные классы потоков, форматированный ввод-вывод базовых типов, дополнительные возможности ввода-вывода. Освещена обработка исключений. Показаны особенности создания приложений API Windows и MFC. Представлены внутренняя их организация, создание диалоговых окон и меню, механизм обработки сообщений, работа с картой сообщений. Приводятся многочисленные примеры отлаженных программ. На компакт-диске содержатся тексты листингов примеров программ, приведенных в книге.

Для начинающих программистов

УДК 681.3.068+800.92VisualC++
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Григорий Добин*
Редактор	*Елена Кашлакова*
Компьютерная верстка	*Натальи Караваевой*
Корректор	*Виктория Пиотровская*
Дизайн серии	*Игоря Цырульникова*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 28.09.07.
Формат 70×100$^{1}/_{16}$. Печать офсетная. Усл. печ. л. 42,57.
Тираж 3000 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 978-5-94157-918-1

© Довбуш Г. Ф., Хомоненко А . Д., 2007
© Оформление, издательство "БХВ-Петербург", 2007

Оглавление

Предисловие

Система программирования Microsoft Visual C++ (MVC++) входит в состав Visual Studio и является популярным инструментом, широко используемым для разработки различного рода приложений, выполняемых под управлением Windows, и обеспечивающим высокую эффективность их кода. Это обусловлено (в частности) тем, что MVC++ разрабатывалась фирмой Microsoft — создателем семейства операционных систем Windows.

По языку C++ и системе программирования MVC++ имеется большое множество учебной литературы. Предлагаемая книга также ориентирована на использование в учебном процессе. При ее подготовке преследовалась цель последовательно и систематично изложить приемы программирования в среде MVC++, приводя сравнительно небольшой объем теоретических сведений и продуманный набор содержательных примеров. Все приводимые примеры отлажены в среде MVC++ из состава Visual Studio 2005.

В системе программирования MVC++ можно создавать достаточно большое число различных видов приложений. Мы рассматриваем три основные разновидности приложений: консольные приложения, приложения API Windows и приложения MFC.

Консольные приложения в книге используются для рассмотрения основных приемов программирования на языке C++ при создании приложений, в которых обмен информацией ведется в основном с помощью клавиатуры и экрана.

Приложения API Windows позволяют обеспечить развитый графический интерфейс, основанный на использовании соответствующих функций и диалоговых окон.

Приложения MFC также позволяют обеспечить развитый графический интерфейс, основанный на использовании диалоговых окон. Вместо функций API здесь используются компоненты библиотеки MFC, при этом повышается удобство разработки приложений.

Книга состоит из шести частей.

В *части I* описываются среда программирования MVC++, вопросы создания, открытия и активизации проекта приложения в новой или существующей рабочей области. Освещаются этапы подготовки программы к исполнению: трансляция, компоновка и отладка. Рассматривается технология создания консольного приложения, описывается функция `main ()`, и затрагиваются вопросы вывода текста на экран.

В *части II* описываются простые типы данных, ввод данных с клавиатуры и вывод данных на экран, операции над операндами простых типов. Рассматриваются операторы: составные, условные, цикла и передачи управления. Описываются массивы (одномерные, многомерные, символьные) и операции над ними, операции с указателями, использование указателей при работе с обычными и динамическими массивами, действия с массивами указателей. Освещаются структуры: операции доступа к элементам структуры, инициализация структур и действия с массивами структур. Рассматриваются функции и их использование: прототип, определение, возвращаемое значение, вызов, область видимости, включение функции в проект приложения. Описываются способы передачи параметров функции, в том числе массивов, функции обработки символов и строк, перегрузка функций и шаблонные функции.

Часть III рассматривает объекты и классы: спецификаторы доступа к объектам класса, объявление класса и объекта класса, доступ к членам объектов, конструкторы и деструктор. Описан указатель `this` на объект класса, данные и методы класса, а также стандартный класс `string` для строкового типа данных. Освещаются композиция, одиночное и множественное наследование, чистые виртуальные функции и абстрактные классы. Описывается перегрузка операторов с помощью операторных функций-членов класса и функций-друзей класса, а также перегрузка операторов в производных классах. Рассматриваются шаблоны классов: объявление шаблона класса и объектов шаблона класса, параметры по умолчанию, наследование при работе с шаблонами.

В *части IV* рассматриваются основы ввода-вывода: классификация способов, принципы работы с потоками и файлами, стандартные классы потоков, форматированный ввод-вывод базовых типов. Освещаются дополнительные возможности ввода-вывода: форматированный ввод-вывод пользовательских типов, файловый ввод-вывод на верхнем уровне, неформатированный ввод-вывод, использование библиотеки `stdio`. Описывается технология обработки исключительных ситуаций.

В части V дается характеристика приложений API Windows: графический интерфейс приложений, контекст устройства, состав приложения, функ-

ция `WinMain()`, оконная процедура обработки сообщений, шаги создания приложения. Описывается технология разработки интерфейса приложения API Windows: создание меню и диалогового окна, использование элементов управления, задание оконных процедур.

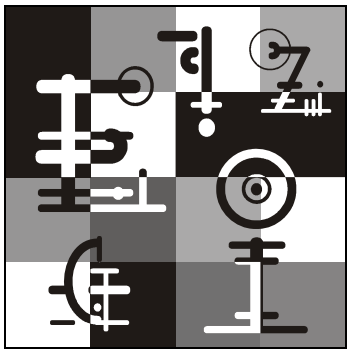
В части VI дается характеристика приложений MFC: библиотека MFC, графический интерфейс. Рассматриваются этапы создания, типы и состав приложений MFC. Описывается организация обработки сообщений: карта и макросы обработки сообщений, типы передаваемых сообщений. Рассматривается технология разработки интерфейса приложения MFC: общая характеристика интерфейса приложения, создание диалогового окна и класса окна, доступ к элементам управления окна. Освещается ввод-вывод с помощью класса `CFile`.

Прилагаемый компакт-диск содержит тексты листингов для примеров программ, приводимых в книге.

Подготовка книги основана на опыте преподавания ряда дисциплин, связанных с изучением приемов и технологий программирования на языке C++ в среде Visual Studio, преподавателями кафедры информационных и вычислительных систем Санкт-Петербургского государственного университета путей сообщений и кафедры математического обеспечения военно-космической академии имени А. Ф. Можайского.

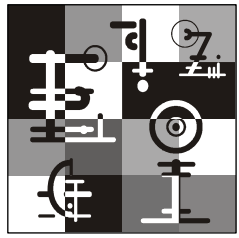
Выражаем признательность Бобровскому А. И., участвовавшему в подготовке материалов четвертой части книги. Благодарим также Сухоногова А. М. и Темплинга А. А. за подготовку материалов с примером консольного приложения (в приложении к книге).

Авторы



Часть I

Простейшая программа на языке C++



Глава 1

Подготовка программы к исполнению

Подготовка прикладных программ к исполнению на компьютере включает в себя три этапа.

Первый из них заключается в создании и редактировании файлов исходной программы, написанной на языке программирования. Как правило, программа на языке C++ содержит множество файлов, которые делятся на файлы спецификации и файлы реализации. Файл спецификации, или заголовочный файл для C++ имеет расширение `.h` и содержит описание используемых в программе типов данных, а также прототипов функций. Файл спецификации включается при помощи специальной директивы препроцессора `include` в соответствующий текст файла реализации, который содержит инструкции языка для выполнения тех или иных действий. Файл реализации имеет расширение `.cpp`.

На втором этапе из исходных файлов формируются объектные программы, то есть программы в машинных кодах, полученные после компиляции исходных файлов. Объектные файлы имеют расширение `.obj`. Каждый исходный файл реализации обрабатывается отдельно. В процессе компиляции в исходной программе могут быть обнаружены синтаксические ошибки, которые следует исправить, иначе объектная программа не будет построена. Если компиляция не выявила ошибок, то можно переходить к следующему этапу.

Третий этап состоит в построении исполняемого файла, имеющего расширение `.exe`. На этом этапе все объектные программы и функции из системной библиотеки объединяются в единое целое, и происходит компоновка программы. В процессе компоновки так же, как и в процессе компиляции, могут возникнуть ошибки. В случае появления таких ошибок их тоже следует исправить, после чего повторить компиляцию исправленных исходных файлов и выполнить компоновку.

После третьего этапа программа готова к исполнению. Сначала необходимо проверить работоспособность программы на заранее подготовленных контрольных (тестовых) примерах. Если в результате проверки будет получен хотя бы один несхожий с ожиданиями результат, следует найти место ошибки исполнения в исходной программе, а затем повторить заново все этапы.

Схема процесса подготовки программы к исполнению показан на рис. 1.1.

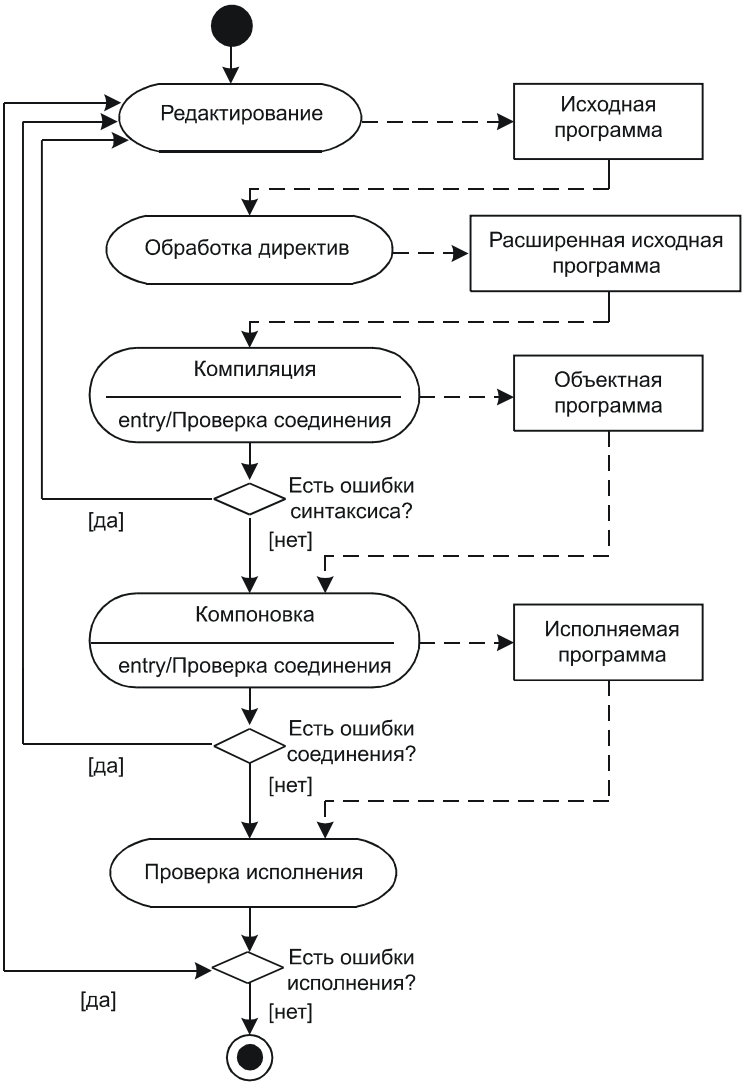
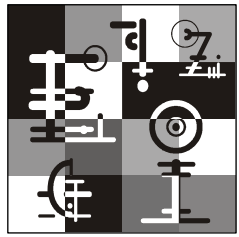


Рис. 1.1. Схема процесса подготовки программы к исполнению



Глава 2

Среда программирования

Среда программирования (program environment) образуется необходимыми для создания программ средствами, к которым относятся:

- редактор;
- препроцессор;
- компилятор;
- компоновщик;
- отладчик.

Редактор (editor) — программа, используемая для написания и изменения исходных программ или данных.

Основные функции редактора среды программирования MVC++:

- ввод и сохранение на диске файлов с текстами исходных программ;
- выделение фрагментов текста, их копирование и перенос;
- синтаксическая раскраска текста программы. По умолчанию редактор MVC++ раскрашивает текст программы в черный цвет с комментариями зеленого цвета и ключевыми словами (служебными словами языка C++) голубого цвета. Синтаксическая раскраска помогает замечать ошибки, допущенные при наборе текстов программ;
- форматирование текста программы. Редактор автоматически расставляет отступы при наборе операторов. Эта функция редактора, так же как и предыдущая, позволяет избежать многих ошибок.

Препроцессор (preprocessor) — программа, которая выполняет предварительную обработку исходной программы для компилятора. Перед тем как попасть на вход компилятора, исходная программа проходит через препроцессор,

работа которого регламентируется директивами (preprocessor directive) и операторами (preprocessor operator).

Компилятор (compiler) — средство для перевода (трансляции) исходной программы, написанной на языке программирования, в совокупность понятных компьютеру машинных команд. Компилятор может обнаружить ошибки трансляции (compile errors), а значит, он не сможет создать объектную программу. Как правило, это синтаксические ошибки (syntax errors). Компилятор также может сообщать предупреждения (warnings), которые необходимо обработать как ошибки. Все ошибки в исходной программе следует исправить в редакторе, после чего повторить ее трансляцию. Для решения одной задачи создается несколько исходных программ, каждая из которых проходит трансляцию. Исходные программы используют различные функции, размещенные в специальных библиотечных файлах (например, в библиотеке стандартного потока ввода/вывода). Компилятором из исходных программ генерируются объектные, требующие соединения для решения поставленной задачи. Точно так же в соединении с объектными программами нуждаются и используемые библиотечные функции.

Компоновщик (linker) — средство связывания объектных программ с кодами функций из стандартных библиотек, обеспечивающее сборку (компоновку) исполняемой программы. Во время связывания объектных файлов могут возникать ошибки компоновки (linker errors). Эти ошибки показывают, что указанные в программе внешние ссылки не могут быть найдены компоновщиком. В таком случае следует уточнить ссылки в исходных программах, а затем повторить трансляцию и компоновку снова.

Результаты работы исполняемой программы проверяются на тестовых примерах. Тестовые примеры (test cases) — контрольные наборы тестовых данных для проверки правильности функционирования исполняемой программы. Тестовые примеры подготавливаются заранее. Для каждого примера записываются контрольные значения необходимых исходных данных, а также записываются ожидаемые выходные данные (результаты работы исполняемой программы). Тестовые примеры должны учитывать следующие возможные варианты для входных данных:

- значения данных принадлежат допустимому диапазону изменения;
- значения данных находятся на границе диапазона изменения;
- значения данных запрещены, то есть находятся вне разрешенного диапазона изменения.

Во время проверки исполняемой программы, могут возникать ошибки исполнения (run-time errors). Если программа делает что-то запрещенное, она

тут же завершается. Однако ошибка не всегда приводит к остановке программы, и только часть результатов может оказаться неверной. Такая форма ошибки исполнения является логической ошибкой (logical error). Как правило, появление логических ошибок происходит вследствие неправильного понимания задачи или допущенных на этапе разработки программы просчетов. Если хотя бы один тестовый пример будет выполнен некорректно, то следует найти ошибку, исправить исходную программу, откомпилировать ее, создать новую исполняемую программу и повторить тестовые примеры.

Для поиска ошибок исполнения проводится отладка, являющаяся существенной частью процесса программирования. Отладка (debugging) позволяет обнаружить, локализовать и устранить ошибки в программе. Ключевым понятием при отладке программ является точка останова.

Точка останова (breakpoint) — это место в исходной программе, где следует остановиться в процессе выполнения программы. В точке останова происходит прекращение исполнения программы перед выполнением отмеченного точкой останова оператора, после чего можно завершить выполнение программы или работать дальше, анализируя передачу управления в программе и изменение значений переменных.

Для устранения ошибок исполнения необходимо проверить алгоритм решения задачи и воспользоваться пошаговой отладкой программы с помощью отладчика.

Отладчик (debugger) — средство, предназначенное для анализа поведения исполняемой программы, обеспечивающее ее пошаговое выполнение (трассировку).

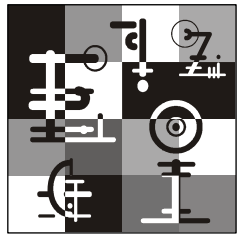
Основные функции отладчика среды программирования MVC++:

- установка точек останова для прерывания выполнения программы в указанных местах с целью последующей трассировки. Установленная точка останова отмечается красной точкой на левой границе окна в строке текста программы, содержащей оператор. После остановки исполнения программы можно добавить или убрать точки останова;
- трассировка с самого начала или с любого места программы. Желтая стрелка на левой границе текста программы показывает оператор, который будет выполняться. При трассировке программы желтая стрелка движется, чтобы показать, какой оператор будет выполняться;
- пошаговое выполнение программы с трассировкой вызываемых ею функций или без их трассировки;
- просмотр значений переменных. В специальном окне отладчика отображаются значения используемых программой в данный момент времени

переменных. Можно проанализировать переменные и продолжить или прервать выполнение;

- изменение значений переменных в процессе отладки. В специальном окне отладчика можно установить новые (другие) значения для переменных, чтобы понять, как изменится процесс выполнения программы с измененными значениями.

Проверенная и отлаженная исполняемая программа может выполняться на компьютере до тех пор, пока она существует. После запуска исполняемой программы для решения задачи загрузчик размещает программу в оперативной памяти. Центральный процессор выбирает и выполняет каждую инструкцию программы, сохраняя новые значения данных по мере выполнения программы.



Глава 3

Создание консольного приложения

Консольное приложение — это программа, которая выполняется из командной строки окна DOS или Windows и не имеет графического интерфейса. Проект консольного приложения создается пустым и предполагает добавление в него исходных файлов вручную.

Среда программирования MVC++ использует концепцию рабочей области (solution), что на один уровень абстракции выше, чем проект (project). Проект — это множество всех файлов, необходимых для построения исполняемого файла. В одной рабочей области может содержаться несколько проектов. Несмотря на наличие в рабочей области нескольких проектов, работать можно только над одним, называемым активным. Для рабочей области создается отдельная папка, включающая каталог \Debug и несколько файлов, основным из которых является конфигурационный файл области с расширением `.sln`. Внутри рабочей области каждый проект имеет собственный каталог, в котором находятся конфигурационный файл проекта с расширением `.vcproj` и исходные файлы проекта с расширениями `.h` (заголовочные файлы) и `.cpp` (файлы реализации). Во время создания проекта в его каталоге создается пустая папка \Debug, где будут храниться объектные файлы `.obj`. Исполняемые файлы `.exe` всех проектов рабочей области записываются в каталог рабочей области.

Запуск MVC++

Для запуска среды создания приложений MVC++ необходимо последовательно выполнить следующие действия:

1. Нажать кнопку **Пуск** (Start).
2. Выбрать пункт меню **Программы** (Programs).

3. Выбрать пункт меню **Visual Studio 2005**. В результате откроется стартовое окно, которое показано на рис. 3.1.

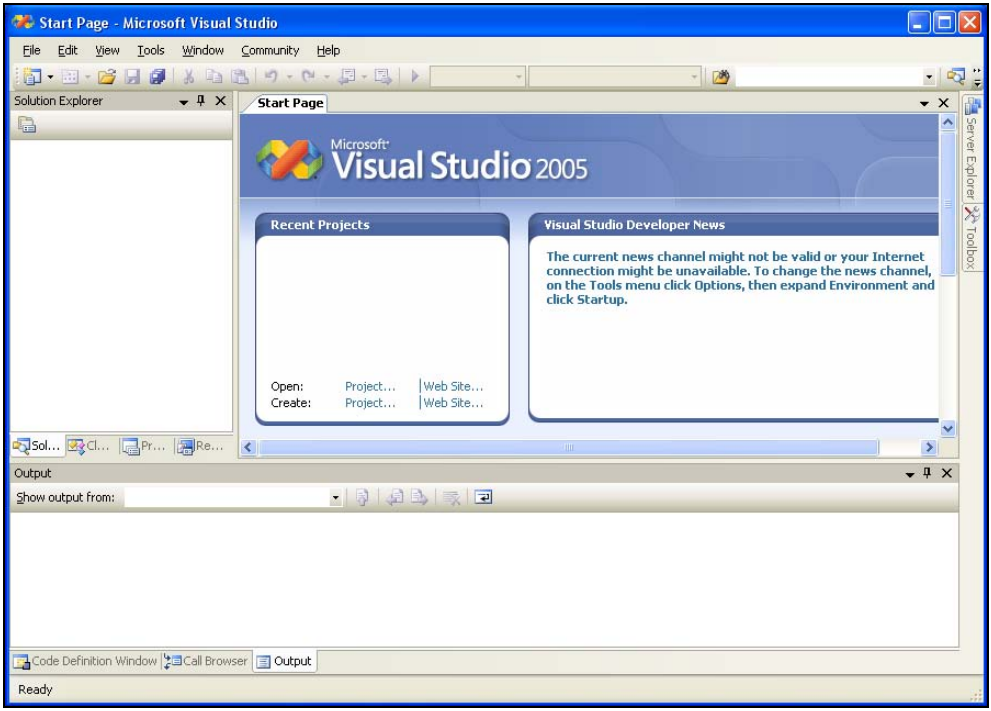


Рис. 3.1. Стартовое окно MVC++

В верхней части окна расположено меню, с помощью которого создаются, изменяются и выполняются программы C++. После выбора пункта меню появляется выпадающее меню, в котором следует выбрать соответствующий пункт. Состояние меню и окон меняется в зависимости от выполняемых программистом действий.

Создание проекта в новой рабочей области

Как упоминалось, рабочая область (solution) создается с целью объединения схожих по тематике проектов. Для создания проекта консольного приложения в новой рабочей области необходимо:

1. Выбрать пункт меню **File → New → Project**.

2. В открывшемся окне **New Project**:
   - в строке **Location** ввести каталог расположения рабочей области или выбрать этот каталог в окне **Project Location**, нажав кнопку **Browse**;
   - в нижней части окна в строке **Solution Name** вместо **<Enter_Name>** набрать название рабочей области;
   - убедиться, что установлен флажок **Create directory for solution**;
   - в строке **Name** вместо **<Enter_Name>** ввести имя проекта;
   - в левой части окна **Project types** выбрать тип проекта **Visual C++ General**;
   - в правой части окна **Templates** выбрать шаблон проекта **EmptyProject**;
   - нажать кнопку **OK**.

На рис. 3.2 демонстрируется вид окна **New Project** при создании проекта p1 в новой рабочей области Exercise, которая будет расположена на диске E.

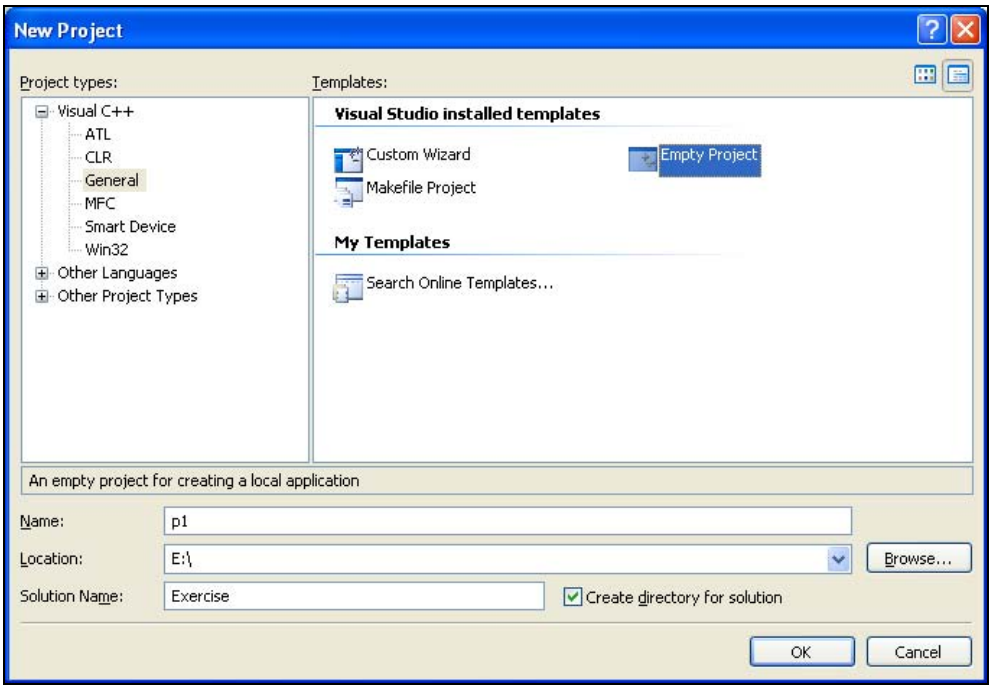


Рис. 3.2. Создание проекта в новой рабочей области

В результате работы мастера MVC++ на диске E в корневом каталоге будет создана папка \Exercise, внутри которой будет сформирован файл рабочей

области Exercise.sln, а также папка \Debug для хранения исполняемых файлов рабочей области. В браузере активной вкладки **Solution Explorer** правой части экрана будет выведено состояние рабочей области, как показано на рис. 3.3.

Рис. 3.3. Состояние рабочей области после создания проекта

Открытие существующей рабочей области

Открыть существующую рабочую область можно различными способами, например, используя MVC++ или проводник.

Для открытия области с помощью MVC++ необходимо выполнить следующие действия:

1. Открыть MVC++.
2. Выбрать нужную область из списка **Recent Project** в стартовом окне.

Если в списке стартового окна нет названия необходимой рабочей области:

1. Выбрать меню **File → Open Project/Solution**.

2. В окне **Open Project** выделить файл рабочей области и открыть его, нажав кнопку **Open**.

Примечание

Если во вкладке **Solution Explorer** браузера MVC++ уже имеется открытая рабочая область, то проекты вновь открываемой рабочей области могут быть добавлены в состав существующей. Для этого следует в окне **Open Project** установить переключатель **Add to Solution** и нажать кнопку **Open**. В результате проекты будут добавлены в уже открытую рабочую область. Если в добавлении проектов нет необходимости, должен быть установлен переключатель **Close Solution**, тогда открытая рабочая область закроется, и будет открыта выбранная в окне **Open Project**.

Для открытия рабочей области при помощи проводника Windows следует выполнить такие шаги:

1. Открыть Проводник.
2. Выделить файл рабочей области (например, **Exercise.sln**).
3. Дважды щелкнуть левой кнопкой мыши по названию файла.

Создание нового проекта в рабочей области

Для создания нового проекта консольного приложения внутри рабочей области проще всего выполнить следующую последовательность действий:

1. Нажать правую кнопку мыши на имени рабочей области в браузере активной вкладки **Solution Explorer**.
2. В появившемся меню выбрать пункт **Add**.
3. В появившемся меню выбрать пункт **New Project**.
4. В окне Add New Project:
   - в левой части окна **Project types** выбрать тип проекта **General**;
   - в правой части окна **Templates** выбрать шаблон проекта **Empty Project**;
   - в нижней части окна в строке **Name** вместо **<Enter_name>** набрать без пробелов название проекта;
   - нажать кнопку **OK**.

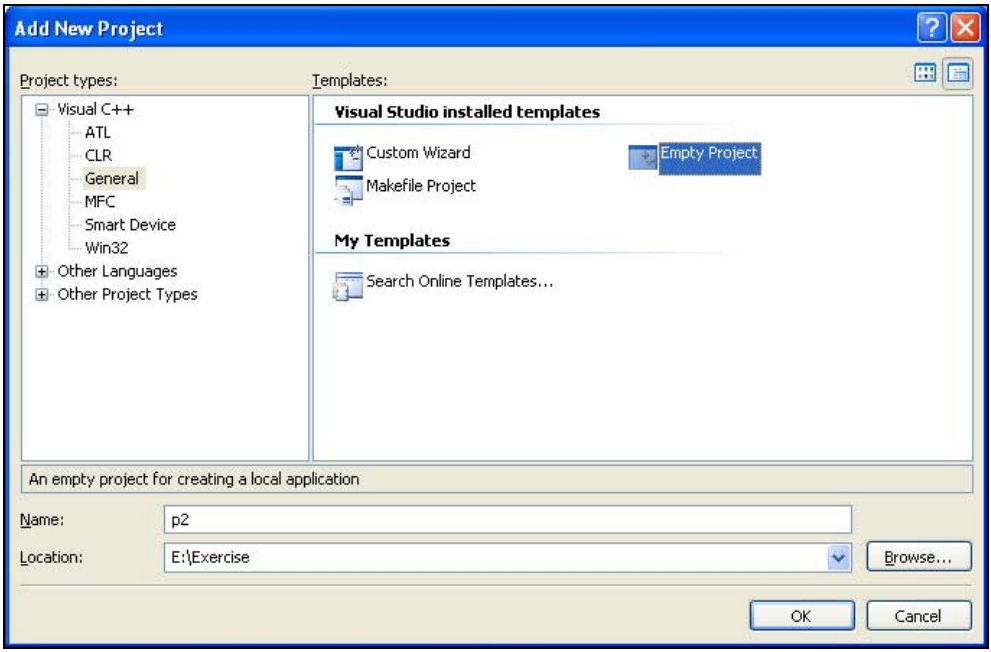


Рис. 3.4. Создание нового проекта в открытой рабочей области

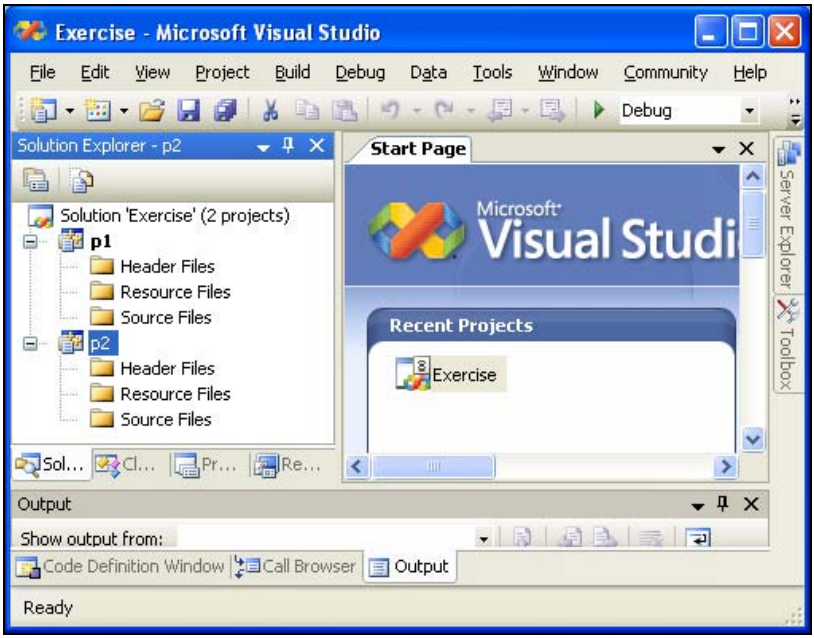


Рис. 3.5. Состояние браузера рабочей области с проектами

На рис. 3.4 показано окно при создании нового проекта p2 в области Exercise.

В результате на диске E в каталоге \Exercise будет создана папка p2, внутри которой будет сформирован файл проекта консольного приложения p2.vcproj. Состояние браузера в активной вкладке **Solution Explorer** показано на рис. 3.5.

Активизация существующего проекта

Рабочая область, как правило, содержит множество проектов. Чтобы сделать проект активным, следует во вкладке **Solution Explorer** выделить имя нужного проекта, нажать правую кнопку мыши и выбрать в меню пункт **Set as StartUp Project**. Название активного проекта в браузере выделяется при этом полужирным шрифтом.

Добавление исходных файлов в проект

В проект консольного приложения необходимо вручную включить исходные файлы: заголовочные файлы (Header Files) и файлы реализации (Source Files).

Включение обоих файлов производится одинаково. Разница состоит только в выборе шаблона: для заголовочного файла шаблон называется Header File (.h), а для файла реализации — C++ File (.cpp).

Далее указаны действия, необходимые для подключения файла реализации:

1. Нажать правую кнопку мыши на имени проекта в браузере Solution Explorer.
2. В появившемся меню выбрать пункт **Add**.
3. В появившемся меню выбрать пункт **Add New Item**.
4. В окне Add New Item:
   - выбрать в правой части окна **Categories** категорию Code;
   - выбрать в левой части окна **Templates** шаблон **C++ File (.cpp)**;
   - ввести в окне **Name** вместо **<Enter_name>** имя файла;
   - нажать кнопку **Add**.

На рис. 3.6 приводится окно для включения файла с именем main в проекте p1 рабочей области Exercise.

В результате на диске в каталоге проекта будет создан файл main.cpp, и в редакторе MVC++ появится окно и вкладка для работы с этим файлом. Состояние главного окна MVC++ после включения файла реализации показано на рис. 3.7.

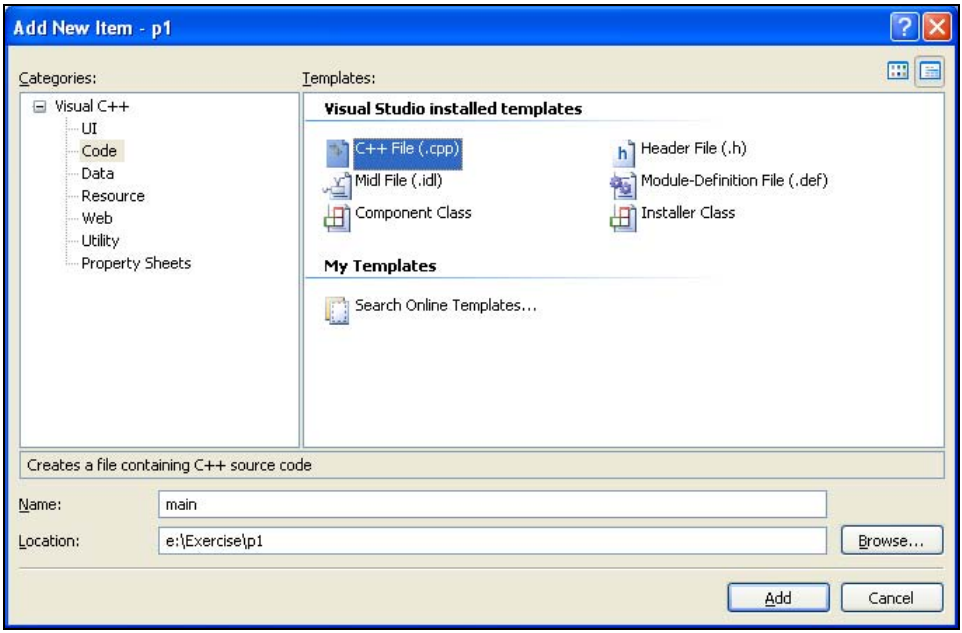


Рис. 3.6. Добавление файла реализации в проект

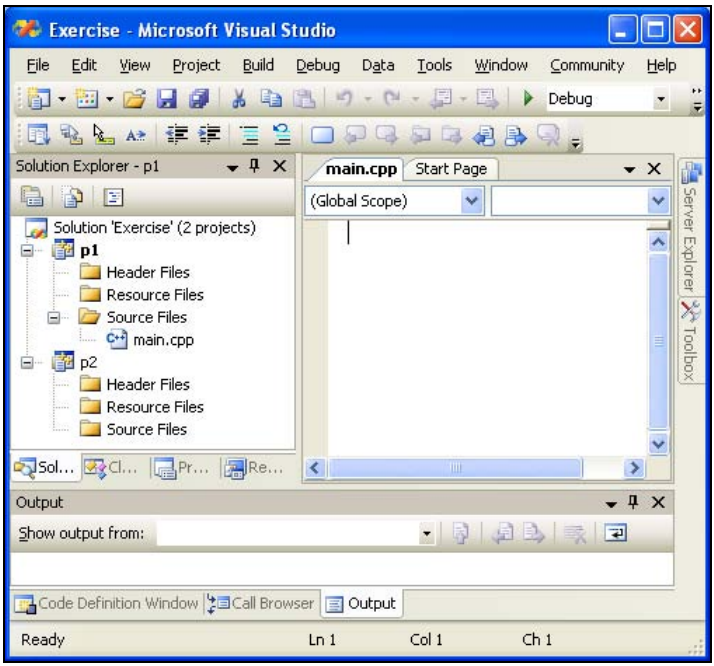


Рис. 3.7. Состояние главного окна после подключения файла реализации

Примечание

Имена исходных файлов задаются автором программы. В начале книги рассматриваются простейшие проекты, состоящие только из одного файла, который содержит исходный код главной функции `main ( )`. Для проектов, состоящих из нескольких файлов, рекомендуется имя заголовочного файла и файла реализации главной функции выбирать таким же, как и имя проекта, а для файлов с описанием и реализацией используемых в программе классов использовать имена самих классов, но без первого символа `C`, с которого обычно они начинаются.

Ввод и редактирование исходного текста осуществляется в активном окне редактора. Редактор поддерживает команды **Cut**, **Copy** и **Paste**, **Undo** и **Redo**, расположенные в меню **Edit**, где также показаны горячие клавиши и пиктограммы для этих действий.

В листинге 3.1 приведен текст первой программы, которая просто выводит текст на экран. Из этой программы будет создаваться исполняемый файл.

Листинг 3.1. Первая программа

```
#include <iostream>
using namespace std ;
int main ( )
{
       cout << "I will be super programmer!" << endl ;
return 0 ;
}
```

Активизация исходного файла для редактирования

Если в окне редактора существует вкладка с именем файла, требующего редактирования, то достаточно щелкнуть левой кнопкой мыши по вкладке. Редактор сделает окно редактирования этого файла активным.

Если такой вкладки не существует, то необходимо дважды щелкнуть левой кнопкой мыши по имени файла в окне **Solution Explorer**. Файл откроется и загрузится в активное окно редактора, в котором появится вкладка с именем открытого файла.

Если файла не существует в проекте, то следует его добавить в проект так, как описано в предыдущем разделе, и приступить к его редактированию.