Борис Пахомов

bhv®

C# для начинающих

Основные элементы языка C#
Среда программирования SharpDevelop
Создание основных типов приложений

для начинающих

Борис Пахомов

C# для начинающих

Санкт-Петербург
«БХВ-Петербург»
2014

УДК 004.438 C#
ББК 32.973.26-018.1
П12

Пахомов Б. И.

П12 C# для начинающих. — СПб.: БХВ-Петербург, 2014. — 432 с.: ил.

ISBN 978-5-9775-0943-5

Книга является руководством для начинающих по разработке приложений на языке C#. Приведены общие сведения о языке C# и платформе .NET. Рассмотрены базовые типы данных, переменные, функции и массивы. Показана работа с датами и перечислениями. Описаны основные элементы и конструкции языка: классы, интерфейсы, сборки, манифесты, пространства имен, коллекции, обобщения, делегаты, события и др. Приведены сведения о процессах и потоках Windows, а также примеры организации работы в многопоточном режиме. Рассмотрено создание консольных приложений, приложений типа Windows Forms и приложений для работы с базами данных. В качестве среды разработки в книге использован бесплатный пакет SharpDevelop.

Для начинающих программистов

УДК 004.438 C#
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Екатерина Капалыгина*
Редактор	*Анна Кузьмина*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Марины Дамбиевой*

Подписано в печать 30.12.13.
Формат 60×90 1/16. Печать офсетная. Усл. печ. л. 27.
Тираж 700 экз. Заказ №
"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20.

Первая Академическая типография "Наука"
199034, Санкт-Петербург, 9 линия, 12/28

ISBN 978-5-9775-0943-5

Оглавление

Введение

Предлагаемая читателю книга по современному языку C# — результат спонтанного решения автора, долго занимавшегося языком C/C++ и интегрированными средами разработки, такими как Borland C++Builder и Visual C++. Но первой уже нет, а вторая все еще дышит. Но с каждым разом — все реже и реже. Так, по крайней мере, мне кажется. Тут я подвожу к мысли, что не такое уж и спонтанное было мое решение. Все дело в том, что первый звонок прозвенел, когда вышла среда Visual C++ 2008. С удивлением обнаружил, что из среды разработки исчез целый раздел работы с базами данных. Кое-что там, конечно, осталось, но вот основного, увы, не стало. Сколько было вопросов к Microsoft по этому поводу в Интернете! Сколько негодований! Но фирма уклонялась от ответа. Мол, якобы, да, того... Были намеки, что в следующей версии среды все поправится. Нет. Не поправилось ни в следующей (2010), ни в недавней (2012). Стало ясно, что это уже политика фирмы. Пользователей упорно отворачивали от C++, развивая C#. Но так как в мире уже много чего сделано на C++ и в ближайшее даже десятилетие-двадцатилетие придется пользоваться этим языком хотя бы для сопровождения уже наработанного, фирма прозорливо поступает, не отказываясь от выпуска изделий для работы в C++. Но в нем уже столько латок, столько заплаток, столько всего подобного, что порой кажется, что и сами разработчики потеряли контроль над языком. Не зря компания, не бросая разработки по C++, начала развивать более "отточенный" язык C#. Разработчики учли все неприятности, через которые сами проходили и заставили спотыкаться и пользователей-программистов. Я всегда удивлялся, глядя на язык Java. Вроде бы не очень заметный, не очень распространенный, мало рекламируемый, но какой удобный! А фирма, создавшая Java, учла неприятности, заложенные в C++, и избежала их. А теперь настала очередь разработчиков C#: они не потеряли хорошего из C++, взяли замечательное из Java и получили более совершенное из-

делие — C#. По сравнению с C++ он действительно отточенный. Это понимаешь, когда принимаешься за его изучение. Те, кто не мучился с C++, станут уныло изучать C# и не заметят его прелести, как не заметил сначала ее я. Пугают классы. В этом языке даже обычные, казалось, типы переменных, такие привычные как `int`, `float`, `string`, `double`, тоже, оказываются, классы! Но, слава Богу, для этих классов введены ключевые слова, которые я только что написал, и поэтому, не зная классов, можно вначале работать с такими типами как бы по-старому, не пугаясь.

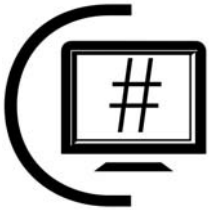
Материал книги построен так, чтобы не попадать сразу в неведомое новое, хотя бы тем, кто раньше изучал языки программирования. Но не всегда удается это сделать. Поэтому на определенных этапах изучения придется кое-что принимать просто на веру, а потом уже в дальнейшем материале видеть, откуда что бралось и почему.

В книге много примеров, которые надо не только разбирать (хотя в них есть комментарии), но и желательно самому их записывать, а не скачивать сразу в приложение, если представится такая возможность. Один читатель моих предыдущих книг как-то прислал мне письмо, что, мол, неудобно из книги заводить себе примеры, составляя из них свое приложение. Муторно. Нельзя ли, мол... Нельзя. Я не сторонник. Когда вы вручную заносите текст в поле редактора, вы одновременно изучаете язык, сами того не подозревая. Пробуете его на вкус. Запоминаете его правила и тем самым избегаете в будущем большого количества ошибок при вводе текста. Хотя сегодняшние компиляторы большинство ошибок быстро отлавливают. Но вот как раз в этот-то момент и происходит ваше близкое знакомство с языком, потому что компилятор вам подсказывает, мол, так нельзя, надо вот так. А если вы отлаженный текст вычлените из книги и вставите его в поле редактора, то ничего этого не увидите и в дальнейшем окажетесь беспомощным при вводе собственного текста.

Чем еще интересен C#? Пользуясь его средствами, вы можете решать проблемы создания графических интерфейсов, не прибегая непосредственно к таким средам, как Windows Forms, ADO.NET и др. Хотя это не рационально, потому что, пользуясь этими средами напрямую, вы пользуетесь теми сервисами, которые в них уже заложены. В C# надо будет делать все вручную. Но сделав хотя бы один примерчик для одной из сред, вы при изучении отдельно этих сред увидите и поймете, почему там в них все так устроено. Такой пример приведен в последней, 18-й главе книги для среды Windows Forms.

Приятного изучения C#.

Автор



ЧАСТЬ I

Базовые сведения о программировании на языке C#

ГЛАВА 1

Общие сведения о языке C# и платформе .NET

.NET (читается "дот нэт") или .NET Framework — это платформа программирования. Вообще, компьютерная платформа — это аппаратный и/или программный комплекс, служащий основой для различных вычислительных систем. Примером платформы программирования может служить операционная система компьютера. Алгоритмический язык C# (читается "си шарп") как раз и создан для работы на платформе .NET.

Разработка программного обеспечения (ПО) на платформах операционных систем (ОС) семейства Windows подразумевала использование языка программирования C (читается "си" в соответствии с английской фонетикой) в сочетании со специальными средствами ОС Windows, которые называются сокращенно API (читается не по буквам "эй пи ай", а "апи"). Это аббревиатура от Application Programming Interface — интерфейс прикладного программирования. В этом интерфейсе сосредоточены крупные программные структуры, позволяющие путем их настройки на конкретное приложение автоматизировать процесс трудоемкого программирования на C. Тот, кому "повезло" испробовать на себе это "удовольствие", думаю, до сих пор видит по ночам кошмарные сны. Но это и понятно: все вновь созданное обычно очень несовершенно и дорабатывается в процессе длительной эксплуатации. Необходимость уйти от использования напрямую в программировании средств API привела к созданию более совершенных систем программирования типа, например, Borland C++Builder, которые значительно облегчили и облагородили тяжелый труд программиста. Однако жизнь не стоит на месте, и язык C на определенном этапе перестал обеспечивать потребности программирования. На горизонте появилась концепция так называемого *объектно-ориентированного программирования* (ООП), которая позволяла посмотреть на сам процесс создания программного продукта со-

всем с другой стороны, предоставляя программисту более широкие возможности для автоматизации его труда и создания более качественной программной продукции. Основой ООП явились понятия *класса* и *объекта*. Разработчики языка C пошли путем добавки к C структуры "класс". Получился язык C++. Этот процесс оказался настолько непростым, что, думаю, в свое время сами разработчики очень пожалели, что приняли именно такую концепцию быть на уровне современных требований к процессу создания программного продукта. В погоне за скоростью обработки приложениями данных и за необходимой надежностью и безопасностью работы приложений разработчикам пришлось организовывать два вида памяти при обработке данных: неуправляемую (в C памятью приходится управлять вручную) и управляемую (в C++ эту функцию берет на себя специальная среда, так называемая *управляемая куча*, поэтому управление памятью — автоматическое), организовывать специальный и довольно неприятный аппарат указателей. Но мы знаем, что чем дальше в лес, тем больше дров. Разработчикам пришлось строить аппарат перехода между данными из управляемой памяти в неуправляемую и наоборот. Легче было похоронить C и создать заново другой язык на новой концепции. Но разработчики были связаны по рукам: очень много программного продукта на C уже работало в мире, и поставить на нем крест значило подорвать производственный процесс множества предприятий и организаций. Поэтому приходилось не только заботиться о сохранении C, но и придерживаться современных требований (создание C++), поддерживать совместимость старых программ при работе в новых средах. То есть надо было тащить за собой хвосты C в новый язык C++, которые только мешали новому языку и осложняли процесс разработки программ на этом языке. В конце концов, видимо, у разработчиков терпение лопнуло, и они создали новый язык под названием C#, учитывающий новые веяния в программировании (ООП) и свободный от недостатков C++. Однако и C++ не оказался заброшенным по причине, отмеченной ранее (совместимость и поддержка уже работающих в мире программ). Да и большое количество программистов, работающих на C++, не очень жаждут изучать новый язык, зная, что переход на более высокий уровень всегда есть шаг назад на некоторое время. В заключение своего пассажа на тему старых-новых программ приведу пример из собственного наблюдения. Одна испанская транснациональная компания, приобретя предприятие, на котором я работал, стала внедрять, что вполне естественно, свою технологию (передовую по тем временам) управления производственным процессом. Привезла с собой свои программы, которые у нее давно работали в других ее "дочках". Оказалось, что многие из программ написаны на языке КОБОЛ, о котором мы забыли еще лет пятнадцать назад.

Но руководство не собиралось из-за наших принципов терять свои деньги и приказало вспомнить КОБОЛ для сопровождения старых программ. Думаю, что я убедил читателя в необходимости создания C# и, тем более, в необходимости его изучения. Замечу также, что C# — это язык семейства языков C, он является гибридом языков C, Java, Visual Basic 6. Следуя за М. В. Ломоносовым, сказавшем о русском языке, что он содержит в себе "великолепие ишпанского, живость французского, крепость немецкого, нежность итальянского, сверх того богатство и сильную в изображениях краткость греческого и латинского языка", про C# можно сказать, что он с синтаксической точки зрения является таким же чистым, как Java, столь же простым, как Visual Basic 6, и таким же гибким и мощным, как C++. Если установить бесплатный продукт фирмы Microsoft .NET 4.0 Framework Software Development Kit (SDK) или среду Visual Studio 2010, то для программирования на основе платформы .NET становятся доступными языки C#, F#, JScript .NET, Visual Basic, C++/CLI. Здесь CLI (Common Language Infrastructure, общеязыковая инфраструктура) — привязка C++ к платформе .NET. Вернемся все-таки к платформе .NET, на базе которой функционирует C#. Эта платформа представляет собой программную платформу для создания приложений не только на базе ОС семейства Windows, но и других операционных систем, которые создавались не фирмой Microsoft, как Windows. Это системы Mac OS X, UNIX, Linux. Платформа обеспечивает взаимодействие с уже существующим программным обеспечением. Приложения на платформе .NET можно создавать с помощью многих языков программирования, таких как C#, F#, S#, Visual Basic и др. Сегодня фирма Microsoft выпускает продукт под названием Visual Studio (2008, 2010, 2012), который дает возможность создавать приложения на разных языках на платформе .NET. Все языки, поддерживаемые .NET, имеют общий исполняющий механизм. Здесь уже нет такой неразберихи, как в C++ (управляемая и неуправляемая память, разные указатели для обоих видов памяти, аппарат перехода от одного вида памяти к другому). Платформа содержит в себе обширную и, что важно, общую для всех поддерживаемых языков библиотеку базовых классов, которые обеспечивают, например, ввод-вывод данных, работу приложений с графическими объектами, создание не только веб-интерфейсов, но и обычных (настольных) и консольных (без графики) приложений, работу с базами данных, дают возможность создавать интерфейсы для работы с удаленными объектами. В частности, платформа .NET Framework — это управляемая среда выполнения, предоставляющая разнообразные службы работающим в ней приложениям. Она состоит из двух основных компонентов: исполняющей среды общего языка (Common Language Runtime, CLR), являющейся механизмом, управляющим выполняющие-

ся приложения, и библиотеки классов .NET Framework, предоставляющей библиотеку проверенного кода, предназначенного для повторного использования, который разработчики могут вызывать из своих приложений. Службы (точнее — сервисы, а еще точнее — услуги), которые платформа .NET Framework предоставляет работающим приложениям:

- *управление памятью.* Во многих языках программирования разработчики самостоятельно назначают и выделяют ресурсы памяти и решают вопросы, связанные со временем жизни объектов. В приложениях платформы .NET Framework среда CLR предоставляет эти сервисы автоматически;
- *система общего типа.* В традиционных языках программирования базовые типы определяются компилятором, что осложняет взаимодействие между языками. В платформе .NET Framework базовые типы определяются единственной системой типа .NET Framework, называемой CTS (Common Type System). При этом используются одни и те же базовые типы для всех языков .NET Framework;
- *расширенная библиотека классов.* Вместо того чтобы писать много кода для выполнения стандартных низкоуровневых операций программирования, разработчики могут использовать легкодоступную библиотеку типов и члены из библиотеки классов .NET Framework;
- *платформы и технологии разработки.* Платформа .NET Framework включает библиотеки для конкретных областей разработки приложений, например ASP.NET для веб-приложений, ADO.NET для доступа к данным и Windows Communication Foundation для приложений, ориентированных на службы (сервисы);
- *взаимодействие языков.* Языковые компиляторы на платформе .NET Framework компилируют приложение не в исполнительный код сразу, а в промежуточный код, называемый языком CIL (Common Intermediate Language), который впоследствии компилируется во время исполнения приложения средой CLR. Такой подход приводит к тому, что программы, написанные на одном языке, доступны в других языках, а разработчики могут сосредоточиться на создании приложений на предпочитаемом языке или языках;
- *совместимость версий.* За редкими исключениями, приложения, которые разрабатываются с помощью платформы .NET Framework определенной версии, могут выполняться без изменений на более поздней версии;
- *параллельное выполнение.* Платформа .NET Framework помогает в разрешении конфликтов версий, разрешая установку нескольких

версий среды CLR на одном компьютере. Это означает, что несколько версий приложений также могут сосуществовать, и что приложение может выполняться на версии платформы .NET Framework, для которой оно было создано;

- *настройка для различных версий.* Ориентируясь на переносимую библиотеку классов платформы .NET Framework, разработчики могут создавать сборки (exe- или dll-файлы, предназначенные для исполнения), которые работают на нескольких платформах .NET Framework. Например, на .NET Framework, Silverlight, Windows Phone 7 или Xbox 360.

.NET Framework для пользователей

Если вы не разрабатываете приложения .NET Framework, но используете их, вам не требуется обладать какими-либо специальными знаниями о платформе .NET Framework или ее работе.

Если используется операционная система Windows, платформа .NET Framework может быть уже установлена на компьютере. Кроме того, если устанавливается приложение, требующее платформу .NET Framework, программа установки приложения может инсталлировать конкретную версию .NET Framework на вашем компьютере. В некоторых случаях можно увидеть диалоговое окно, которое запрашивает установку платформы .NET Framework.

Как правило, не требуется удалять какие-либо версии .NET Framework, уже установленные на вашем компьютере, потому что используемое приложение может зависеть от конкретной версии. В случае удаления какой-либо версии его исполнение может завершиться ошибкой. Обратите внимание, что на одном компьютере может быть одновременно загружено несколько версий платформы .NET Framework. Это означает, что не нужно удалять предыдущие версии для установки более поздней версии.

.NET Framework для разработчиков

Разработчик может выбрать любой язык программирования, который поддерживает платформу .NET Framework, для создания приложения. Поскольку платформа .NET Framework обеспечивает независимость и взаимодействие языков, можно взаимодействовать с другими приложениями и компонентами платформы .NET Framework независимо от языка, с помощью которого они были разработаны.

Для разработки приложений или компонентов платформы .NET Framework выполните следующие действия:

1. Установите версию платформы .NET Framework, на которую будет нацелено ваше приложение. Последняя рабочая версия — это .NET Framework 4.5.
2. Выберите язык или языки платформы .NET Framework, которые вы будете использовать для разработки приложений. Доступны языки Visual Basic, C#, F# и C++ от Microsoft. Язык программирования, который позволяет разрабатывать приложения для платформы .NET Framework, соответствует спецификации Common Language Infrastructure (CLI). В спецификации описываются требования к исполняемому коду приложения и среде исполнения этого кода. Определяется среда, позволяющая многим языкам высокого уровня использоваться на различных компьютерных платформах. Иначе говоря, если ваша программа создана на одном из языков из платформы .NET, то она сможет работать на другом компьютере с другой платформой (не на любой, конечно, а на той, что согласована с .NET).
3. Выберите и установите среду разработки, которая будет использоваться для создания приложений и которая поддерживает выбранный вами язык программирования. Интегрированная среда разработки (Integrated Development Environment, IDE) Microsoft для приложений .NET Framework — это Microsoft Visual Studio, например, версии 2010. Она доступна бесплатно в версии Express. Но существует и другой бесплатный продукт, на котором можно создавать приложения в среде .NET. О нем будет рассказано в *главе 2*.

ГЛАВА 2

Средства создания приложений на языке C#

Описание средств

Чтобы создавать приложения на C#, как и на любом другом языке программирования, надо иметь возможность записать в файл (или в файлы) сам текст программы, а затем этот текст откомпилировать с помощью компилятора, создавая исполняемый файл, при запуске которого на выполнение получается результат работы разработанного приложения. Какие же средства можно использовать для создания приложений на C#?

Простейшим средством для записи и сохранения текста приложения являются текстовые редакторы WordPad и Блокнот. С их помощью можно записать текст приложения на C# и при сохранении текста дать этому тексту расширение cs ("си шарп"). На рис. 2.1 и 2.2 показаны фрагмент текста C#-приложения в WordPad и окно редактора в момент сохранения текста программы.

Далее следует сохраненный текст откомпилировать. Где взять компилятор? Если у вас имеется интегрированная среда разработки приложений Visual Studio 2010 или 2011 (у автора на момент написания этой главы была версия 2011), то в этой среде в главном меню **Tools** надо выбрать команду **Visual Studio Command Prompt** (запуск exe-файлов из командной строки). В результате на экране появится консольное окно, предлагающее вводить исполняемые файлы. Компилятор C# имеет исполняемый файл, названный csc.exe. Наберите имя компилятора в командной строке и нажмите клавишу <Enter>. Вы увидите, что среда требует указать компилируемый файл. Чтобы не писать длинные пути к искомому файлу в консольном окне (капризном, т. к. у него мало возможностей редактирования текста), можно сначала при сохранении в

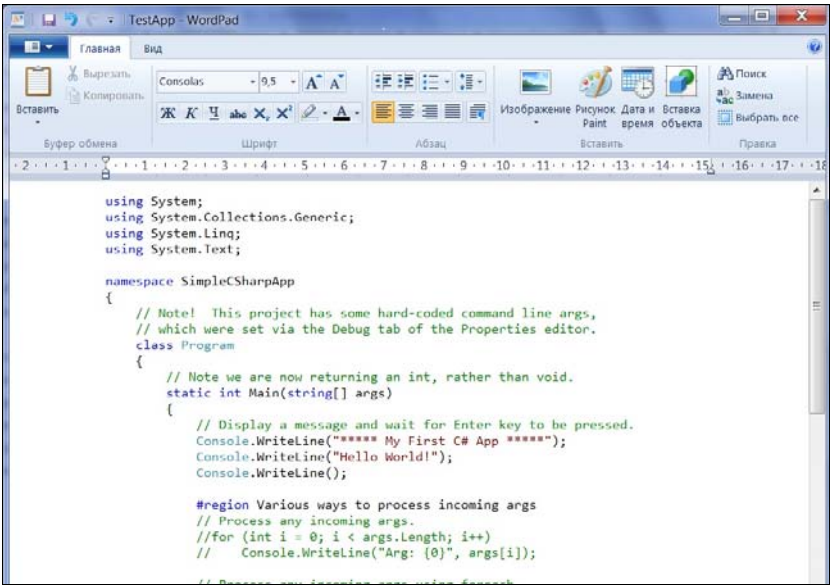


Рис. 2.1. Фрагмент текста приложения в WordPad

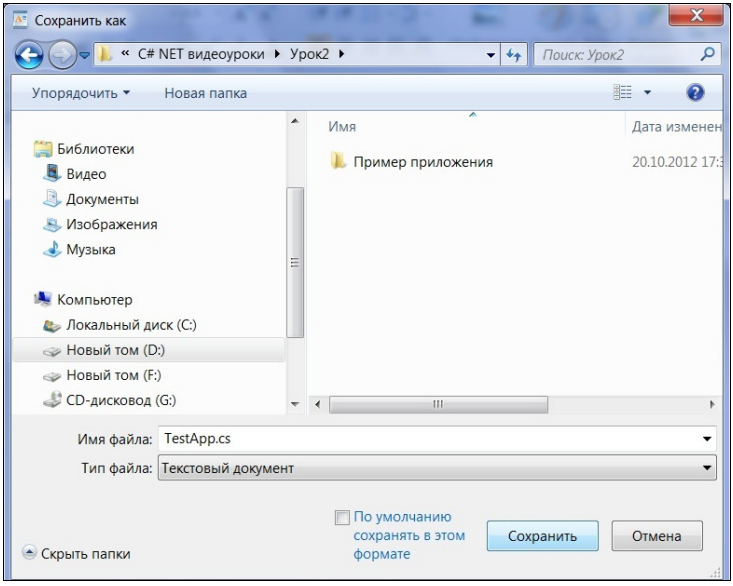


Рис. 2.2. Сохранение текста приложения редактором WordPad

WordPad поместить файл в корневой каталог, в котором находится среда обработки. У автора среда находится в одной из папок на устройстве D:. Поэтому для простоты общения с консольным окном я стану сохранять свои приложения в WordPad в каталоге D:\. Приложение было названо TestApp.cs (рис. 2.2). Переместите его в D:\, теперь попробуйте его откомпилировать. Этот процесс показан на рис. 2.3.

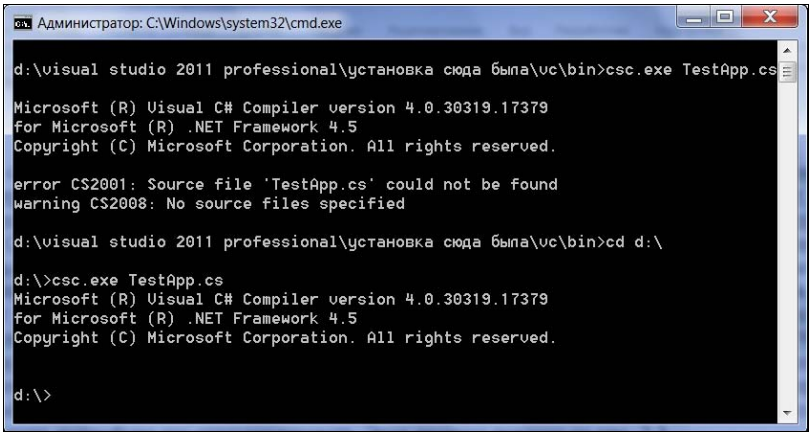


Рис. 2.3. Компиляция из командной строки приложения C#

Из рис. 2.3 видно, что сначала компилятор не нашел исходного файла для компиляции, т. к. файл был перемещен в корневой каталог. Поэтому в окне надо было сначала выполнить команду MS-DOS "Перейти в корневой каталог D:\". Эта команда — `cd` (от англ. *change directory*). После перехода в корневой каталог компилятор нашел исходный файл, и компиляция прошла успешно. Результат компиляции (файл с тем же именем, но с расширением exe) помещен в тот же каталог, что и исходный файл.

Компиляцию исходного кода на C# можно произвести, имея только всем доступную платформу Microsoft .NET Framework, которую можно загрузить бесплатно. Компилятор для C#, csc.exe (C-Sharp Compiler), входит в эту среду. Конечно, большие приложения на нем компилировать будет достаточно проблематично, но все же порой полезно знать, как это сделать. Перед тем как воспользоваться компилятором, нужно его настроить. Для того чтобы проверить, находит ли ваша операционная система файл csc.exe, введите его в командную строку `csc /?`. В ответ должен появиться список опций настройки, поддерживаемых компилятором C#. С командной строкой в разных операционных систе-

мах тоже могут быть проблемы. Если в ОС Windows XP командная строка сразу видна в меню кнопки **Пуск**, то этого нельзя сказать об ОС Windows 7. Там, чтобы добраться до командной строки, надо нажать кнопку **Пуск**, а затем в поле поиска в нижней части меню кнопки (в самом поле виден текст "Найти программы и файлы") надо ввести слово `команд`. Среди множества найденных строк вы увидите и значок командной строки с названием **Командная строка**. Щелкнув по значку, на экране увидите консольное окно с текстом и приглашение вводить данные (рис. 2.4).

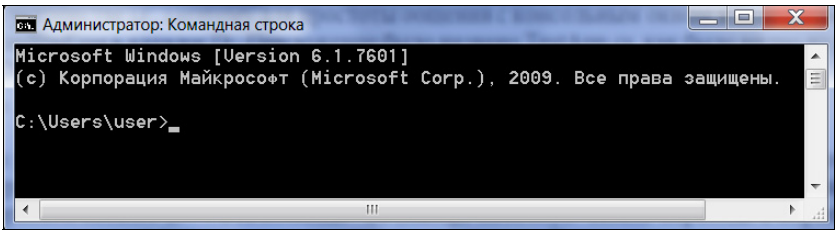


Рис. 2.4. Консольное окно командной строки операционной системы

Если компилятор не находится, значит, в системе не прописан к нему путь. Чтобы прописать путь, нужно щелкнуть правой кнопкой мыши на **Мой компьютер** (в Windows XP), выбрать опцию **Свойства**. Если у вас Windows XP, то нужно выбрать опцию **Дополнительно** (в Windows 7 — ссылка слева **Дополнительные параметры системы**) и щелкнуть по кнопке **Переменные среды** в нижней части открывшегося окна **Свойства системы**. Откроется диалоговое окно **Переменные среды**, в котором в прокручиваемом поле **Системные переменные** найдите переменную `Path`, щелкните на ней мышью, нажмите кнопку **Изменить** под этим полем. Откроется диалоговое окно с содержимым переменной `Path` для ее корректировки. Вам необходимо дописать к концу значения переменной точку с запятой и путь к размещению .NET Framework SDK. Обычно это C:\Windows\Microsoft.NET\Framework\v3.5, но версию лучше уточнить, войдя в папку C:\Windows\Microsoft.NET\Framework.

Для создания C#-приложения можно воспользоваться возможностями какой-либо интегрированной среды типа Visual Studio. Если вы — начинающий программист, то у вас может не оказаться такой среды, а пока вы ее приобретете, у вас пропадет охота изучать язык. Лучше воспользоваться специально разработанным и бесплатным продуктом Microsoft для изучения C# под названием SharpDevelop.

SharpDevelop можно бесплатно скачать и установить у себя на компьютере, зайдя на интернет-страницу по адресу **www.sharpdevelop.com**. На

этой странице надо перейти на вкладку **SharpDevelop** в верхней части страницы и в открывшемся окне выбрать вкладку **Download**. Во вновь открывшемся окне щелкнуть по ссылке **Downloads for SharpDevelop 4.3**.

Далее мы рассмотрим, как пользоваться этой средой для создания C#-приложений. Отметим сразу следующее: те, кто работал в средах Visual Studio, найдут в новой среде много сходств.

Интегрированная среда SharpDevelop для создания приложений на языке C#

Общий вид главного окна SharpDevelop показан на рис. 2.5.

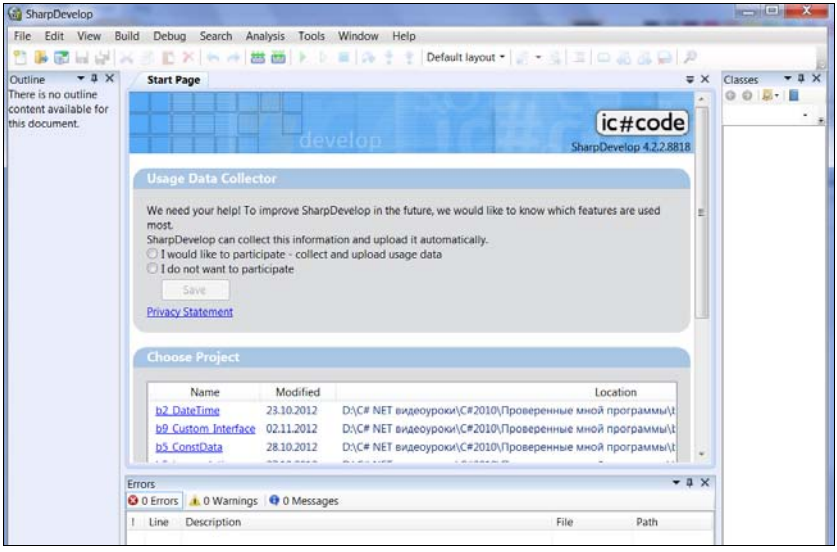


Рис. 2.5. Общий вид главного окна SharpDevelop

В центре окна находится стартовая страница (**Start Page**), в разных частях которой отражаются такие элементы, как Usage Data Collector — сведения об использовании SharpDevelop разработчиками. Сведения собираются фирмой, создавшей продукт SharpDevelop (так называемая обратная связь, служащая для возможного совершенствования выпу-

щенного продукта на основе мнений его пользователей). Ниже расположено окно **Choose Project**, в котором отображаются проекты, выполненные ранее и доступные для повторного вызова через это окно. Для этого достаточно дважды щелкнуть на соответствующей строке, и нужный проект появится в своем окне на экране, готовый для обработки. Но можно и просто один раз щелкнуть по названию проекта и затем нажать кнопку **Open solution**, находящуюся ниже этого окна. Если ее не видно, надо мышью протянуть ползунок окна вниз (ползунок прокрутки окна). Это показано на рис. 2.6.

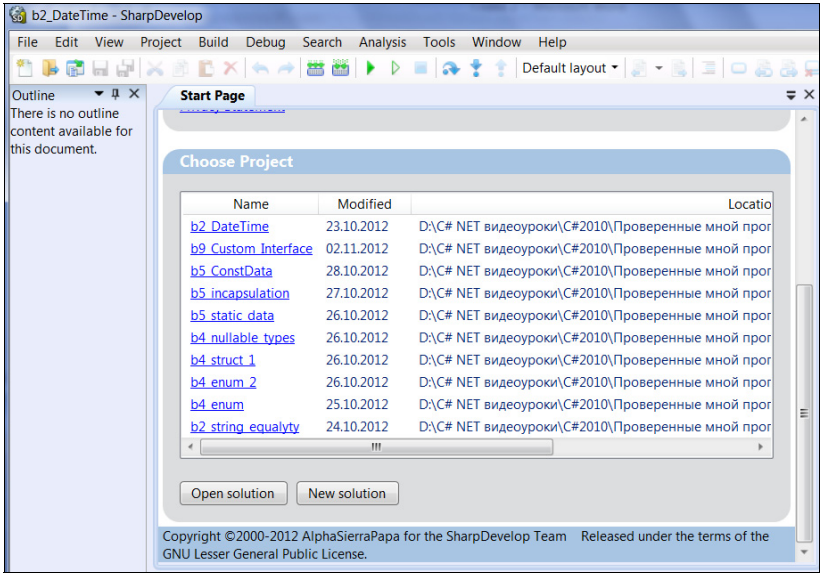


Рис. 2.6. Прокрутка окна **Choose Project**

Ниже окна **Start Page** находится окно **Errors** (Ошибки). В это окно компилятор выводит сведения об ошибках компиляции. Фразы "окно выше", "окно ниже", "окно слева", "окно справа" весьма относительно указывают о расположении окон. Дело в том, в SharpDevelop, как и в последних версиях Visual Studio, окна снабжены свойствами, позволяющими им сворачиваться (делаться невидимыми), перемещаться по экрану с помощью протягивания мышью за заголовок окна, захватывать друг друга при перемещении, когда одно окно попадает в поле захвата другого окна. Эти захваты называются *причаливанием* (Dock), подобно причаливанию судов в доках порта. Порт как бы захватывает или отпускает суда. Так и здесь с окнами. Только роль порта играет окно, которое

не перемещается, а роль судна — перемещаемое окно. Перечень свойств окна можно увидеть, как и свойств любого объекта Windows, щелкнув на самом объекте правой кнопкой мыши. Только для окон надо устанавливать курсор мыши на заголовок окна, а уже потом щелкать правой кнопкой. Свойства окна показаны на рис. 2.7.

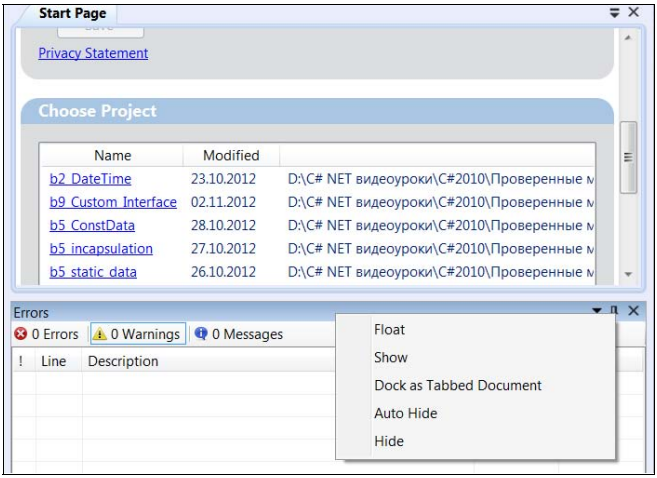


Рис. 2.7. Свойства окна в SharpDevelop

Посмотрим, какой вид приобретает окно, когда мы выбираем то или иное его свойство.

- **Float**. Это свойство обеспечивает окну "свободное плавание". Вы можете перетаскивать мышью окно в любое место экрана, и, когда отпустите кнопку мыши (протяжка идет ведь при постоянно нажатой кнопке), окно останется на том месте, где случилось отпускание кнопки мыши. Но при движении оно может быть захвачено другим окном, если попадет в область захвата этого окна.
- **Hide**. Окно исчезает с экрана — становится невидимым. Чтобы оно снова появилось на экране, надо воспользоваться пунктом **View** главного меню среды. Главное меню среды расположено в виде опций в самой верхней строке окна среды (рис. 2.8).

  Пункт меню **View** содержит, кроме прочего, имена окон среды. Если щелкнуть на соответствующем имени, окно станет видимым на экране. Например, вы можете закрыть стартовую страницу, если она вам мешает, а потом при необходимости снова ее показать, воспользовавшись командой **View | Show start page**.

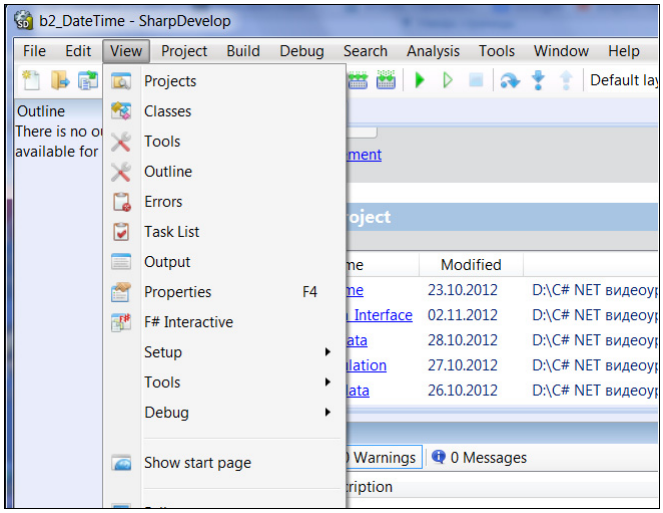


Рис. 2.8. Вид главного меню SharpDevelop и команд меню **View**

- **Dock as Tabbed Document**. Это свойство окна, если оно выбрано, автоматически заставляет окно причалить к главному окну среды в качестве его очередной вкладки (рис. 2.9).

Остальные свойства окна рассматривать не будем.

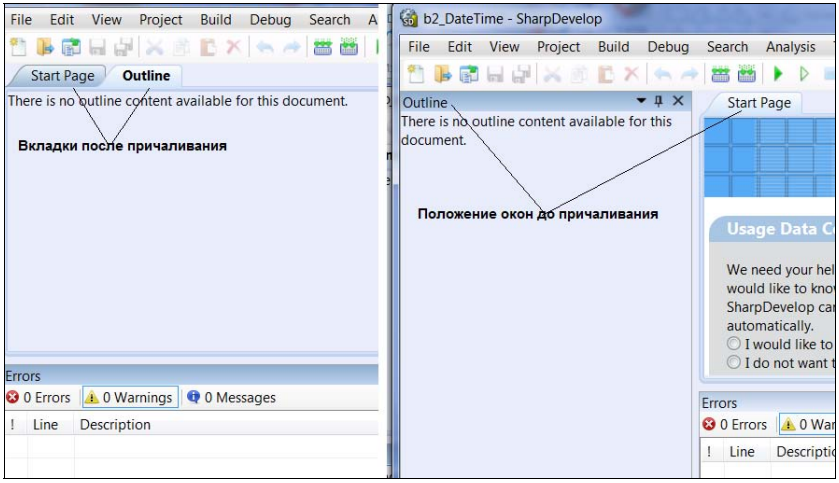


Рис. 2.9. Причаливание окна со свойством **Dock as Tabbed Document**

Теперь посмотрите, как на экране проявляются области захвата, когда мы протягиваем некоторое окно. Возьмите любое из окон рабочего стола SharpDevelop, например то, которое только что в предыдущем свойстве причаливали, а потом установили на прежнее место (окно **Outline** из рис. 2.9). Зацепите его (за его заголовок) мышью и начните перетаскивать вправо. Как только вы установите курсор на заголовок окна и нажмете левую кнопку мыши, готовясь к протяжке окна, на рабочем столе среды тут же появятся указатели — индикаторы областей захвата (рис. 2.10).

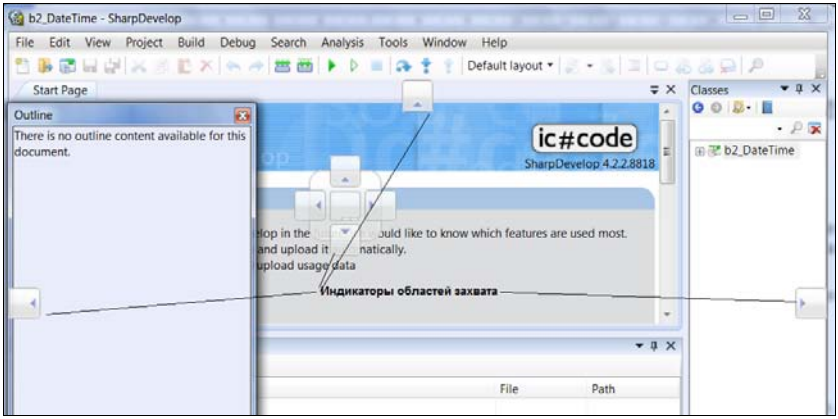


Рис. 2.10. Указатели областей захвата рабочего стола SharpDevelop

Причем, появляются не все, что показаны на рис. 2.10. По мере протяжки появляются и другие. Те индикаторы, в чье поле действия попадает протягиваемое окно, изменяют свой цвет с белого на голубой (рис. 2.11).

Если теперь отпустить левую кнопку мыши, то произойдет захват окна той областью, на которую указал подсвеченный индикатор (рис. 2.12).

Теперь попробуйте поставить окно на его прежнее место. Для этого опять зацепите курсором мыши заголовок окна и начните его тянуть медленно на прежнее место. Вы увидите, что вскоре слева на рабочем столе среды разработки появится темная вертикальная полоса, тоже показывающая область захвата окна, но уже не в виде прямоугольников, как раньше (рис. 2.13).

Если теперь отпустите левую кнопку мыши, прекратив тем самым протяжку окна, и обратите внимание, что окно встало на свое прежнее место.

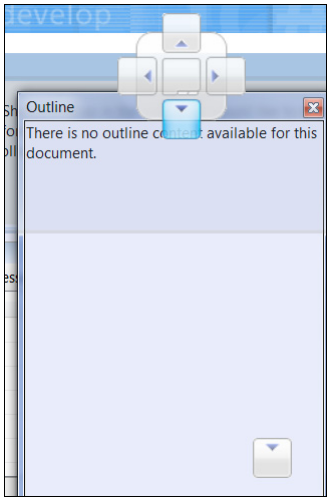


Рис. 2.11. Активизация области захвата на рабочем столе при протягивании окна

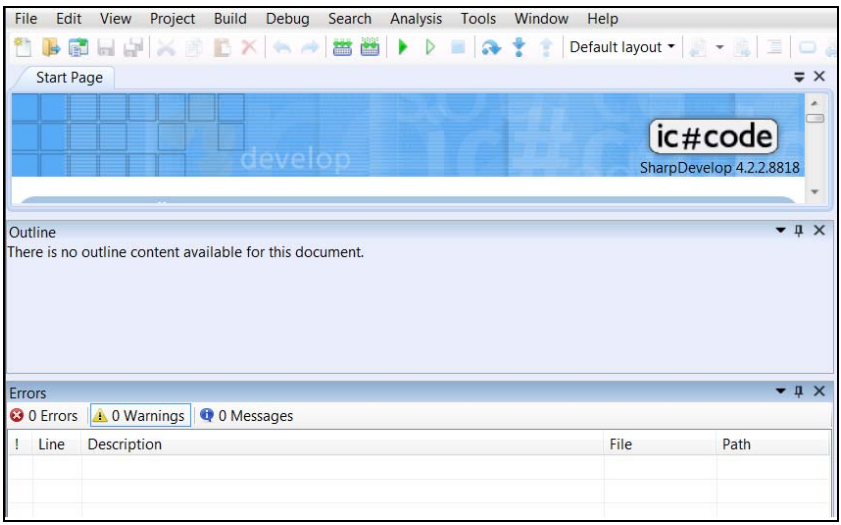


Рис. 2.12. Результат захвата окна

Далее вы самостоятельно можете поэкспериментировать с перемещением окон в нужное для вас место. При этом надо учесть, что если одно окно захватывает другое, то оно помещает захватываемое окно внутрь себя в виде вкладки (становится видна только вкладка с названием окна, возможно, и неполным из-за недостатка места). Окно, помещенное внутрь другого окна, можно снова вытащить, захватив курсором мыши

за вкладку и потянув окно в сторону. Однако перемещениями окон, особенно на начальном этапе работы с SharpDevelop, увлекаться не стоит: можно такого нагородить, что потом вообще станет ничего не понятно. Придется закрывать все окна и затем по одному открывать с помощью меню **View** и настраивать на нужное месторасположение.

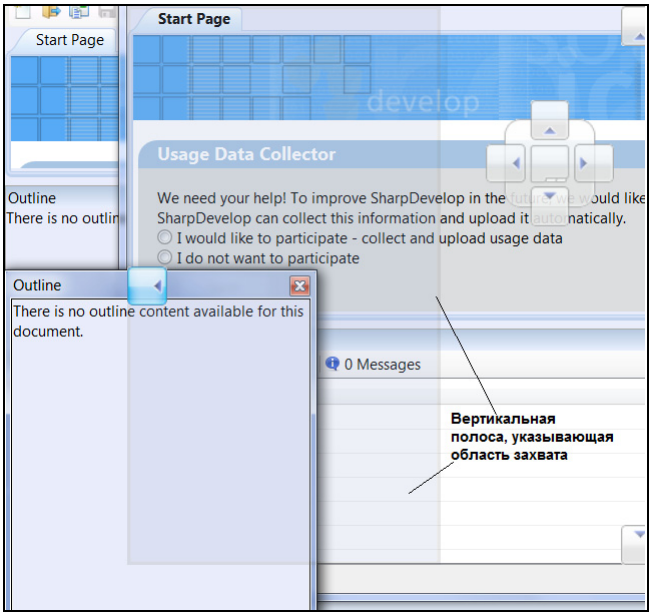


Рис. 2.13. Область захвата в виде вертикальной прямоугольной полосы рабочего стола

Суть остальных окон, возможно, и не всех, мы рассмотрим по мере создания C#-приложений, на примерах которых станем изучать сам язык.

Как создавать приложения в рамках SharpDevelop? Здесь принята та же система, что и в последних версиях Visual Studio: приложения оформляются в виде структур, которые называются *решениями* — solution. Решение состоит из нескольких проектов (project), что дает возможность формировать приложение из нескольких проектов, подключая их к данному решению, запускать приложение из некоторого проекта, делая его стартовым (см. меню **Project**). Кроме того, к проекту можно добавлять и отдельные файлы, расширяющие функциональность проекта. Этими сложными образованиями мы заниматься не станем, т. к. у нас цель другая: изучить C# на таком уровне, чтобы можно было начинать строить на нем приложения, хотя бы не очень сложные. Поэтому, создавая любое приложение, начинают с создания нового решения, которое