

C# для начинающих

Основные элементы языка C#

Среда программирования SharpDevelop

Создание основных типов приложений



ДЛЯ НАЧИНАЮЩИХ



УДК 004.438 С#
ББК 32.973.26-018.1
П12

Пахомов Б. И.

П12 С# 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 С#. Приведены общие сведения о языке С# 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 С#
ББК 32.973.26-018.1

Группа подготовки издания:

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

Подписано в печать 30.12.13.
Формат 60×90¹/₁₆. 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

© Пахомов Б. И., 2014
© Оформление, издательство "БХВ-Петербург", 2014

Оглавление

Введение.....	9
---------------	---

ЧАСТЬ I. БАЗОВЫЕ СВЕДЕНИЯ О ПРОГРАММИРОВАНИИ НА ЯЗЫКЕ C#	11
---	-----------

Глава 1. Общие сведения о языке C# и платформе .NET	13
--	-----------

.NET Framework для пользователей	17
--	----

.NET Framework для разработчиков.....	17
---------------------------------------	----

Глава 2. Средства создания приложений на языке C#.....	19
---	-----------

Описание средств.....	19
-----------------------	----

Интегрированная среда SharpDevelop для создания приложений

на языке C#.....	23
------------------	----

Глава 3. Базовые типы данных, переменные.....	33
--	-----------

Переменные.....	37
-----------------	----

Тип целочисленных данных.....	38
-------------------------------	----

Тип данных с плавающей точкой.....	40
------------------------------------	----

Десятичный тип данных.....	41
----------------------------	----

Первые программы.....	43
-----------------------	----

Логический тип данных.....	49
----------------------------	----

Оператор <i>for</i>	50
---------------------------	----

Символьные типы данных.....	54
-----------------------------	----

Тип <i>char</i>	55
-----------------------	----

Тип <i>string</i>	60
-------------------------	----

Программы работы с переменными типа <i>string</i>	62
---	----

Программа для проверки некоторых базовых функций работы со строками	63
--	----

Программа копирования символьного файла.....	65
Ввод текста.....	66
Подсчет количества введенных строк.....	68
Подсчет количества слов в тексте.....	70
Тип <i>var</i>	72
Некоторые обобщения по объявлению и работе с переменными.....	73
Объявление констант.....	73
О преобразовании данных разных типов.....	74
Арифметические действия.....	76
Простые операторы.....	76
Порядок выполнения арифметических операторов.....	78
Оператор присваивания.....	78
Операторы инкремента и декремента.....	79
Операторы сравнения.....	79
Логические операторы.....	80
Операторы сдвига.....	82
Глава 4. Функции.....	85
Создание некоторых функций.....	90
Оператор <i>if</i>	93
Оператор <i>goto</i>	94
Функция выделения подстроки из строки.....	94
Функция копирования строки в строку.....	97
Функция с выходными параметрами.....	100
Переключатель <i>switch</i>	102
Область действия переменных.....	105
Рекурсивные функции.....	106
Глава 5. Массивы.....	107
Одномерные массивы.....	107
Оператор <i>foreach</i>	111
Многомерные массивы.....	113
Глава 6. Еще раз о функциях консольного ввода-вывода.....	115
Ввод.....	115
Вывод.....	116
Глава 7. Работа с датами и перечислениями.....	121
Даты.....	121
Форматный вывод дат.....	122
Операции с датами.....	125

Перечисления	128
Типы перечислений как битовые флаги.....	133

ЧАСТЬ II. ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ135

Глава 8. Введение в классы137

Ключевое слово <i>this</i>	146
Ключевое слово <i>static</i>	147
Статический конструктор.....	149
Статические классы	149
Принципы объектно-ориентированного программирования.....	150
Инкапсуляция	150
Инкапсуляция с использованием методов <i>get</i> и <i>set</i>	152
Инкапсуляция с использованием свойств	155
О доступности и статичности свойств	160
Автоматические свойства.....	160
Инициализация объекта.....	161
Организация работ при описании класса.	
Атрибут <i>partial</i>	163
Наследование.....	167
Запрет на наследование	171
Конструкторы и наследование.....	171
Добавление к классу запечатанного класса	175
Вложенность классов.....	177
Полиморфизм	178
Абстрактные классы	182
Сокрытие членов класса.....	183
Приведение классов к базовому и производному	184
Тернарный условный оператор.....	184
Операторы <i>as</i> и <i>is</i>	185
Структуры.....	191
Резюме	193

Глава 9. Обработка исключительных ситуаций195

Блоки <i>try</i> и <i>catch</i>	195
Блок <i>finally</i>	200

Глава 10. Интерфейсы203

Глава 11. Сборки, манифесты, пространства имен.	
Утилита IL DASM.....	211
Сборки	212
Пространства имен	214
Глава 12. Коллекции. Обобщения	223
Коллекции.....	223
Интерфейсы <i>IEnumerable</i> и <i>IEnumerator</i>	229
Создание собственного класса коллекций.....	233
Интерфейс <i>IDictionary</i>	242
Итератор	254
Получение копий	255
Классы <i>Array</i> и <i>List<T></i>	258
Класс <i>Array</i>	258
Класс <i>List<T></i>	271
Интерфейс <i>IList</i>	277
Создание сравнимых объектов	283
Обобщения	286
Ограничения для параметров типа.....	292
Глава 13. Делегаты и события	293
События	295
Анонимные методы	302
Лямбда-выражения	306
Лямбда-операторы.....	308
Глава 14. Введение в запросы LINQ	311
Три части операции запроса	312
О применении типа <i>var</i> в запросе.....	326
Глава 15. Некоторые сведения о процессах и потоках	
Windows	327
Вывод списка процессов	330
Вывод информации по процессу	332
Потоки процесса	333
Модули процесса	340
Запуск и остановка процессов в программе	343
Глава 16. Файловый ввод-вывод.....	349
Класс <i>DirectoryInfo</i>	350

Класс <i>Directory</i>	354
Класс <i>DriveInfo</i>	356
Класс <i>FileInfo</i>	358
Класс <i>File</i>	363
Класс <i>Stream</i>	366
Класс <i>FileStream</i>	367
Классы <i>StreamWriter</i> , <i>StreamReader</i>	369
Классы <i>StringWriter</i> и <i>StringReader</i>	378
Класс <i>StringReader</i>	384
Классы <i>BinaryWriter</i> и <i>BinaryReader</i>	385
Глава 17. Работа в многопоточном режиме.....	391
Класс <i>Thread</i>	393
Программное создание вторичных потоков	396
Класс <i>AutoResetEvent</i>	406
Проблемы разделения ресурсов	411
Класс <i>Timer</i>	413
Глава 18. Приложения типа Windows Forms	417
Создание пользовательского интерфейса.....	420
Типы <i>System.EventArgs</i> и <i>System.EventHandler</i>	426
Предметный указатель	429

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 память 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 С, он является гибридом языков С, 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 установить 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

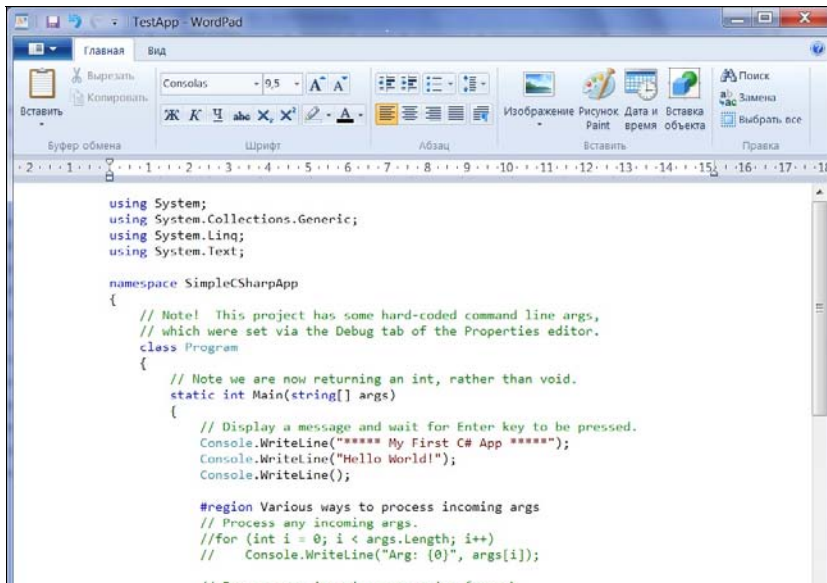


Рис. 2.1. Фрагмент текста приложения в WordPad

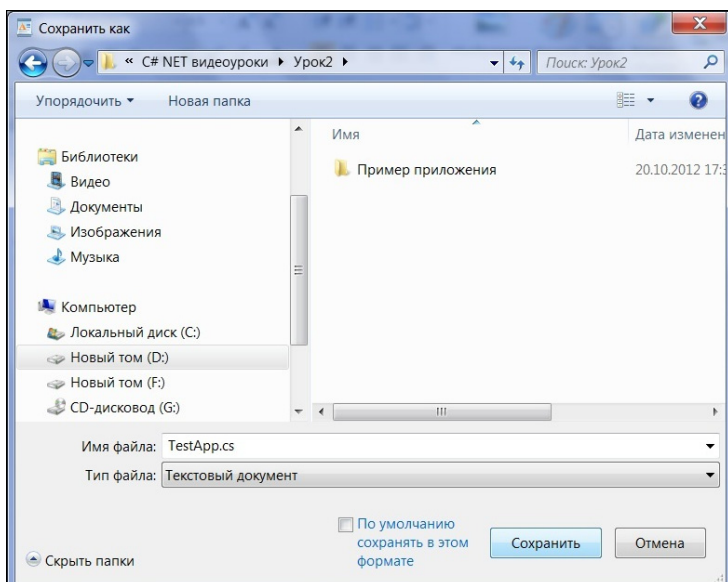
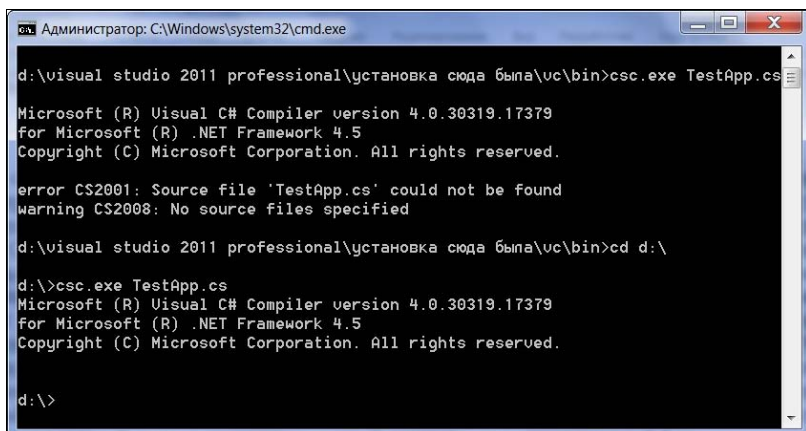


Рис. 2.2. Сохранение текста приложения редактором WordPad

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.



```
Администратор: C:\Windows\system32\cmd.exe
d:\visual studio 2011 professional\установка сюда была\vc\bin>csc.exe TestApp.cs
Microsoft (R) Visual C# Compiler version 4.0.30319.17379
for Microsoft (R) .NET Framework 4.5
Copyright (C) Microsoft Corporation. All rights reserved.
error CS2001: Source file 'TestApp.cs' could not be found
warning CS2008: No source files specified

d:\visual studio 2011 professional\установка сюда была\vc\bin>cd d:\

d:\>csc.exe TestApp.cs
Microsoft (R) Visual C# Compiler version 4.0.30319.17379
for Microsoft (R) .NET Framework 4.5
Copyright (C) Microsoft Corporation. All rights reserved.

d:\>
```

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).

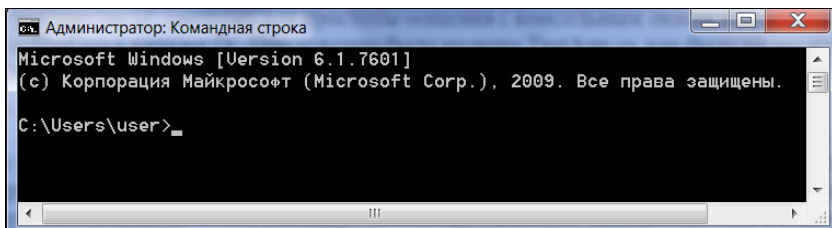


Рис. 2.4. Консольное окно командной строки операционной системы

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.sharppdevelop.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.

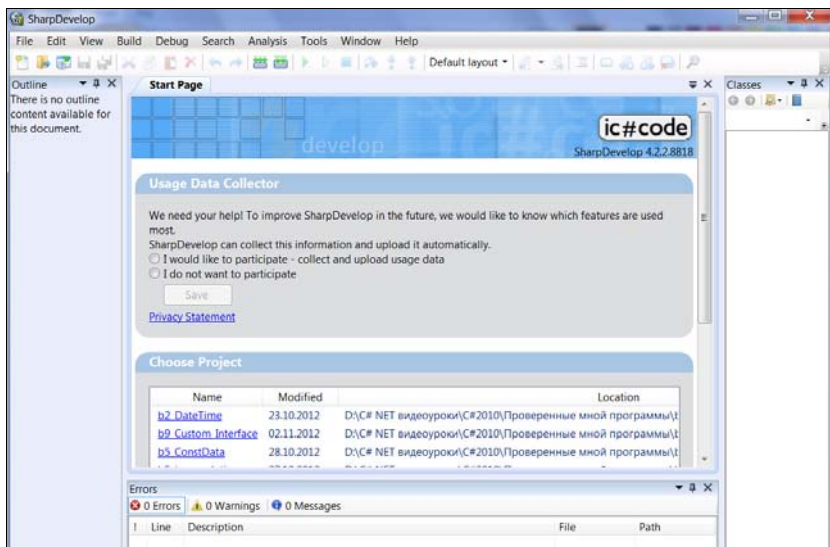


Рис. 2.5. Общий вид главного окна SharpDevelop

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-

шенного 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.

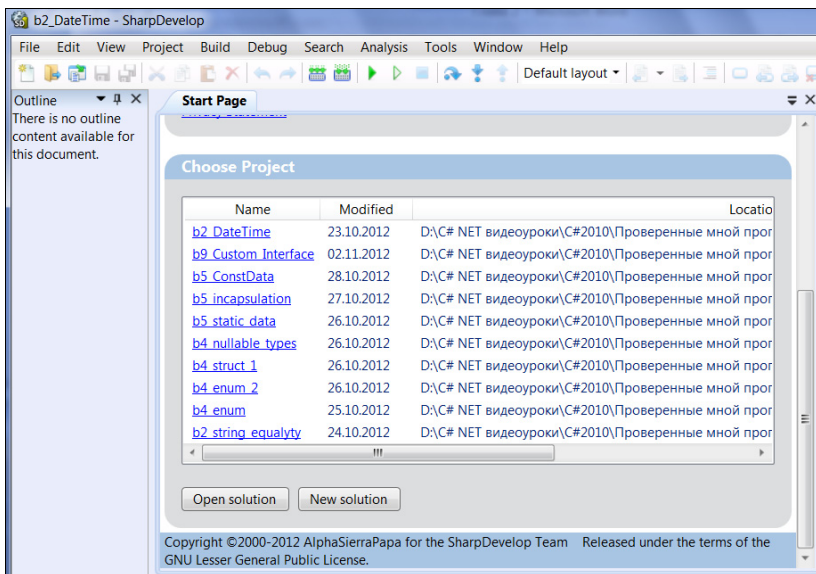


Рис. 2.6. Прокрутка окна **Choose Project**

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.

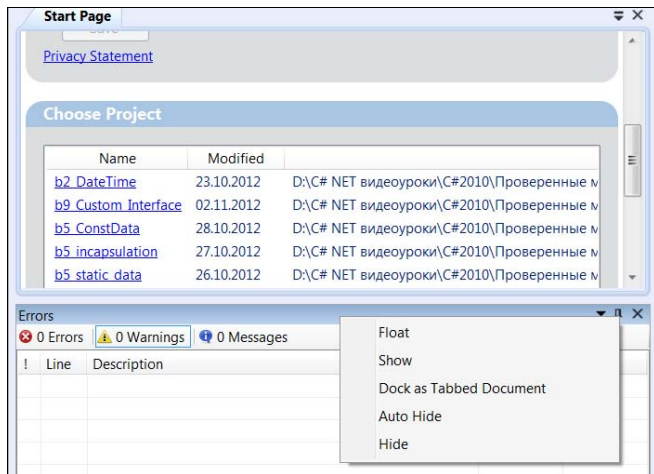


Рис. 2.7. Свойства окна в SharpDevelop

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 по мере необходимости снова ее показать, воспользовавшись командой **View | Show start page**.