Б. Пахомов

bhv

C/C++ и MS Visual C++ 2008

для начинающих

Основные элементы языков C/C++

Визуальная среда программирования

Создание основных типов приложений

Работа с наборами данных

+ DVD Дистрибутив Microsoft Visual Studio 2008 Express Edition

Microsoft Visual Studio Express Editions

УДК 681.3.068+800.92C/C++
ББК 32.973.26-018.1
П22

Пахомов Б. И.

П22 C/C++ и MS Visual C++ 2008 для начинающих. — СПб.: БХВ-Петербург, 2008. — 624 с.: ил. + Дистрибутив (на DVD)

ISBN 978-5-9775-0267-2

Книга является руководством для начинающих по разработке приложений в среде Microsoft Visual C++ 2008 Express Edition. Рассмотрены основные элементы языков программирования C/C++ и примеры создания простейших классов и программ. Изложены принципы визуального проектирования и событийного программирования. На конкретных примерах показаны основные возможности визуальной среды разработки Visual C++ 2008 Express Edition, назначение базовых компонентов и процесс разработки различных типов консольных и Windows-приложений. На компакт-диске размещен дистрибутив пакета Microsoft Visual Studio 2008 Express Edition, содержащий Visual C++ 2008 Express Edition и другие компоненты пакета.

Для начинающих программистов

УДК 681.3.068+800.92C/C++
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Григорий Добин*
Редактор	*Римма Смоляк*
Компьютерная верстка	*Натальи Смирновой*
Корректор	*Зинаида Дмитриева*
Дизайн обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 30.04.08.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 50,31.
Тираж 3000 экз. Заказ №
"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Санитарно-эпидемиологическое заключение на продукцию № 77.99.60.953.Д.002108.02.07
от 28.02.2007 г. выдано Федеральной службой по надзору
в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-9775-0267-2

Оглавление

ГЛАВА 1

Типы данных, простые переменные и основные операторы цикла

Структура рабочего стола среды программирования

Цель этой главы — продемонстрировать начальные элементы программирования на языке C. Однако, чтобы построить программу на этом языке, нам надо воспользоваться средой программирования Visual C++ 2008 (ее английская аббревиатура — IDE: Integrated Development Environment), которая содержит в себе средства создания программы, ее компиляции, отладки и запуска на выполнение. В этой связи рассмотрим кратко структуру этой среды, а точнее ее интерфейс, с нами, пользователями. Интерфейс — это аппарат, который позволяет удобно взаимодействовать пользователю со средой.

После установки на своем компьютере среды Visual C++ 2008 вы можете ее загрузить, воспользовавшись командой главного меню **Пуск|Программы** (после установки продукта, его имя автоматически появится в списке команды **Программы**).

Для удобства дальнейшей работы с установленным программным продуктом следует мышью перетянуть его значок  на линейку быстрого запуска программ, которая находится на рабочем столе операционной системы (обычно ее располагают в нижней части стола). Находящийся на этой линейке любой программный продукт запускается одинарным щелчком мыши на значке соответствующего продукта. Итак, загружаем наш продукт Microsoft Visual C++ 2008 Express Edition (для краткости в дальнейшем станем его называть VC++). На экране появится главное окно — рабочий стол, структуру которого мы и рассмотрим.

Главное окно

Общий вид окна показан на рис. 1.1.

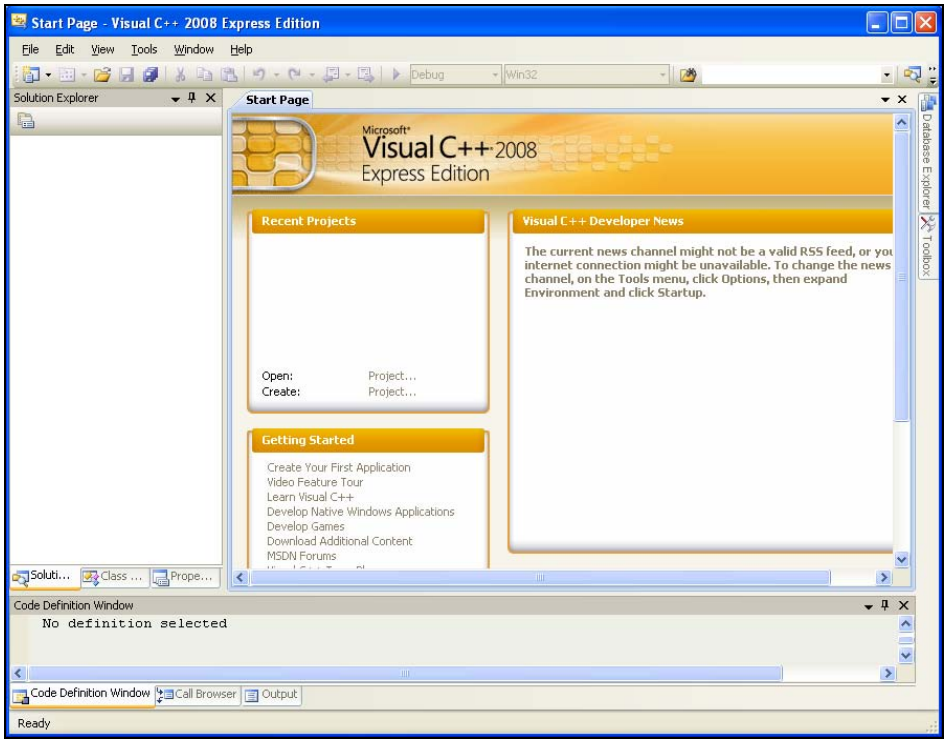

Рис. 1.1. Вид главного окна IDE после загрузки VC++

В верхней части окна расположена строка с *командами главного меню* среды (команды: **File**, **Edit**,…) — это строка горизонтального меню. При вызове этих команд (их еще называют опциями, т. е. элементами выбора из нескольких значений) открываются так называемые "выпадающие меню" — это вертикальные меню, представляющие собой набор команд, располагающихся на экране сверху вниз. Пример такого меню показан на рис. 1.2.

Во второй строке главного окна, расположенной под строкой главного меню, находятся *кнопки быстрого вызова* некоторых команд на исполнение. Все эти кнопки имеют всплывающие подсказки (надо навести курсор мыши на кнопку, немного подождать, после чего появится подсказка о том, для чего предназначена данная кнопка).

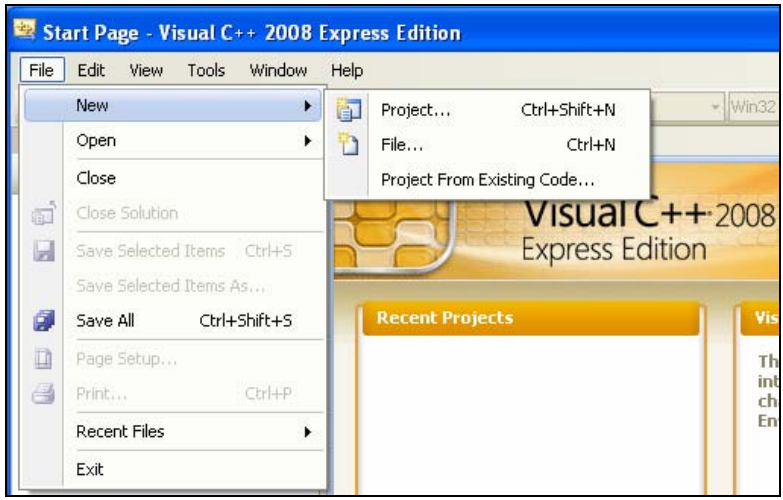

Рис. 1.2. Меню для задания типа формируемого приложения

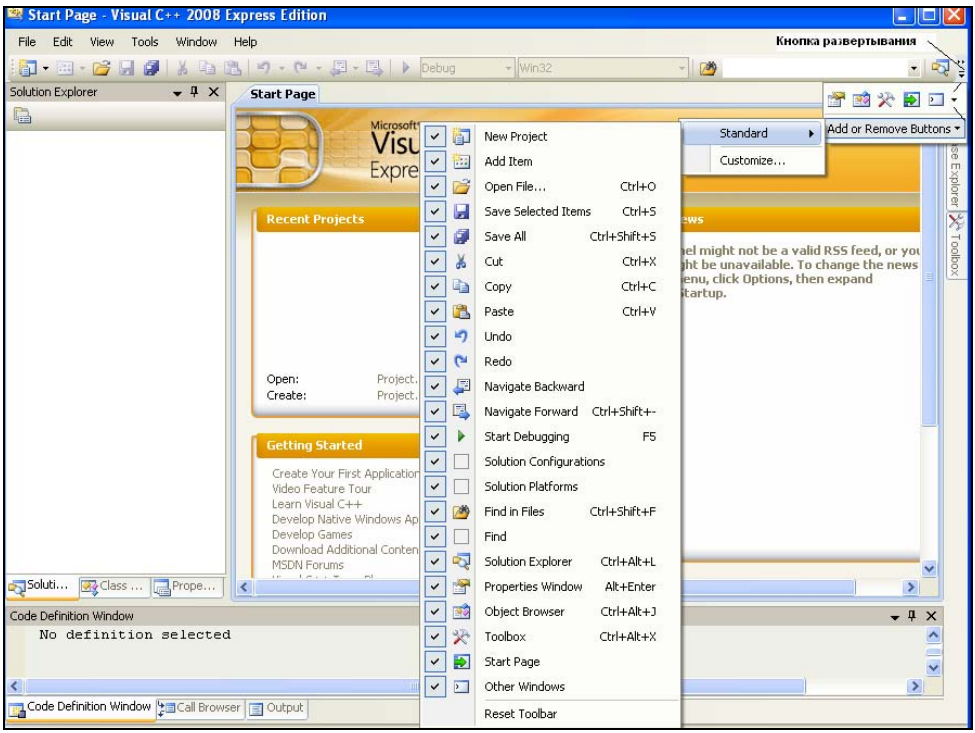

Рис. 1.3. Кнопки быстрого вызова

Рядом с такими кнопками могут быть дополнительные кнопки для раскрытия списка значений основной кнопки. Так как все кнопки не помещаются в отведенное им место на рабочем столе, то они свернуты в небольшие полосы с кнопками их развертывания точно так же, как это выполнено во всем известном Word'е (рис. 1.3).

Вид главного окна, в свою очередь, изменяется при задании типа создаваемого приложения. С этим мы познакомимся, когда начнем создавать приложения.

Структура программ в VC++

Программы в VC++ называются приложениями (очевидно, приложениями к среде IDE). Мы так и дальше станем их называть. Приложения строятся средой в виде специальных конструкций — проектов, которые выглядят для пользователя как совокупность нескольких файлов.

Программа на языке С — это совокупность функций, т. е. специальных программных образований, отвечающих определенным требованиям. Причем приложение — это главная функция, внутри которой помещаются операторы, реализующие алгоритм приложения. Среди операторов имеются такие, которые служат для вызова других функций, требующихся при реализации алгоритма. Запуск любой программы начинается с запуска *главной функции*, содержащей в себе всю остальную часть программы. Часть функций создается самим программистом, другая часть — библиотечные функции — поставляется пользователю со средой программирования и используется в процессе разработки программ. При изучении C/C++ мы будем пользоваться специальным видом приложений — консольными приложениями, которые формируются на основе заранее заготовленных в среде проектирования шаблонов.

Консольные (т. е. опорные, базовые) приложения — это приложения без графического интерфейса, которые взаимодействуют с пользователем через специальную командную строку или (если они работают в рамках IDE) запускаются специальной командой из главного меню среды. Такие приложения создаются с помощью специального шаблона, доступного из диалогового окна, открывающегося после выполнения команды **File|New Project**.

Шаблон консольного приложения добавляет в создаваемое приложение необходимые элементы (создается заготовка будущего приложения), после чего разработчик вставляет в этот шаблон свои операторы на языке C/C++. Затем приложение компилируется в автономный исполняемый файл и может быть запущено на выполнение. Общение с пользователем происходит через специальное так называемое консольное окно, открывающееся средой после запус-

ка приложения (в это окно выводятся сообщения программы, через него вводятся данные для расчета и в него же выводятся результаты расчетов).

Компиляция и сборка проекта осуществляется через команду **Build** главного меню среды. После компиляции и сборки проект можно запустить на выполнение. Запуск на выполнение осуществляется с помощью команды **Debbug** главного меню среды.

Переход к созданию консольного приложения

Для создания консольного приложения необходимо выполнить следующие шаги:

1. Загрузить среду VC++.
2. Выполнить команды главного меню **File|New|Project**. Откроется диалоговое окно, показанное на рис. 1.4. В этом окне выполните последовательно пронумерованные действия. После последнего действия откроется диалоговое окно (рис. 1.5).

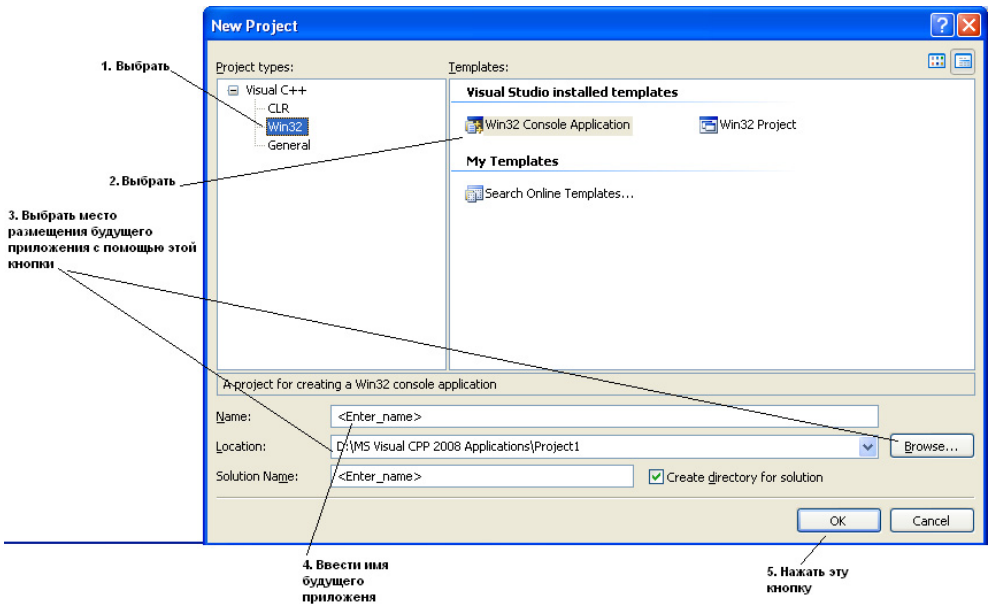

Рис. 1.4. Диалоговое окно **New Project**

Рис. 1.5. Диалоговое окно для определения типа создаваемого проекта

Рис. 1.6. Диалоговое окно для уточнения типа создаваемого проекта

3. В окне, представленном на рис. 1.5, предлагается выбрать тип создаваемого проекта (точнее подтвердить, что вы создаете консольный проект). Для

этого следует нажать на кнопку **Finish**. Дело в том, что среда позволяет создавать множество других типов проектов и задавать к ним дополнительные режимы, поэтому если вы вместо кнопки **Finish** нажмете на кнопку **Next**, то откроется другое диалоговое окно (рис. 1.6).

4. В окне, представленном на рис. 1.6, выбор типа осуществляется с помощью включения радиокнопок (это кнопки в виде окружностей, обладающие тем свойством, что когда одна кнопка включена, то все остальные выключены, что позволяет делать однозначный выбор). Заметьте, что на рисунке включена кнопка для консольного приложения. Если теперь нажать на кнопку **Finish**, то получится такой же результат, как если бы мы нажали на эту кнопку на предыдущем шаге. А результатом будет заготовка консольного приложения (рис. 1.7).

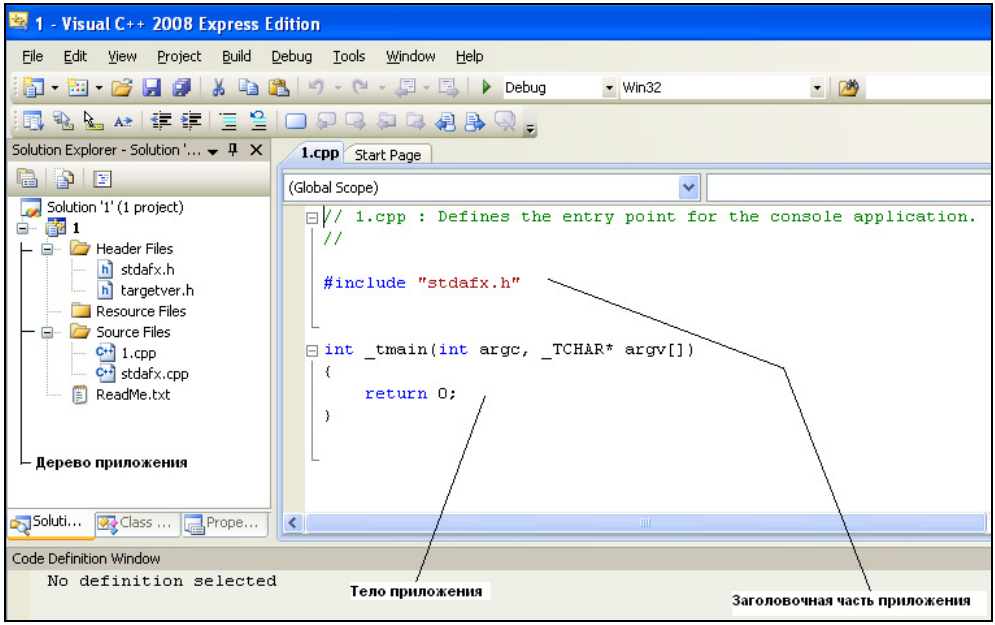

Рис. 1.7. Заготовка консольного приложения

5. Заготовка состоит из заголовка главной функции `int _tmain(int argc, _TCHAR* argv[])` и тела, ограниченного фигурными скобками. Преобразуем заголовок функции `_tmain` к виду `_tmain()`, а из тела удалим оператор `return 0`.

Все это проделаем с помощью Редактора кода, который открывается одновременно с появлением заготовки консольного приложения на экране (заготовка сразу помещается в поле Редактора кода). Чтобы убедиться, что вы находитесь в Редакторе, щелкните кнопкой мыши в любом месте поля заготовки и увидите, что курсор установится в месте вашего щелчка (Редактор ждет в этой точке ваших дальнейших действий). Далее можно набирать любой текст как в обычном современном текстовом редакторе, работать клавишами <Delete>, <Backspace>, клавишами-стрелками и другими необходимыми для ввода и редактирования клавишами.

Мы привели заголовок функции `_tmain` к виду `_tmain()`. Это означает, что наша главная функция не будет иметь аргументов, которые служат для связки консольных приложений. Этим мы заниматься не будем.

Среда VC++ оформляет создаваемое приложение в виде двух контейнеров, вложенных один в другой. Один (главный контейнер) называется Решение (Solution), а другой — Проект (Project). Проект определен как конфигурация (каркас, контейнер), объединяющий группу файлов.

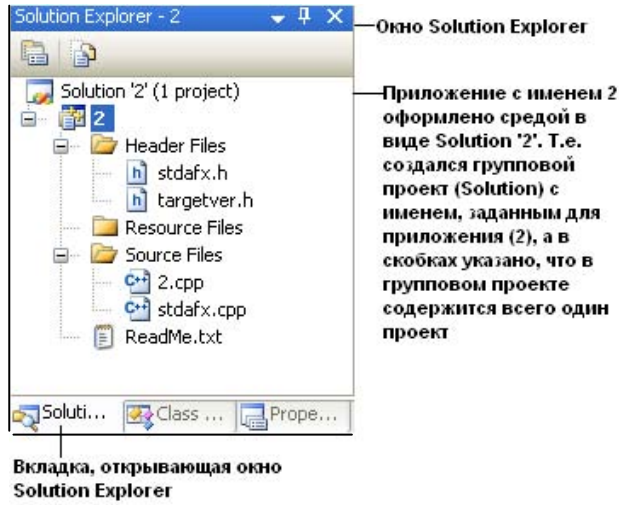

Рис. 1.8. Формирование проекта приложения

В рамках проекта создается программа, в т. ч. и подлежащая исполнению, т. е. откомпилированная и построенная. Каждый проект содержит по крайней мере две подконфигурации: отладочную и обычную (исполнительскую). Проекты являются частью другого каркаса, другого контейнера, который на-

зывается Solution (решение) и который отражает взаимосвязь между проектами: одно Решение может содержать множество проектов, а проект содержит множество элементов, обеспечивающих существование приложения как такового. Можно сказать, что Решение — это не что иное, как группа объединенных проектов. Назовем его просто: Групповой проект, чтобы термин "Решение" не вводил нас в заблуждение. Существует специальный инструмент работы с групповым проектом, называемый Solution Explorer. К нему можно добраться через опцию **View** меню среды разработки. Сама среда автоматически формирует создаваемое приложение как групповой проект, содержащий собственно проект (это видно из рис. 1.8).

Такой подход к оформлению приложения позволяет работать с группой проектов как с одним целым, что ускоряет процесс разработки приложений. Следует отметить, что не все файлы проекта отображаются в окне **Solution Explorer**.

В качестве примера создадим проект с именем **1** и добавим его к имеющейся группе **2**. Для этого щелкнем правой кнопкой мыши на строке **Solution '2'** и в появившемся контекстном меню выберем команду **Add|Existing Project…** (рис. 1.9).

При этом откроется диалоговое окно для поиска проекта, затем обычным способом откроем проект **1**, в результате чего он добавится к **Solution '2'** (рис. 1.10).

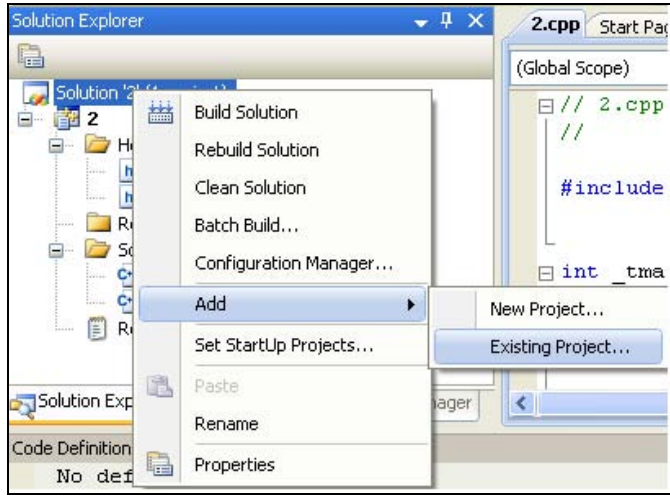

Рис. 1.9. Процесс добавления созданного ранее проекта к группе проектов (Solution 2)

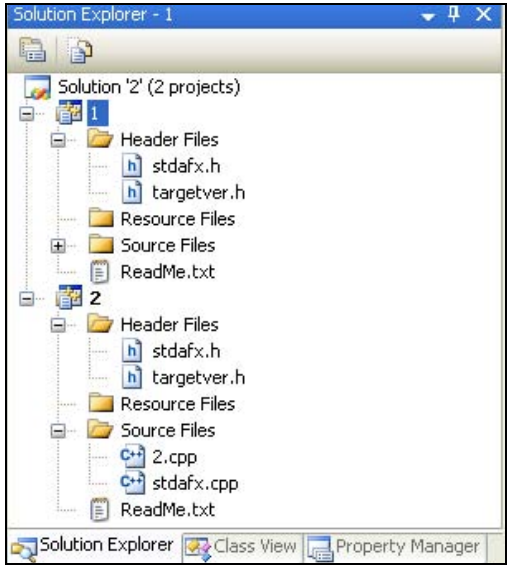

Рис. 1.10. Solution '2' с добавленным к нему проектом **1**

Добавленный проект, как и любой другой, входящий в группу проектов, можно удалить. Для этого надо, щелкнув мышью на имени этого проекта, открыть его контекстное меню и выполнить в нем команду **Remove**.

Примечание

Часто бывает необходимо быстро загрузить ранее сохраненные проекты (не искать их в каталогах). Для этого существует команда главного меню **File|Recent Projects**, которая открывает меню с названиями и путями к ним для ранее выполненных проектов.

Удалим из группы проектов проект **2** и оставим в ней проект **1**. Получим вид рабочего стола среды, показанный на рис. 1.11.

Из рисунка мы видим, что в окне **Recent Projects** (где указаны проекты, с которыми недавно работала среда) имеется список проектов. К каждому из них можно вернуться, если щелкнуть на его имени. Выполнив команду **Remove** (Удалить проект), мы всего лишь удалили сведения о проекте из окна **Solution Explorer**, но не из памяти, поэтому-то и возможно восстановление проекта.

Чтобы удалить проект из памяти, надо найти папку, в которой создан проект, удалить его из папки, а потом щелкнуть на его имени в окне **Recent Projects**,

показанном на рис. 11.1. Если проект из памяти был вами удален, то после щелчка среда выдаст сообщение, что такого проекта нет и попросит подтвердить его удаление. При подтверждении имя проекта удалится из окна **Recent Projects**. Таким способом вы сможете удалять ненужные проекты не только из группы, но и из окна **Recent Projects**, чтобы они вас не раздражали. После всех манипуляций с проектами мы получим созданную заготовку нашего приложения **1**, показанную на рис. 1.12.

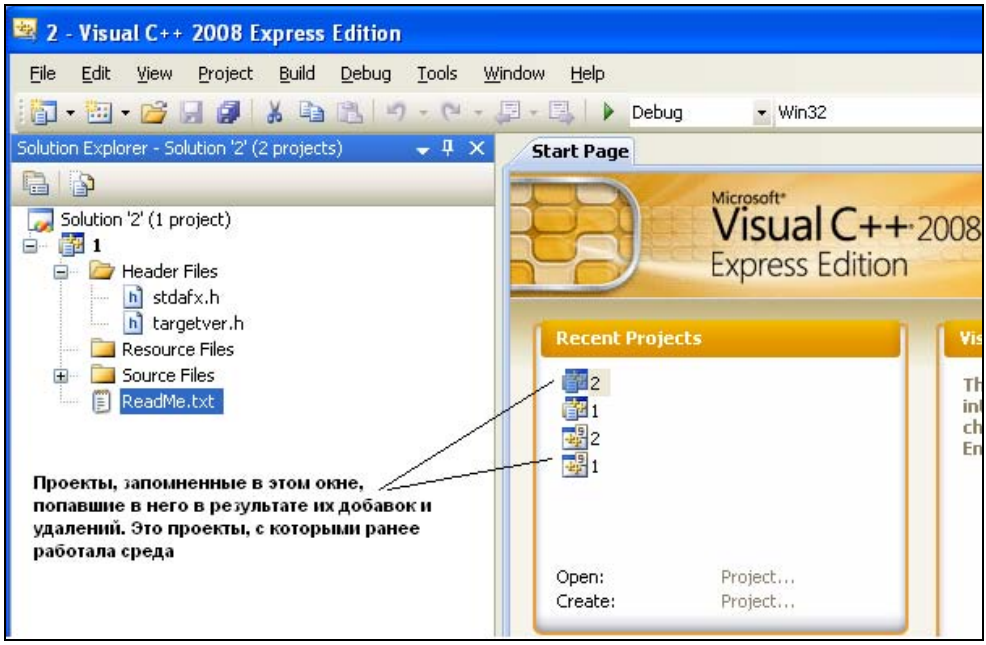

Рис. 1.11. Вид рабочего стола среды после нескольких манипуляций с группой проектов

Мы видим, что над самой заготовкой консольного приложения имеются две вкладки: **1.cpp** и **Start Page**. Если щелкнуть мышью на 1-й из них, то на рабочем столе появится консольная заготовка-проект. Расширение cpp означает, что это исходный текст проекта на языке C++. Если щелкнуть мышью на вкладке **Start Page**, то на рабочем столе появится стартовая страница среды (нижняя часть рис. 1.12). Таким способом можно переключаться с одного объекта на другой. Вообще, если вы добавите к проекту еще какой-нибудь элемент, воспользовавшись контекстным меню проекта и выполнив команду **Add** (например, новый cpp-файл), то для добавленного элемента тут же на

рабочем столе откроется новая вкладка с именем добавленного элемента, чтобы можно было переключаться между элементами проекта.

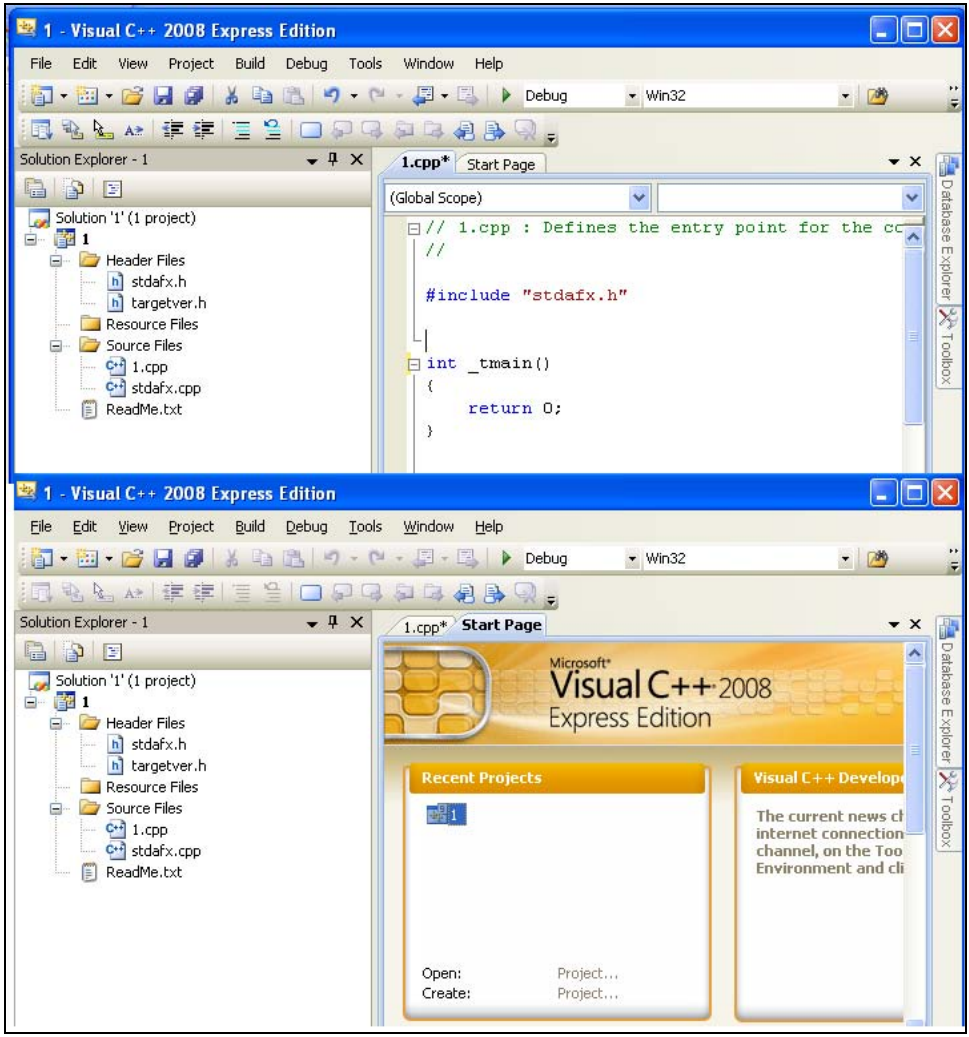

Рис. 1.12. Заготовка консольного приложения и стартовая страница

Теперь посмотрим, что за файлы попали в проект, показанный на рис. 1.12:

- **1.cpp** — это главный исходный файл и точка входа в создаваемое приложение (1 — это в данном случае имя проекта);

- **stdafx.cpp** — подключает специальный файл Vs9VimTestCpp.pch для компиляции приложения;
- **stdafx.h** — подключает специальные файлы для компиляции приложения. Вот его содержимое:

```
#pragma once // команды для компиляции:
#include "targetver.h"
#include <stdio.h>
#include <tchar.h>
```

- **targetver.h** — позволяет использовать специфические свойства Windows Vista. Вот содержимое этого файла:

```
#pragma once  //команды для компиляции:
#ifndef _WIN32_WINNT   // Allow use of features
                       //specific to Windows Vista or later.
#define _WIN32_WINNT 0x0600    // Change this to the
                               //appropriate value to target
                               //other versions of Windows.
#endif
```

- **ReadMe.txt** — файл, описывающий некоторые из созданных шаблоном консольного приложения файлов проекта. Посмотреть содержимое файла можно через его контекстное меню, если выполнить в нем команду **Open**.

Примечание

Мы создали заготовку, построенную по шаблону Win32 Console Application. Но существует и шаблон CLR Console Application, однако мы его не выбрали для построения заготовки. Почему? Этот шаблон предполагает подключение к тексту программы специального пространства System, содержащего классы, которые задают ссылочные типы данных и функции работы с ними. Мы этого аппарата еще не касались и столкнемся с ним в *главе 7*. Там нам и понадобится другой тип консольного приложения, а именно CLR Console Application. Подключение шаблона CLR Console Application к созданию приложения происходит там же, где и подключение к шаблону Win32 Console Application.

Создание простейшего консольного приложения

Запишем в теле функции `_tmain()` следующие две строки:

```
printf("Hello!\n");
_getch();
```

Это код нашего первого приложения. Он должен вывести на экран текст "Hello!" и задержать изображение, чтобы оно не исчезло, пока мы рассматриваем, что там появилось на экране. Вывод на экран выполняет оператор `printf("Hello!\n");`, а задержку изображения — оператор `_getch();`.

Заметим, что оператором в C/C++ называют некоторое выражение C/C++, оканчивающееся точкой с запятой. В первый оператор входит функция `printf("Hello!\n")`, а во второй — функция `_getch()` (эта функция из C/C++ введена вместо ранее использовавшейся функции `getch()`).

В итоге наше консольное приложение будет иметь вид, представленный на рис. 1.13.

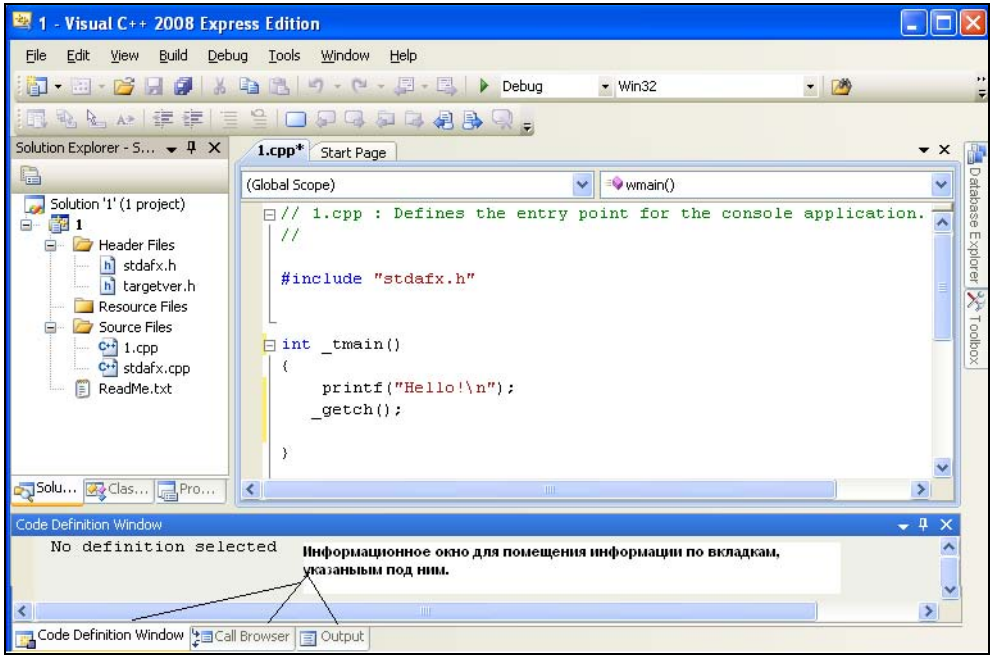

Рис. 1.13. Вид консольного приложения до компиляции

Чтобы приложение заработало, его надо *откомпилировать*, т. е. перевести написанное на языке C в машинные коды. Для этого запускается программа-компилятор. Запускается она либо нажатием клавиши <F7>, либо выполнением опции главного меню **Build|Solution**. Если мы проделаем подобные действия, то получим картинку, показанную на рис. 1.14.

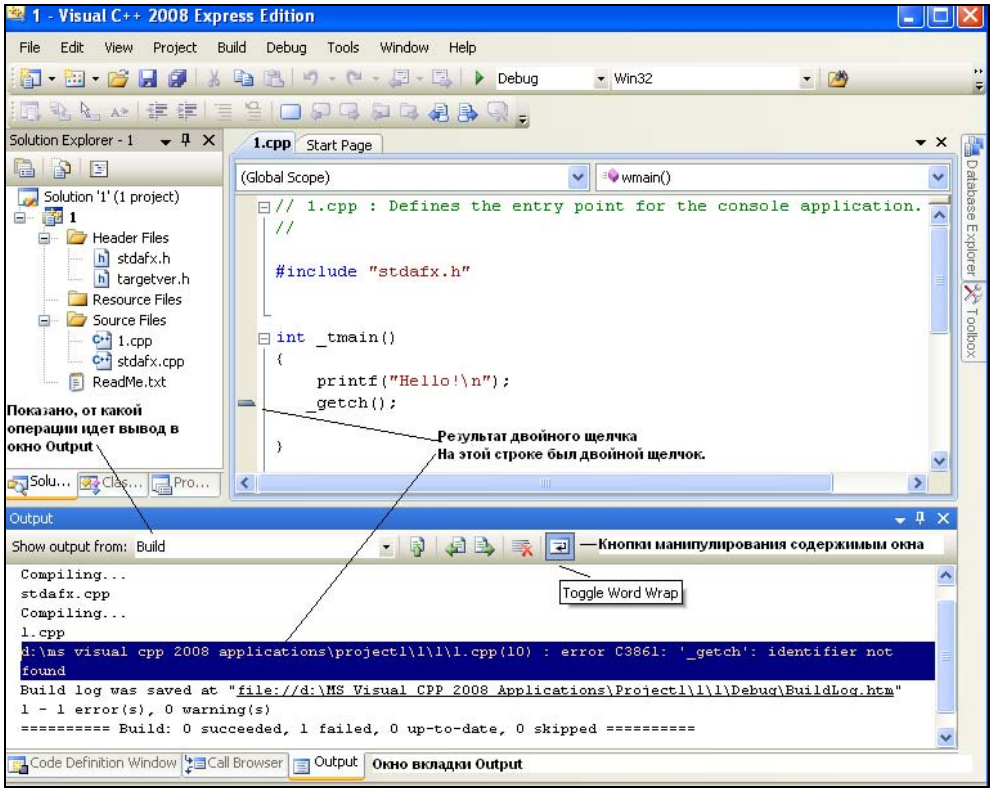

Рис. 1.14. Результат 1-й компиляции приложения

Картинка показывает, что наша компиляция не удалась: в окне вывода высветились сообщения об ошибках. В частности, мы видим сообщение: "error C3861: '_getch': identifier not found". Это означает, что компилятор не узнает функцию `_getch`.

А почему же компилятор узнал функцию `printf()`? Если посмотреть по справочной системе, то увидим, что местонахождение этой функции находится в файле stdio.h. Но если вспомнить, какие файлы автоматически входят в проект при его построении, то увидим, что файл stdafx.h как раз и подключает к программе файл stdio.h, вот поэтому-то компилятор ведет себя спокойно по отношению к функции `printf()`.

Если кнопкой мыши дважды щелкнуть на каждой строке с информацией об ошибке, то в поле функции `_tmain()`, т. е. в нашей программе в поле подшивки (вертикальная полоса слева от текста) отметится та строка, в которой эта ошибка обнаружена. Этот процесс также показан на рис. 1.14.

А теперь разберемся с обнаруженными ошибками.

Щелкнем дважды на имени функции `_getch()`, чтобы пометить ее, и нажмем клавишу <F1>. Откроется окно помощи. В появившемся окне (его фрагмент приведен на рис. 1.15) мы найдем сведения о необходимой нам функции, в том числе и о местонахождении ее описания (Header file conio.h).

Рис. 1.15. Фрагмент окна **Help** после нажатия клавиши <F1> на отмеченном элементе `_getch()`

Если этот файл подключить к нашей программе, то компилятор станет находить сведения об этой функции и ошибка устранится. Включение файла с описанием функции (Header file — это файл оглавления, отсюда и символ h в его расширении) осуществляется оператором `#include` — это оператор компилятора. Он включает в текст программного модуля файл, указанный в угловых скобках. Но вы и сами можете задавать в подобных файлах необходимую для вашей программы информацию. Тогда для ее подключения к вашей программе имя такого файла в операторе `#include` должно быть в двойных кавычках.

После подключения файла с описанием функции `_getch()` запускаем компилятор клавишей <F7>. Результат показан на рис. 1.16. Заметим, что Редактор текста программы рассматривает две косые черты `//` как начало комментария — набора символов, не участвующих в выполнении программы. Заметим также, что на самом деле не запускают отдельно компиляцию, а выполняют