Борис Пахомов

C/C++ и MS Visual C++ 2010

+ДИСТРИБУТИВ Microsoft® Visual Studio® 2010 Express Edition

Основные элементы языков C/C++

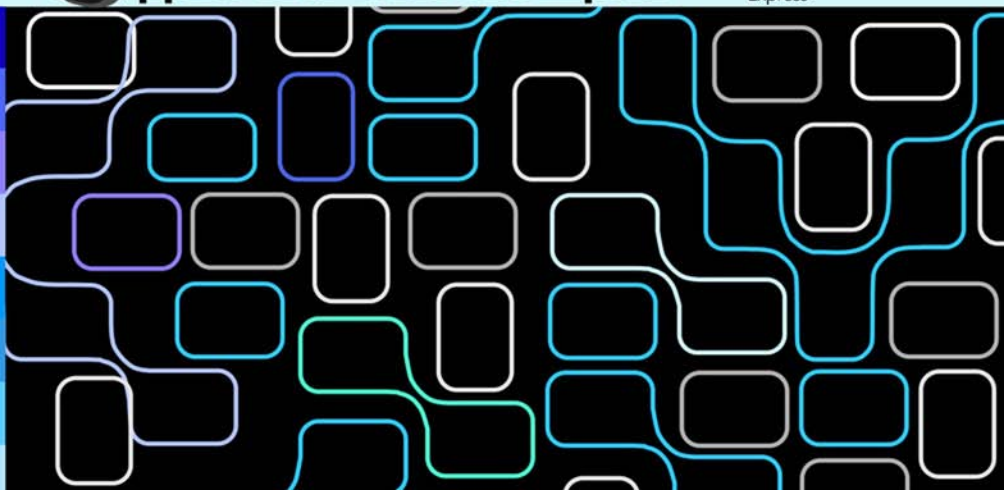
Визуальная среда программирования

Создание основных типов приложений

Работа с наборами данных

ДЛЯ НАЧИНАЮЩИХ

Microsoft Visual Studio 2010 Express



Борис Пахомов

C/C++ и MS Visual C++ 2010

для начинающих

Санкт-Петербург
«БХВ-Петербург»
2011

УДК 681.3.06
ББК 32.973.26-018.1
П22

Пахомов Б. И.

П22 С/С++ и MS Visual C++ 2010 для начинающих. — СПб.: БХВ-Петербург, 2011. — 736 с.: ил. + дистрибутив (на DVD)

ISBN 978-5-9775-0599-4

Книга является руководством для начинающих по разработке приложений в среде Microsoft Visual C++ 2010. Рассмотрены основные элементы языков программирования С/С++ 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++ 2010, назначение базовых компонентов и процесс разработки различных типов консольных и Windows-приложений. На DVD размещен дистрибутив пакета Microsoft Visual Studio 2010 Express Edition, содержащий Visual C++ 2010 Express Edition и другие компоненты пакета.

Для начинающих программистов

УДК 681.3.06
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	*Наталия Першакова*
Дизайн серии и оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 30.08.10.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 59,34.
Тираж 2000 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Санитарно-эпидемиологическое заключение на продукцию № 77.99.60.953.Д.005770.05.09 от 26.05.2009 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 978-5-9775-0599-4

Оглавление

Введение

В этой книге рассматривается Visual C++ 2010 версии RC (в дальнейшем VC++).

Книга предназначена для начинающих программистов, поэтому многие вопросы раскрыты не во всей глубине, ибо читатель должен научиться пользоваться продуктом, а не тонуть в подробностях. Представленного материала достаточно для написания консольных приложений, а также многих приложений типа Windows Form. Автор надеется, что полученные знания явятся ступенью для изучения интересного и полезного (несмотря на его недостатки) программного продукта, каким, несомненно, является MS Visual C++ 2010. Хотя выхода в Интернет на создание Web-приложений с помощью ASP.NET в этом облегченном и бесплатно-доступном продукте нет, но для начинающего вполне достаточно изучить язык и Windows-формы, чтобы в дальнейшем, имея определенный багаж знаний, заниматься проблемами Интернета. Кстати, и в этом урезанном варианте некоторые выходы в Интернет имеются (например, через компонент `LinkLabel`).

Существенным недостатком этой версии (как и версии 2008 г.) является отсутствие возможности работы с базами данных. При представлении Visual C++ 2010 в апреле 2010 г. в городах России представители разработчика, внимание которых было обращено на указанный недостаток, заверили присутствующих, что он будет в ближайшее время устранен.

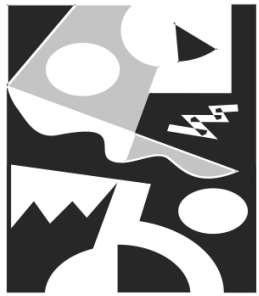
В связи с этим в данной книге описана имеющаяся возможность работы с базами данных в версии 2005 г. (*глава 12*), чтобы читатель был подготовлен для работы с будущей усовершенствованной версией продукта. А в *главе 15*, посвященной выводу отчетов и содержащей создание и поддержку БД типа MS Access средствами Visual C#, приводится искусственный прием, позволяющий обойти неудобства работы с БД в версии 2010 г.

Хочу отметить, что данная книга не только уточняет многие моменты, опущенные в прежнем издании, но и дополнена новыми примерами и новыми главами. Поэтому на ее фундаменте начинающему программисту будет значительно легче войти в среду VC++, нежели с помощью моей предыдущей книги "C/C++ и Microsoft Visual C++ 2008 для начинающих".

ЧАСТЬ I

Изучение основных элементов языка C/C++

Глава 1



Обзор среды программирования

Общие положения

Допустим, вам требуется решить некоторую задачу (с помощью компьютера, конечно). Например, нужно рассчитать движение материальных ценностей по некоторому складу: сколько чего было на данную дату, сколько чего поступило, сколько ушло, сколько осталось. С чего обычно начинают? Ясно сразу, что если задачу надо решать на компьютере, то следовало бы ее решение как-то формализовать, т. е. алгоритм (набор последовательных действий, исполнение которых приводит к решению задачи) требуется привести к последовательности неких формальных действий, понятных машине (говорят, что надо построить машинный алгоритм решения задачи). Затем надо продумать форму общения (интерфейс) для пользователя с самим компьютером, исходя из максимального удобства общения. Это чуть ли не одна из главных трудностей проектирования решения задачи, ибо неудобство общения раздражает пользователя, который начинает совершать ошибки, что в конечном счете может привести к тому, что ваш проект просто будет отвергнут. Имеются еще некоторые шаги по подготовке к решению, но мы их здесь опустим, т. к. это не наша проблема.

Итак, мы изучили задачу, создали машинный алгоритм ее решения на компьютере, разработали интерфейс взаимодействия будущего пользователя с компьютером. И что дальше? А дальше все эти разработки надо перевести на понятный компьютеру язык, т. е. запрограммировать наши действия, составить машинную программу, которая представляет из себя последовательность определенных команд, записанных на алгоритмическом языке. Для решения конкретной задачи с учетом разработанного интерфейса подходит не всякий алгоритмический язык. Поэтому разработчику надо выбрать подходящий к данной ситуации язык, который бы удовлетворял определенным

требованиям. Который не просто обеспечивал бы ее программирование, но и отвечал бы еще множеству других условий: позволял программисту быстро и надежно создавать программу, обеспечивал удобное сопровождение программы в период ее эксплуатации и т. д.

Когда программа написана на некотором алгоритмическом языке, она должна быть переведена в машинный язык — в язык, на котором работает компьютер, в его систему команд. Для этого существуют специальные программы, называемые компиляторами. Эти программы имеют параметры, задание которых позволяет компилятору создавать машинные программы в той или иной плоскости. Например, существуют параметры, позволяющие компилятору минимизировать размер памяти, которую будет занимать скомпилированная программа. Или (как мы будем рассматривать в данной книге) существует параметр, позволяющий компилятору создавать программы (часто говорят просто "коды"), состоящие из так называемых управляемых или неуправляемых кодов. Параметры компилятора называют по-разному: ключами, опциями.

Но программу мало откомпилировать. Компилирование — это только первый этап создания машинной программы. Дело в том, что для решения конкретной задачи (т. е. реализации ее машинного алгоритма) требуется подключение неких стандартных библиотек, содержащих стандартные программы, которые разрабатываются один раз и используются во многих алгоритмах (например, перевод десятичных чисел в двоичные или шестнадцатеричные). Ни один программист, пишущий прикладную программу, не станет всякий раз заниматься этим переводом. Поэтому в подобных случаях разработчик программной среды, в рамках которой создается программный продукт (в нашем случае это Visual Studio 2010) сам создает подобные библиотеки и поставляет их со средой разработки, которая содержит и компиляторы с разных языков среды в машинные коды. В свою очередь, и опытный программист может самостоятельно создавать такие библиотеки и включать их в общий перечень библиотек среды программирования, чтобы они в дальнейшем автоматически подключались к решению задач. Для этой цели среда поставляет специальные средства.

Но вернемся к компиляции. Компилятор, просматривая программу (код, как говорят), переводит этот код в машинные команды. Но не просто в набор команд, а формирует все это множество в виде отдельных (объектных) модулей. В каждом таком модуле создается своя таблица имен со ссылками на их месторасположение. Таким образом, компилятор создает не исполняемый код, а так называемый объектный код, содержащий неконкретные (как говорят "неразрешенные", т. е. неопределенные) ссылки. На этом его работа за-

вершается. Чтобы получить исполняемый модуль, надо "разрешить" (т. е. конкретизировать) ссылки, сформированные компилятором с учетом конкретного размещения данного кода в выделенной для него памяти компьютера. Эту работу выполняет специальная программа, которая называется поразному: редактор связей, компоновщик, линковщик, построитель. После работы этой программы получается набор машинных команд, готовых к исполнению на компьютере.

В свете всего вышесказанного можно утверждать, что для того чтобы создать программу, требуется еще иметь средства ее компиляции и компоновки. Эти средства поставляются в рамках изучаемой нами среды MS Visual C++ 2010. То есть прежде, чем перейти к изучению собственно языка C/C++, надо познакомиться хотя бы в общих чертах с интерфейсом среды обработки данных MS Visual C++ 2010, чтобы иметь возможность с его помощью компилировать и строить исполняемые программы в рамках языка C/C++.

Структура рабочего стола среды программирования

Цель этой главы — продемонстрировать начальные элементы программирования на языке C./C++. Язык C++ является дальнейшим развитием языка C, поэтому все конструкции языка C поддерживаются в C++. Однако в C++ появились новые возможности синтаксиса, не имеющиеся в C (это мы увидим по мере рассмотрения материала).

Чтобы построить программу на этом языке, нам надо воспользоваться средой программирования Visual C++ 2010 (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++ 2010 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++ 2010 (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.

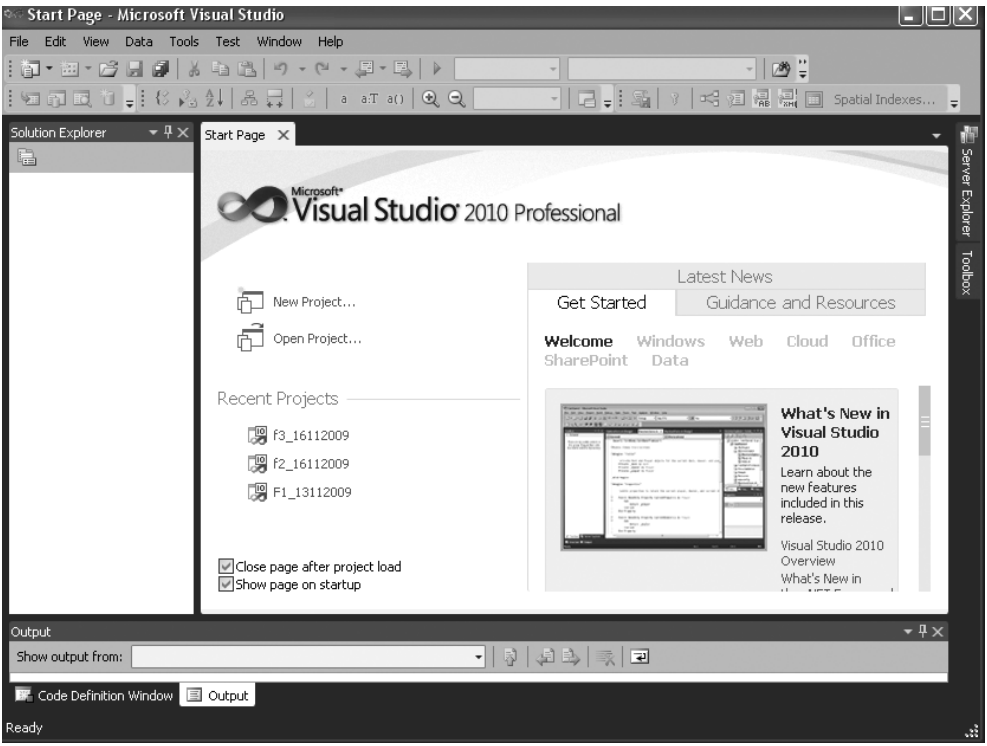


Рис. 1.1. Вид главного окна IDE после загрузки VC++

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 несколь-

ких 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.

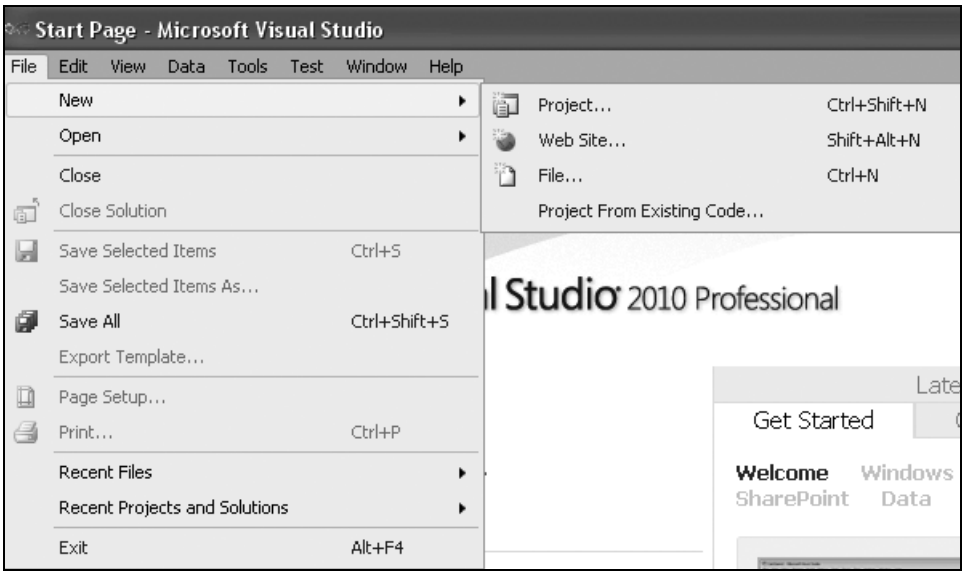


Рис. 1.2. Меню для задания типа формируемого приложения

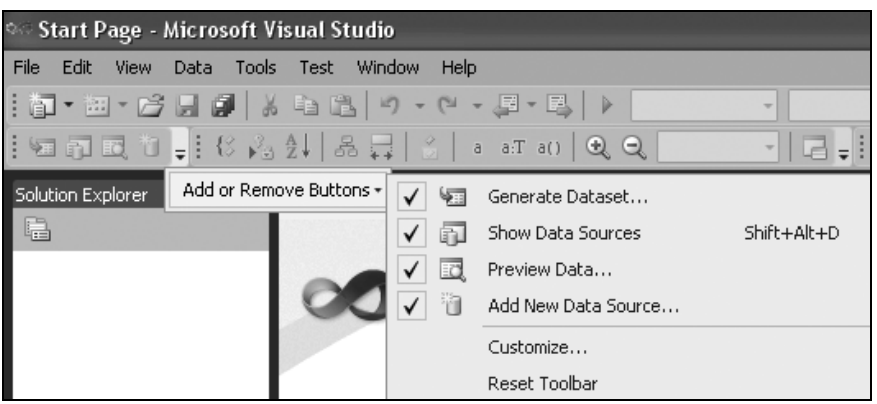


Рис. 1.3. Кнопки быстрого вызова

Ниже строки 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). 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 (рис. 1.3) точно так же, как это выполнено во всем известном Word'е.

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

Некоторые замечания

О рабочем столе

Рабочий стол формируется из набора окон. Каждое окно — это обычное windows-окно, имеющее стандартную заголовочную полосу в своей верхней части. За эту полосу можно окно перемещать, протягивая мышью. У окон имеются свойства, которые открываются, если на заголовочной части щелкнуть правой кнопкой мыши. Перечень свойств окна **Solution Explorer** показан на рис. 1.4.

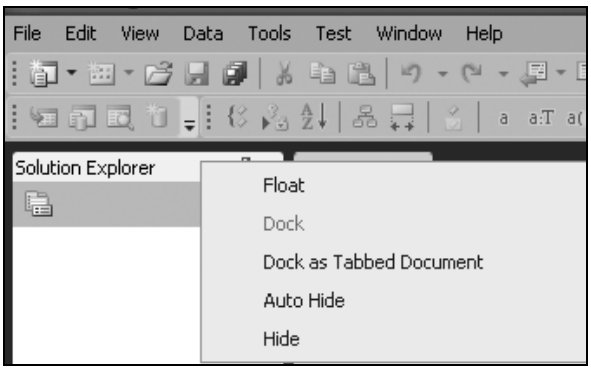


Рис. 1.4. Меню свойств окна

Рассмотрим некоторые свойства:

- **Float** (плавающее). Такое окно можно перетягивать в любую часть рабочего стола;

- **Dock** (причаливающее). Такое окно может перемещаться, но будет причаливать (т. е. захватываться другим окном);
- **Dock as Tabbed Document** (причаливать в качестве вкладки в основное окно рабочего стола). Это окно, в котором первоначально в качестве вкладки располагается стартовая страница Start Page;
- **Auto Hide** (автоматически исчезать). В этом случае окно автоматически "прячется" (причаливает) в качестве вкладки к ближайшей боковой стороне основного окна рабочего стола, а при наведении курсора мыши на имя окна оно автоматически всплывает;
- **Hide** (спрятать). При этом свойстве окно исчезает с экрана. Чтобы оно снова появилось, надо воспользоваться либо опцией **View** главного меню, либо соответствующей данному окну опцией **Other Windows** опции **View**.

Чтобы манипулировать положением окон на рабочем столе, надо хорошо понимать все свойства, иначе может сложиться ситуация, когда на рабочем столе соберется множество окон, которые просто будут мешать работать. В таком случае некоторые окна надо будет "разогнать" по боковым сторонам основного окна, а другие просто спрятать.

Например, как установить справа-сбоку основного окна рабочего стола окна **Toolbox** и **Server Explorer**?

У этих окон надо установить свойство (через правую кнопку мыши) **Dock** (причаливание). Тогда в заглавной строке окна появится значок **Auto Hide** (скрывать автоматически). Значок имеет вид вертикального полупроводника. Если на этом значке щелкнуть, то окно причалит к правой стороне главного окна среды и спрячется (останется видимым только его имя). Если теперь навести курсор мыши на это имя, то окно автоматически всплывет, и в его заголовочной части мы увидим значок **Auto Hide** в виде горизонтального полупроводника.

О справочной системе Help

Все всегда начинается с вопроса: "А где посмотреть, как это делается?" Ясно, что в справочной системе к программному продукту. Откроем опцию **Help** главного меню к MS VC++ 2010 (рис. 1.5).

Если выполнить отмеченную опцию, показанную на рис. 1.5, то вполне возможно, что среда выдаст ошибку. Дело в том, что сначала надо выполнить вторую опцию показанного рисунка, которая дает возможность задать неко-

торые параметры Help'а. Выполним эту опцию. Получим диалоговое окно, показанное на рис. 1.6.

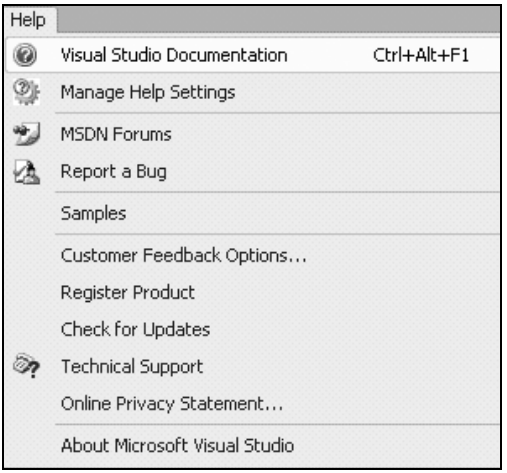


Рис. 1.5. Меню **Help**

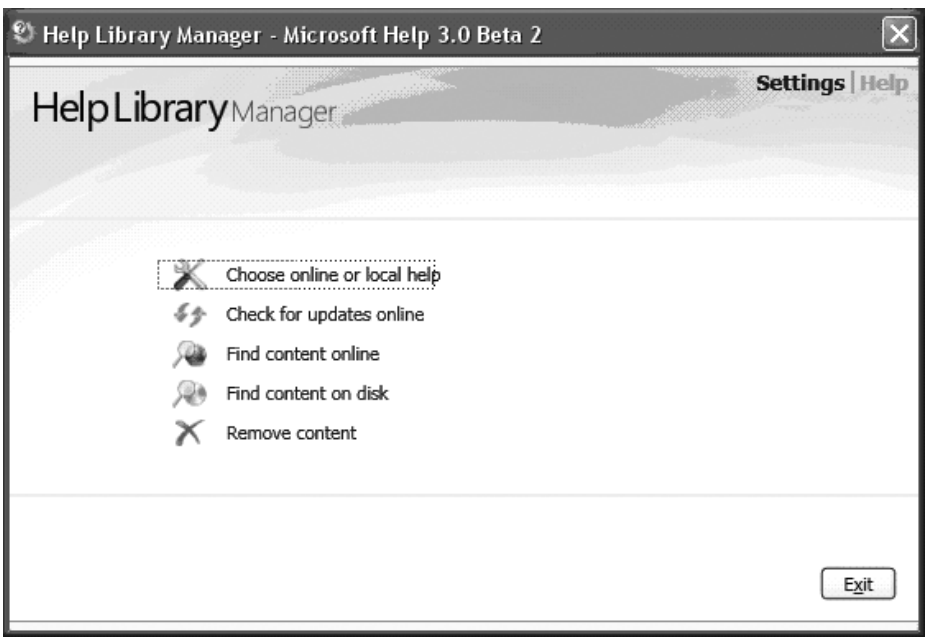


Рис. 1.6. Выход в окно для установок места, откуда будет читаться Help

Нас интересует верхняя опция меню, показанного в этом окне. Она позволяет задать условие, согласно которому вы станете доставать вашу справку: из локального Help'а или из Интернета. В современных программных продуктах преимущество отдается работе с Help'ом в режиме online, т. е. через Интернет. Это и понятно: локальную справку невозможно детально отразить на вашем компьютере, потому что, например, у вас просто может не хватить локальной памяти. А база данных, в которой находится справочная система разработчика, содержит полную справку, и она может в различных разрезах поставляться по запросу пользователя. На рис. 1.6 мы выбираем первую опцию и получаем диалоговое окно, показанное на рис. 1.7.

В этом окне мы щелкаем мышью на переключателе **I want to use online help** (Я хочу пользоваться Help'ом из Интернета) и нажимаем на кнопку **ОК**. Вот теперь, когда вы начнете выполнять первую опцию, показанную на рис. 1.5, ваш браузер откроет вам доступ к справочной системе, как это показано на рис. 1.8.

Если пользоваться Help'ом в английском варианте затруднительно, можно найти в Интернете его русскоязычный аналог.

Вот ссылка на него: **http://msdn.microsoft.com/ru-ru/default.aspx**.

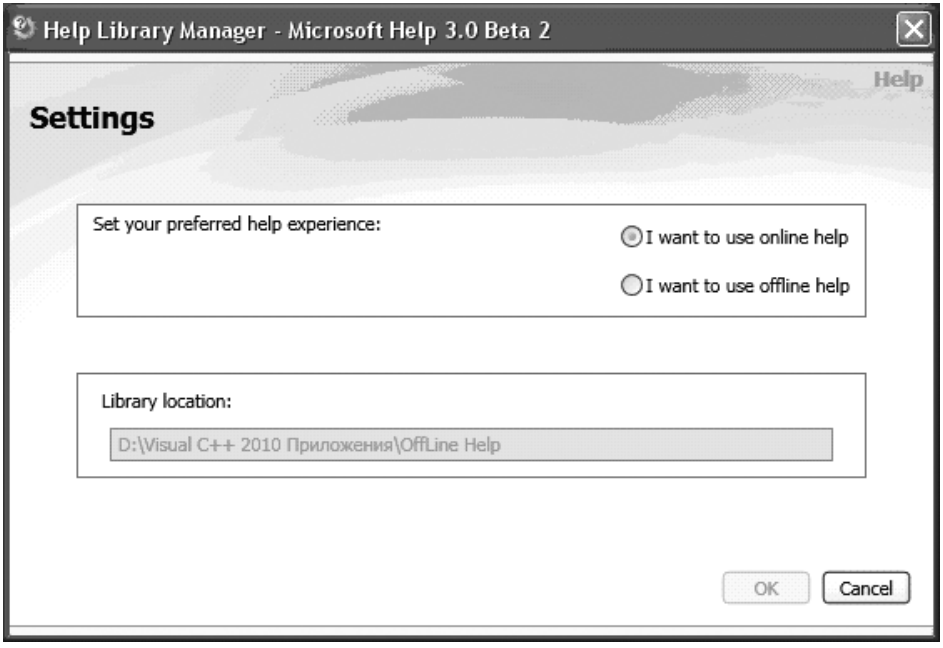


Рис. 1.7. Выбор места, откуда будет читаться Help

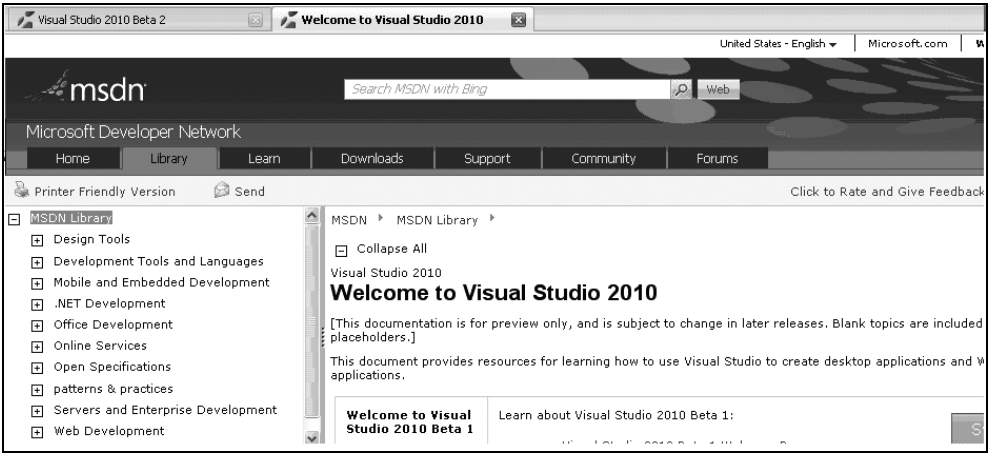


Рис. 1.8. Окно **msdn**

Структура программ в VC++

Программы в VC++ называются приложениями (очевидно, приложениями к среде программирования, 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 C — 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 приложе-

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

Компиляция и сборка проекта осуществляются через команду **Build** главного меню среды (структура главного меню меняется в зависимости от совершаемых действий: при загрузке среды она одна (в ней нет опции **Build**), а например, если мы создадим проект, такая опция появляется). После компиляции и сборки проект можно запустить на выполнение. Запуск на выполнение осуществляется с помощью опции **Debbug** главного меню среды.

Изучение C/C++ мы станем осуществлять на примерах: будем создавать программы, разбирать, как они работают с параллельным изучением их структуры. Консольные приложения, которые мы начнем создавать, имеют шаблон вида CLR Console Application. Последние два слова этого названия понятны — консольное приложение. А что означает аббревиатура CLR?

В Visual C++ 2008 для организации консольных приложений существовало два шаблона: один — тот, что мы рассматриваем, а другой — без аббревиатуры CLR. В новой редакции среды разработки авторы тоже оставили два варианта консольного приложения, только разнесли их использование по разным диалоговым окнам: шаблон с CLR задан в папке CLR, а шаблон для обычного консольного приложения задан в папке Win32.

Так что же это такое CLR? Это специальная среда, которая управляет исполнением программного кода, памятью, потоками данных и работой с удаленными компьютерами, при этом строго обеспечивая безопасность и создавая надежность исполнения кода. CLR является добавкой-расширением C++, введенной фирмой Microsoft, начиная с версии VC++ 2005.

В версиях 2005, 2008 подключение к вашему проекту этой среды осуществлялось на этапе компиляции. Там для консольного приложения (по умолчанию) этот режим не поддерживался (т. е. консольное приложение как бы оставалось в старой версии C++, в "родной" (native), как определили ее авторы добавки CLR).

В этой старой версии, когда вы в своей программе работали с некоторыми объектами (здесь нам приходится забегать вперед, ибо объекты — это предмет более позднего изучения, когда мы станем знакомиться с классами), то должны сами заботиться об их размещении в памяти, выделяемой средой. Память для вашего приложения выделяется в так называемой "куче": в ней вы размещаете свои объекты, там же сами освобождаете память, когда перестаете работать с объектом, иначе куча может переполниться и процесс выполнения приложения прервется. Это так называемая неуправляемая куча. Указатели (о них — позже) на участки памяти в такой куче обозначаются символом "*".

Другое дело, когда включается режим CLR. Такое приложение отличается от обычного тем, что его заготовка обеспечивает подключение к приложению специального системного пространства System, содержащего объекты, размещение в памяти которых надо автоматически регулировать. Так вот: режим CLR работает уже с управляемой кучей памяти, в которой размещение объектов и ее освобождение от них происходит под управлением среды. Такой сервис входит в язык Java, где не надо делить кучу на управляемую и неуправляемую. В этой среде употребляются так называемые регулируемые указатели на объекты.

Регулируемый указатель — это тип указателя, который ссылается на объекты (адреса памяти, по которым можно обращаться к объектам), расположенные в общей регулируемой куче памяти, предоставленной приложению в момент его исполнения. Для таких указателей принято специальное обозначение: вместо символа "*" применяется символ "^".

Создание CLR привело к необходимости разработки аппарата преобразования переменных, относящихся к одной куче, в адреса в другой и т. п. Этот процесс назвали маршализацией. Существует специальная библиотека, обеспечивающая этот процесс. Однако все это усложнило программирование.

Переход к созданию консольного приложения

Для создания консольного приложения мы станем пользоваться шаблоном CLR-приложения. Для этого выполним следующие шаги:

1. Загрузим среду VC++.
2. Выполним команду главного меню **File | New | Project**. Откроется диалоговое окно, показанное на рис. 1.9.

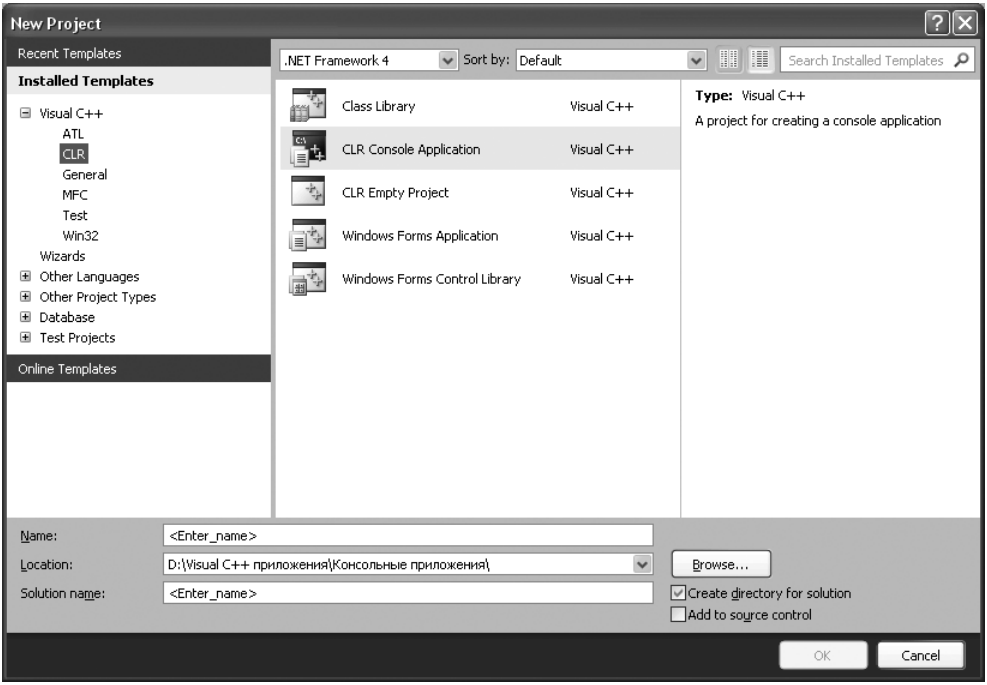


Рис. 1.9. Диалоговое окно **NewProject**

3. В этом окне выберем опцию **CLR Console Applicaton**, зададим в его нижней части имя будущего проекта в поле **Name**, которое перекочует в поле **Solution name**, затем с помощью кнопки **Browse** установим папку, в которую будет помещен наш проект. Нижняя часть окна рис. 1.9 станет выглядеть так, как показано на рис. 1.10.

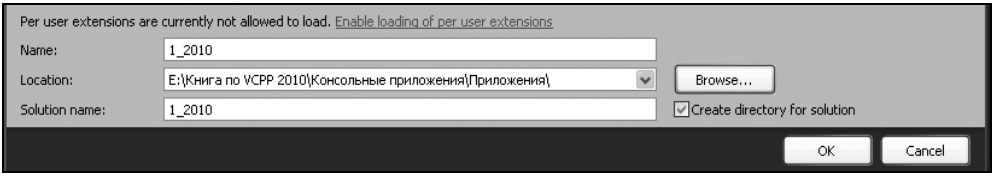


Рис. 1.10. Формирование консольного приложения

4. Затем нажмем кнопку **ОК**. В результате получится то, что показано на рис. 1.11.

Заготовка состоит из заголовка главной функции:

```
int main(array<System::String ^> ^args)
```

и тела, ограниченного фигурными скобками.

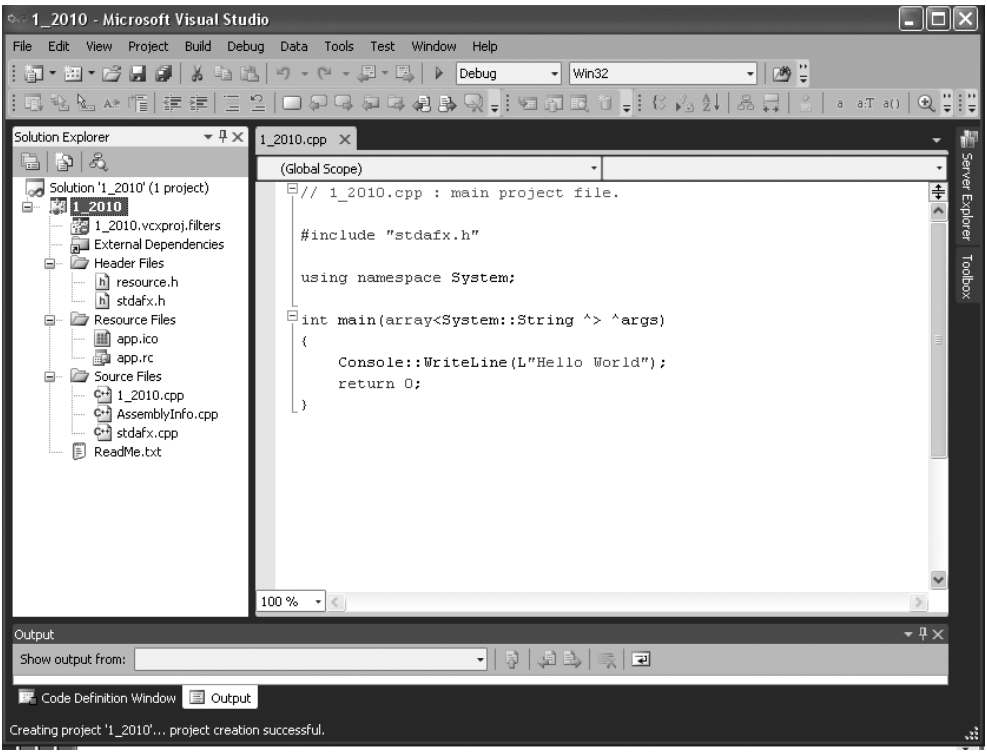


Рис. 1.11. Вид заготовки консольного приложения

5. Преобразуем заголовок функции `main(множество аргументов функции)` к виду `main()`, т. е. к виду без аргументов, 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 мы находимся в Редакторе, щелкнем кнопкой мыши в любом месте поля заготовки и увидим, что курсор установился в месте нашего щелчка (Редактор ждет в этой точке наших дальнейших действий). Далее можно набирать любой текст как в обычном современ-

ном текстовом редакторе, работать клавишами <Delete>, <Backspace>, клавишами-стрелками и другими необходимыми для ввода и редактирования клавишами.

Итак, мы привели заголовок функции `main (аргументы)` к виду `main()`. Это означает, что наша главная функция не будет иметь аргументов, которые служат для связи между собой нескольких консольных приложений. Этим мы заниматься не будем.

Когда мы формировали заготовку консольного приложения, то видели, что при задании имени (**Name**) приложения формировалось и некое поле **Solution** (решение). Дело в том, что среда VC++ оформляет создаваемое приложение в виде двух контейнеров, вложенных один в другой. Один (главный контейнер) называется **Solution** (Решение), а другой — **Project** (Проект). Проект определен как конфигурация (каркас, контейнер), объединяющая группу файлов.

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). Это задается в выпадающем меню, которое по умолчанию установлено на опцию **Debug** (отладка). Выпадающее меню находится в строке окна среды, расположенной ниже строки главного меню.

Проекты являются частью другого каркаса, другого контейнера, который называется **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: "Группа проектов", чтобы термин "Решение" не вводил нас в заблуждение.

Существует специальный инструмент работы с группой проектов, называемый **Solution Explorer**. К нему можно добраться через опцию **View** меню среды разработки. Сама среда автоматически формирует создаваемое приложение как группу проектов, содержащую собственно проект (это видно из рис. 1.12).

Такой подход к оформлению приложения позволяет работать с группой проектов как с одним целым, что ускоряет процесс разработки приложений.

В качестве примера создадим проект с именем 2_2010 таким же способом, как мы создавали проект 1_2010, и добавим его к имеющейся группе 1_2010. При создании нового проекта в нижней части окна **New Project** наряду с по-

лем **Solution Name** ниже кнопки **Browse** (см. рис. 1.9) имеется переключатель **Create directory for solution** (Создать директорию для Решения). Если включить этот переключатель, то будет создана новая группа проектов (**Solution**) с именем, по умолчанию совпадающим с именем создаваемого проекта. Мы включим этот переключатель (при щелчке мышью на нем в его окне появится галочка).

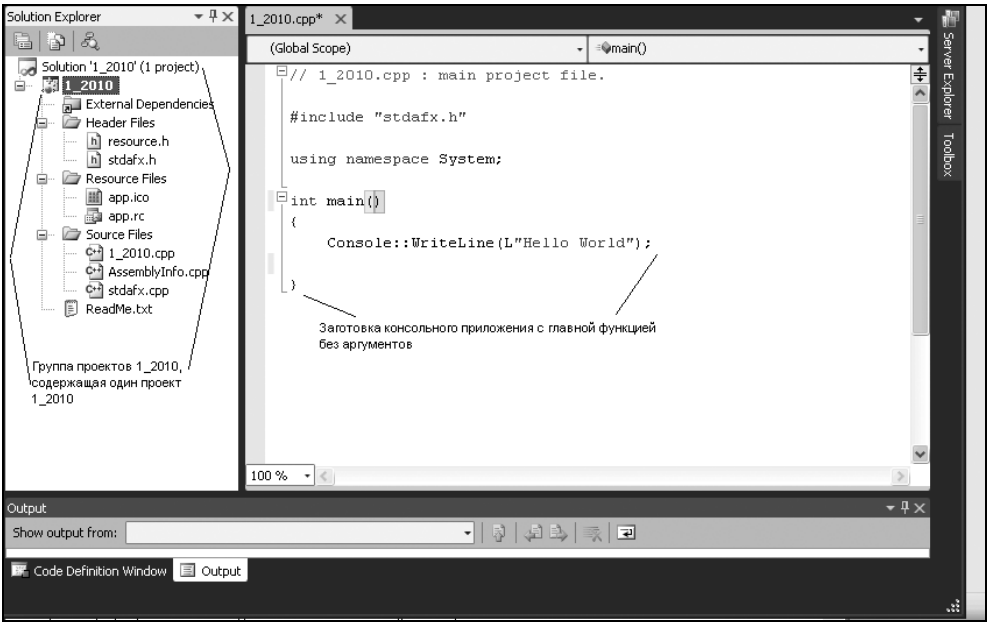


Рис. 1.12. Формирование проекта приложения

Теперь попробуем объединить два проекта в одно Решение (в одну группу проектов). Для этого, во-первых, закроем Решение **2_2010** и откроем Решение **1_2010** (через опцию **File** главного меню), а затем щелкнем правой кнопкой мыши на строке **Solution '1_2010'** и в появившемся контекстном меню выберем команду **Add | Existing Project…** (рис. 1.13).

При этом откроется диалоговое окно для поиска проекта, затем обычным способом откроем проект **2_2010,** в результате чего он добавится к **Solution 1_2010** (рис. 1.14).

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

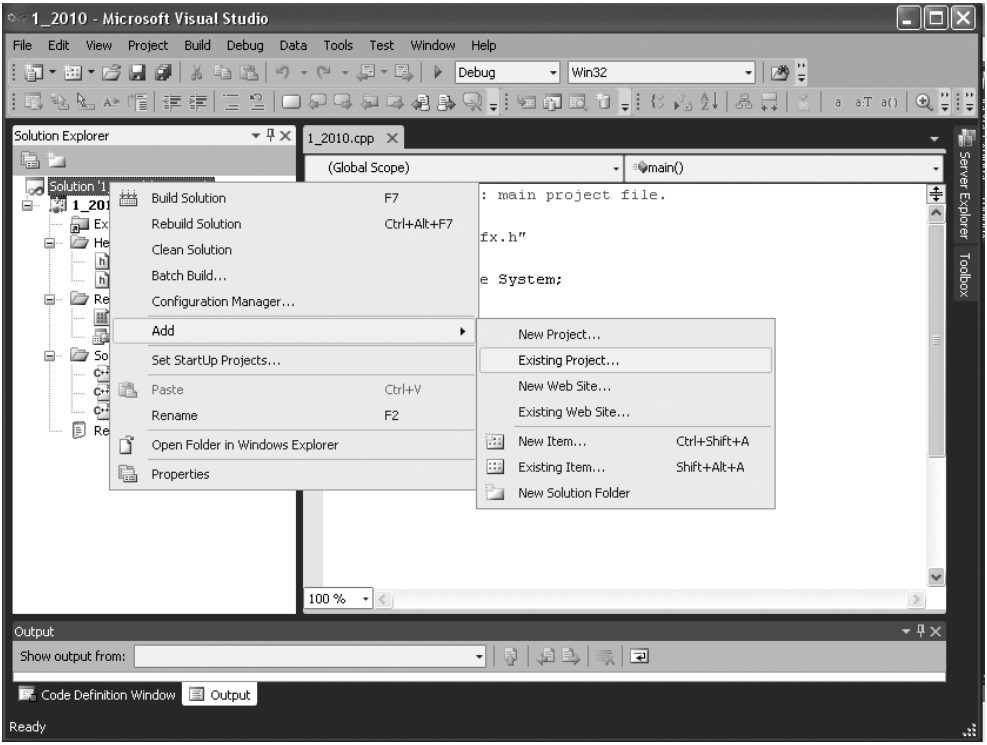


Рис. 1.13. Процесс добавления созданного ранее проекта к группе проектов

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_2010** и оставим в ней проект **1_2010**. Откроем опцию **View** главного меню и выполним в ней подопцию **Start Page**. Получим вид рабочего стола среды, показанный на рис. 1.15.

Из рисунка 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 мышью на его имени. Выполнив команду **Remove** (Удалить проект), мы всего лишь удаляем сведения о проекте из окна **Solution Explorer**, но не из памяти, поэтому-то и возможно восстановление проекта.