

C#

СОВЕТЫ ПРОГРАММИСТАМ



Использование стандартных алгоритмов

Использование особенностей интегрированной
среды разработки IDE

Работа с формами, элементами управления

Использование технологий WMI, WSH

Вызов системных функций Windows API



АЛЕКСАНДР КЛИМОВ

УДК 681.3.068+800.92С#
ББК 32.973.26-018.1
К49

Климов А. П.

К49 С#. Советы программистам. — СПб.: БХВ-Петербург, 2008. —
544 с.: ил. + CD-ROM

ISBN 978-5-9775-0174-3

Книга представляет собой сборник советов, алгоритмов и готовых примеров программ на языке С# в среде MS Visual Studio 2005/2008 из различных областей: работа с формами и элементами управления, папками и файлами, мышью и клавиатурой, мультимедиа и графикой, использование технологий WMI и WSH, взаимодействие с MS Office и другими приложениями, работа в локальной сети и Интернете, особенности использования функций Windows API и др.

На компакт-диске размещены примеры из книги, а также демонстрационная версия справочника по функциям Windows API для .NET Framework и общения Windows для Visual Basic .NET и С#.

Для программистов

УДК 681.3.068+800.92С#
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	<i>Екатерина Кондукова</i>
Зам. главного редактора	<i>Евгений Рыбаков</i>
Зав. редакцией	<i>Григорий Добин</i>
Редактор	<i>Анна Кузьмина</i>
Компьютерная верстка	<i>Натали Смирновой</i>
Корректор	<i>Наталия Першакова</i>
Дизайн обложки	<i>Елены Беляевой</i>
Зав. производством	<i>Николай Тверских</i>

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 26.02.08.

Формат 70×100¹/₁₆. Печать офсетная. Усл. печ. л. 43,86.

Тираж 2500 экз. Заказ №

"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

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

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

ISBN 978-5-9775-0174-3

© Климов А. П., 2008
© Оформление, издательство "БХВ-Петербург", 2008

Оглавление

- ВСТУПЛЕНИЕ1**
 - Для кого предназначена книга.....2
 - Благодарности2
 - Требования2
 - Чего вы не найдете в этой книге3
 - Исходные коды4
 - Обратная связь4
- ГЛАВА 1. ОБЩЕЕ5**
 - Часто задаваемые вопросы5
 - Можно ли запустить программу, написанную на C#, без .NET Framework? ...5
 - В каком редакторе писать программы?5
 - Использование в качестве переменных русских символов7
 - Псевдонимы.....7
 - Копирующий строковой литерал8
 - Символ @ перед идентификатором8
 - Как узнать, присвоено ли переменной значение9
 - Как это назвать?10
 - Какая разница между *string* и *System.String*?10
 - Выберите свои правила наименования.....11
 - Правила для названий классов и методов11
 - Советы по созданию эффективных и масштабируемых приложений.....11
 - Сопряжение11
 - Наследование12
 - Минимизация кода.....12
 - Экономия ресурсов12
 - Создание автоматически обновляемых приложений13
 - Закключение.....13
- ГЛАВА 2. СТРОКИ, ДАТЫ, ЧИСЛА15**
 - Строки.....15
 - Простейшие операции со строками15

Входит ли строка в другую строку?	16
Преобразование строки в число	17
Вставка специального символа	17
Создание строки из повторяющихся символов	19
Метод <i>String.Format</i>	19
Преобразование строки в объект <i>Color</i>	19
Проверка строки на пустоту	20
Переворачиваем строку	21
Сжатие длинных имен файлов	22
Печатающийся текст	23
Бегущая строка	24
Как соединять строки	25
Что лучше: <i>Parse</i> или <i>TryParse</i> ?	26
Сравнение и сортировка строк	27
Даты	28
Как получить текущую дату	28
Дата и время в разных форматах	29
Как использовать дату и время в приложении	31
Сложить и вычесть временной интервал из дат	32
Вычисление разницы между датами	33
Как определить, является ли год високосным?	34
Вычисление даты католической Пасхи	35
Числа	36
Преобразование числа в шестнадцатеричную систему счисления	36
Как перевести число в двоичную систему счисления?	37
Как перевести число в восьмеричное или шестнадцатеричное представление?	37
Является ли выражение числом?	37
Создание собственной функции <i>IsNumeric</i> на C#	38
Создание уникального идентификатора	39
Перечисления	40
Как получить все элементы перечисления	40
Закключение	42
ГЛАВА 3. АЛГОРИТМЫ	43
Найти наименьшее и наибольшее значение из трех чисел	43
Массив строк	44
Преобразование градусов в радианы и радианов в градусы	45
Четное или нечетное число	46
Получить старшее и младшее слова из числа	47

Преобразование градусов по Фаренгейту в градусы по Цельсию	48
Генерирование случайного цвета	49
Подсчет суммы всех целых чисел диапазона	49
Нахождение простых чисел	50
Вывод программой своего исходного кода	53
Заключение	54
 ГЛАВА 4. ИНТЕГРИРОВАННАЯ СРЕДА РАЗРАБОТКИ (IDE)	55
Удобные клавиатурные команды	55
Получение списка всех назначенных клавиш	56
Настройка назначенных клавиш	59
Показ назначенных клавиш во всплывающих подсказках	62
Селектор оконных конфигураций	62
Фрагменты кода (code snippets)	65
Создание XML-файла	68
Встроенные фрагменты кода	72
Распространение собственных фрагментов кода	72
Настройка стартовой страницы Visual Studio	73
Настройки для групповой работы	73
Создание файла параметров	74
Размещение файла параметров в пути UNC	74
Параметр /resetuserdata	75
Ряд мелких советов	75
Как показывать нумерацию строк в редакторе кода?	76
Как изменить цвет для регионов кода?	76
Вертикальное выделение текста	76
Альтернативный метод поиска строк	76
Множественное копирование в буфер обмена	78
Как управлять фрагментами кода в Visual Studio 2008?	78
Быстрое комментирование и раскомментирование фрагментов кода	78
Отображение IntelliSense	78
Прозрачная подсказка IntelliSense	79
Перемещение от открывающей скобки к закрывающей скобке	79
Сворачивание/разворачивание блока (региона, функции, цикла и т. п.)	79
Анимация при автоматическом скрытии панелей	79
Вариант загрузки справочной системы	80
Путь к файлу	80
Быстрый переход к папке, содержащей исходные коды проекта	80
Изменение шаблона заготовки метода в C#, генерируемого автоматически	81

Вспомнить название пространства имен	82
Удобный способ вызвать Smart Tag.....	82
Создание собственных шаблонов приложений	83
Работа в полноэкранном режиме	84
Быстрый поиск в списках.....	84
Поле <i>Find</i>	84
Окно <i>Command</i>	85
Диалоговое окно <i>Find and Replace</i>	86
Еще о настройках.....	88
Скрытие статусной строки	88
Число показываемых последних файлов.....	88
Многодокументный интерфейс	88
Управление панелями Auto Hide и Close.....	88
Меню <i>Window</i>	89
Переключение между окнами.....	89
Помоги себе и команде Visual Studio, или пишем логи.....	89
Графические файлы для проектов.....	89
Надстройки	90
Надстройки сторонних разработчиков	90
GhostDoc	90
SmartPaster	92
PInvoke.NET	93
Paste as Visual Basic	95
Заключение.....	95
ГЛАВА 5. ЭКРАН И ФОРМЫ.....	97
Экран.....	99
Как определить разрешение экрана	99
Как определить рабочую область экрана без панели задач?	99
Как изменить разрешение экрана программным путем.....	100
Формы.....	103
Как вывести форму в центре экрана?	103
Как задать позицию формы на экране?	103
Как программно свернуть или развернуть форму?	104
Поддержка тем рабочего стола Windows	104
Как узнать, используются ли темы Windows XP?	105
Как отобразить форму без передачи ей фокуса?	105
Как не отображать форму при запуске программы?	106
Как сделать так, чтобы форма отбрасывала тень?.....	107
Как вывести запрос при закрытии формы?.....	108

Выбираем варианты закрытия формы	109
Скрытие значка формы на панели задач и при нажатии комбинации клавиш <Alt>+<Tab>	110
Как отобразить форму на весь экран?.....	111
Как установить ограничение на минимальный и максимальный размер окна?	111
Как отловить момент сворачивания или разворачивания формы?.....	112
Как запретить пользователю перемещать форму по экрану?.....	113
Как перемещать форму, не имеющую заголовка?.....	114
Еще два способа буксировки формы, не имеющей заголовка	116
Как добиться эффекта полупрозрачности у формы.....	118
Перемещение формы за заголовок.....	119
Неактивная форма.....	120
Как создать формы без границ и заголовка?.....	121
Как убрать кнопку X из заголовка формы?	121
Убрать кнопку X при помощи управляемого кода.....	123
Создать окно произвольной формы	123
Создание дырявой формы	125
Как создать форму в виде текста?	127
Смена темы Windows XP	128
Как форме получать уведомления о нажатии кнопок, когда фокус ввода находится в каком-либо элементе управления формы?	129
Как получить список всех открытых форм, принадлежащих приложению?.....	130
Сохранение настроек формы	131
Создание и использование параметров командной строки	131
Установить фоновый цвет в родительской MDI-форме	134
Запрет на запуск второй копии приложения.....	135
Как передавать значения между формами Windows Forms.....	136
Способ первый	136
Второй способ	137
Заключение.....	138

ГЛАВА 6. ЭЛЕМЕНТЫ УПРАВЛЕНИЯ.....139

Общие советы.....	139
Как добавить элемент управления на форму во время выполнения программы?	139
Как пройти по всем элементам управления на форме?.....	140
Как изменить цвет границы (<i>Border</i>) у элемента управления?	141
Окантовка вокруг элемента управления	142

Как программно перевести фокус на следующий/предыдущий (в порядке TAB) элемент управления?	143
Как изменить Z-порядок элемента управления?	144
Как узнать размеры строки в пикселах, отображаемой в каком-нибудь элементе управления?	145
Как сделать элемент управления произвольной формы?	146
Кнопки (<i>Button</i>)	148
Как установить кнопку по умолчанию для формы?	148
Как установить кнопку отмены (<i>Cancel</i>) для формы?	148
Как программно вызвать событие <i>Click</i> у кнопки?	148
Как создать западающую кнопку?	149
Список (<i>ListBox</i>)	149
Автоматическая прокрутка списка	149
Подгоняем ширину списка под самый длинный текст	150
Как заполнить список именами файлов, перетаскиваемых из Проводника?	150
Разделить список цветными линиями и заполнить цветным текстом	151
Поле со списком (<i>ComboBox</i>)	153
Подгоняем ширину поля со списком под самый длинный текст	153
Поддержка автозавершения	153
Как раскрыть поле со списком программным способом?	154
Как запретить раскрытие списка?	155
Как изменить высоту элементов списка у элемента управления <i>ComboBox</i> ?	156
Как установить желаемую высоту выпадающего списка у <i>ComboBox</i> ?	157
Как использовать <i>ComboBox</i> для редактирования данных в <i>ListView</i> ?	157
Текстовые поля (<i>TextBox</i>)	158
Подсчет числа строк в многострочном текстовом поле	158
Фильтрация заданных символов при вводе с клавиатуры	159
Как заблокировать контекстное меню в текстовом поле?	160
Запрет вставки текста из буфера обмена Windows	160
Как ввести многострочный текст в текстовое поле программно?	161
Как сделать так, чтобы символы вводились в нужном регистре?	162
Как избавиться от звукового сигнала при нажатии на клавишу ввода?	162
Как выделить текст программным способом?	163
Элемент <i>RichTextBox</i>	164
Просмотр форматированного текста RTF	164
Как управлять цветом и шрифтами в <i>RichTextBox</i> ?	165
Как управлять текстом-гиперссылкой в <i>RichTextBox</i> ?	166
Поддержка Drag'n'Drop	166

Как определить наличие полос прокрутки в элементе <i>RichTextBox</i> ?	167
Как запретить вставку	168
Элемент управления <i>MaskedTextBox</i>	169
Элемент <i>DateTimePicker</i>	169
Как показать пустой текст, если в <i>DateTimePicker</i> не выбрана дата?	169
Как программно раскрыть <i>DateTimePicker</i> ?	170
Элементы <i>Label</i> и <i>Panel</i>	170
Полупрозрачная надпись	170
Использование <i>Label</i> в виде разделительной линии как элемент дизайна	171
Элемент <i>LinkLabel</i>	172
Отображение лишь части текста в виде ссылки	173
Несколько ссылок в одном <i>LinkLabel</i>	174
<i>NotifyIcon</i> — значок в области уведомлений	175
Как создать мигающий значок в области уведомлений?	176
Как создать анимированный значок в области уведомлений?	177
Свертывание формы вместо закрытия приложения	178
Элемент <i>ListView</i>	179
Как убрать выделение элемента в <i>ListView</i> программно?	179
Как программно выбрать элемент в <i>ListView</i> ?	179
Как сортировать элемент управления <i>ListView</i> по колонкам	180
Изменение цвета подэлементов <i>ListView</i> программным путем	185
Элемент управления <i>ToolTip</i>	186
Почему пользователь не видит подсказки в стиле <i>Balloon</i> ?	186
Многострочная подсказка	187
Меню	187
Фон для меню	188
Как добавить контекстное меню элементу управления?	188
Как определить, какой элемент вызвал контекстное меню?	188
Автоматическое закрытие контекстного меню через заданный промежуток времени	189
Дерево (<i>TreeView</i>)	190
Как показать подсказку над узлом <i>TreeView</i> ?	190
Вкладки (<i>TabControl</i>)	192
Программное переключение на другую вкладку	192
Установка фокуса на элементе управления на вкладке во время загрузки формы	192
Как вывести ярлычки внизу вкладки <i>TabControl</i> ?	193
Добавление новой вкладки	193
Удаление вкладки	193
Как вставить вкладку в определенную позицию?	194

Элемент <i>PerformanceCounter</i>	195
Как создать счетчик производительности процессора?	195
<i>StatusBar</i> и <i>StatusStrip</i>	197
Как изменить шрифт и фон для <i>StatusBar</i>	197
Элементы <i>FlowLayoutPanel</i> и <i>TableLayoutPanel</i>	198
Элемент <i>DataGrid</i>	198
Элемент <i>DataGridView</i>	199
Создание собственных элементов управления	200
Как скрыть свойство или метод от IntelliSense в редакторе кода?	200
Как скрыть свойства и события из редактора свойств <i>PropertyGrid</i> при создании собственного элемента управления?	201
Как запретить изменять размер элемента управления во время разработки?	202
Как во время разработки позволить выбирать значение свойства из нескольких предопределенных в поле со списком?	203
Как добиться того, чтобы свойство моего элемента управления было видно в разделе <i>DataBindings</i> окна свойств?	204
Как сделать свой элемент управления, выступающий в роли контейнера для других элементов управления во время разработки?	204
Как присвоить свой значок для собственного элемента управления в панели инструментов?	205
Создание собственного элемента управления <i>SmoothProgressBar</i>	206
Создание элемента <i>SmoothProgressBar</i>	208
Создание клиентской программы для тестирования	213
Заключение	215
ГЛАВА 7. ГРАФИКА	217
Преобразование цвета в HTML-формат	217
Как преобразовать цвет в целое число?	218
Как получить доступ к определенному пикселу изображения?	218
Как нарисовать точку?	218
Как получить цвет любой выбранной точки экрана?	219
Как нарисовать прямоугольник с закругленными краями?	222
Установка фонового изображения	223
Как сделать снимок экрана?	225
Сохранить изображение элемента управления или формы	225
Как получить прокручиваемый рисунок?	226
Получение негатива изображения	226
Сделать изображение серым	227
Как создать затемненную картинку	228

Эффект недоступной кнопки	229
Как нарисовать вдавленный и выпуклый текст?	230
Как получить контурный текст.....	231
Как отразить текст в зеркальном отражении?	232
Как повернуть текст под некоторым углом?.....	233
Вот новый поворот (из песни группы "Машина времени").....	234
Бегущая градиентная строка.....	235
Скроллинг текста.....	236
Анимированные картинки	238
Как сохранить изображение из буфера обмена в файл	239
Шрифты и печать.....	240
Получение списка установленных шрифтов.....	240
Использование собственных шрифтов.....	242
Получение списка установленных принтеров.....	243
Как распечатать документ?	244
Как показать окно предварительного просмотра перед печатью	246
Заключение.....	247
 ГЛАВА 8. РАБОТА С МЫШЬЮ И КЛАВИАТУРОЙ	249
Мышь	249
Как скрыть и показать указатель мыши?.....	249
Как установить позицию указателя мыши?	250
Анимированные курсоры.....	251
Мышеловка.....	251
Право выбора	252
Меняем кнопки мыши местами.....	254
Как узнать координаты мыши?	255
Как преобразовать экранные координаты в клиентские (для данного элемента) и наоборот?	255
Как двигать указателем мыши программно?	256
Как выполнить эмуляцию щелчков мыши?	258
Рисование.....	259
Работа с клавиатурой.....	262
Как переключать раскладки клавиатуры?	262
Как получить текущий язык ввода?	263
Как послать нажатия клавиш программно?	263
Как включать и выключать индикаторы клавиш <Caps Lock>, <Num Lock> и <Scroll Lock>?.....	264
Как определить состояние клавиш-индикаторов?.....	266
Последнее нажатие на клавишу или на кнопку мыши.....	267
Заключение.....	269

ГЛАВА 9. ПРИЛОЖЕНИЯ	271
Работа с процессами	271
Как получить полное имя файла запущенного приложения?	271
Как получить путь к папке, из которой запущено приложение?	272
Как запустить другой исполняемый файл из своего приложения?	272
Как закрыть все копии Блокнота?	274
Запуск программы по имени файла	275
Как узнать число процессоров в системе?	276
Как приостановить выполнение программы на несколько секунд?	277
Как получить список всех процессов, запущенных в системе?	277
Как получить список только оконных процессов на моей машине?	278
Как получить список определенных процессов?	279
Получение списка процессов на удаленной машине	279
Как открыть почтовый клиент, установленный по умолчанию, и установить необходимые параметры для отправки письма?	280
Определение операционной системы пользователя	281
Определение версии .NET Framework и ее сервис-пака	283
.NET Framework 1.0	284
.NET Framework версий 1.1, 2.0 3.0 и v3.5 (Orcas)	284
.NET Framework 3.0	285
Определение папки установки .NET Framework	294
Номер сборки	295
Обновление номера версии сборки в автоматическом режиме	295
Вызов файла справки CHM	296
Получение номера версии файла и другую информацию	296
Определение имени пользователя системы	298
Как определить, имеет ли ваша система мышшь, узнать число кнопок у мыши, размер вашего монитора и другую информацию?	298
Как зарегистрировать файлы DLL и OCX?	300
Извлечение строки или значка из ресурсов	301
Сохранение настроек приложения	302
Работа с реестром	304
Определение архитектуры операционной системы	306
Добавление программы в автозагрузку	306
Получение информации об изменениях в системе	308
Как узнать, что пользователь изменил разрешение экрана?	308
Изменение времени	309
Консольные приложения	310
Журналы событий	311
Как найти доступные журналы событий на компьютере?	311

Чтение и запись логов в журнал событий	311
Запись в журнал	312
Очистка записей в журнале событий	313
Создание собственного журнала событий.....	313
Удаление собственного журнала событий	314
Измерение времени выполнения кода в приложении.....	314
Измерение с помощью функций Windows API.....	315
Измерение с помощью метода <i>Ticks</i>	316
Измерение с помощью <i>TickCount</i>	316
Класс <i>StopWatch</i>	317
Заключение.....	318
 ГЛАВА 10. ДИСКИ, ПАПКИ И ФАЙЛЫ	319
Диски.....	319
Как получить список логических дисков?	319
Как узнать тип диска и его свойства?	320
Папки	321
Как получить список папок?.....	321
Как проверить существование папки?	322
Как переименовать папку?.....	322
Как удалять папки?	323
Как выбрать папку?	324
Как получить путь для папки Мои документы и других специальных папок Windows?	324
Свойства папки	326
Размеры папки.....	327
Как написать свой Проводник?	328
Файлы.....	330
Как получить список файлов в папке?.....	331
Как получить список папок и файлов?	331
Как получить список файлов по маске?	332
Как узнать, существует ли файл?	332
Как получить имя файла из полного пути файла?.....	332
Как получить расширение файла из полного пути?	333
Как создать, удалить, переместить файл?	333
Как установить атрибуты у файла?	334
Свойства файла	334
Как извлечь информацию о файле?	335
Как создать временный файл?	336
Как создать уникальное имя для файла?	336

Как ограничить доступ к файлу?	337
Как работать с бинарными файлами?	338
Как работать с текстовыми файлами?	339
Как добавить текст в существующий файл?	340
Построчное чтение текстового файла	340
Загрузить текстовый файл в список?	341
Как получить короткое имя файла из длинного файла и наоборот?	343
Как удалить файл в Корзину	344
Как записать и прочитать текст в различных кодировках?	346
Как прочитать XML-файлы?	347
Сравнение двух файлов	351
Отслеживание изменений в файловой системе	353
Как установить уровень доступа к файлу?	355
Заключение	356
ГЛАВА 11. БИБЛИОТЕКА WSH	357
Создание ярлыка	357
Получение списка установленных в системе принтеров	358
Установка принтера по умолчанию	359
Получение списка сетевых дисков	359
Заклучение	360
ГЛАВА 12. WMI	361
Использование WMI на удаленной машине	361
Информация об операционной системе	362
Информация о компьютере	364
Информация о производителе	365
Получение информации о процессорах	366
Информация о свойствах видеоконтроллера	369
Получение свойств приводов компакт-дисков	370
Информация о параметрах загрузки Windows	371
Информация о сетевом адаптере	372
Информация о мониторе	373
Материнская плата	373
Вывод списка общих ресурсов	374
Информация о логических дисках	375
Перезагрузка компьютера	376
Дополнительный пример	377
Заклучение	378

ГЛАВА 13. МУЛЬТИМЕДИА	379
Звуковые сигналы	379
Функция Windows API <i>Beep</i>	379
Функция Windows API <i>MessageBeep</i>	379
Функция <i>Beep</i> для Visual Basic	380
Звуковые файлы	380
И снова о <i>Beep</i>	381
Как проигрывать звуки разных форматов?	382
Воспроизведение MIDI и MP3 через неуправляемый код	382
Извлечение информации из файлов MP3	383
Взаимодействие с Winamp	392
Закключение	393
ГЛАВА 14. РАЗРАБОТКА ЛОКАЛИЗОВАННЫХ ПРИЛОЖЕНИЙ.....	395
Общая информация о локализации	397
Локализирующие идентификаторы	398
Культура.....	399
Приложение Culture Explorer	400
Разработка многоязычного приложения	403
Разделяй и властвуй.....	405
Сопутствующие сборки.....	405
Закключение	406
ГЛАВА 15. MICROSOFT OFFICE.....	407
Excel	407
Раннее связывание	407
Автоматизация Excel или как работать массивами	409
Позднее связывание.....	414
Outlook	417
Как получить сообщения из папки Входящие?	417
Получение уведомлений о новых письмах.....	419
VSTO.....	420
Закключение	421
ГЛАВА 16. ЛОКАЛЬНАЯ СЕТЬ И ИНТЕРНЕТ.....	423
Информация о сети.....	423
Как получить хост, порт, протокол из веб-адреса?	423
Как получить IP-адрес компьютера, используя DNS?	424

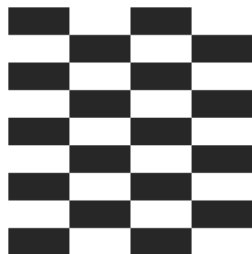
Как получить NETBIOS-имя машины?	425
Как получить IP-адрес локальной машины?	425
Ping	426
Проверка доступности веб-адреса.....	428
Подключен ли компьютер к Интернету?.....	429
Пересылка данных по протоколу HTTP.....	430
Как послать запрос GET и отобразить полученные данные?	430
Как скачать файл из Интернета?	431
Передача файлов по протоколу FTP	432
Закачка файла на FTP-сервер.....	433
Получение оглавления папки.....	435
Загрузка файлов	437
Отправка писем через SMTP	438
Использование браузера Mozilla Firefox.....	439
Работа с локальной сетью	443
Как получить имя текущего пользователя?.....	443
Как выяснить, подключена ли локальная система к сети, и узнать используемый тип соединения?	444
Получение списка всех компьютеров локальной сети.....	445
Список SQL-серверов при помощи управляемого кода	453
Как получить дату и время удаленного компьютера?.....	454
Заключение.....	457

ГЛАВА 17. ФУНКЦИИ WINDOWS API.....459

Вызов функций Windows API, имеющих выходной строковый параметр <i>char*</i>	460
Изменение типа, применяемого для маршалинга по умолчанию	461
Вызов функций, требующих <i>struct</i>	462
Работа с функциями обратного вызова в C#.....	463
Создание собственной управляемой библиотеки	463
Примеры использования функций API.....	464
Блокировка компьютера.....	464
Является ли текущий пользователь администратором?	464
Мигание заголовка формы.....	465
Форматирование дисков.....	466
Открытие и закрытие лотка привода компакт-дисков	467
Создание собственного пункта в системном меню	468
Работа с конфигурационными файлами INI.....	471
Извлечение значков из файлов	473
Вызов диалогового окна <i>Смена значка</i>	474

Продолжаем работать со значками	475
Панель задач, кнопка <i>Пуск</i> и часы в области уведомлений.....	477
Смена обоев Рабочего стола	481
Использование функций обратного вызова	484
Получение списка кодовых страниц, установленных в системе.....	484
Заключение	485
ГЛАВА 18. НОВИНКИ VISUAL STUDIO 2008	487
Новшества в C# 3.0.....	487
Неявно типизированные переменные	487
Инициализация объектов	488
LINQ	489
Вывод чисел из заданного массива с условием	489
Ключевое слово <i>Where</i>	490
Увеличение на единицу ряда чисел.....	490
Вывод имени числа.....	491
Вывод строк из массива в разных регистрах	491
Оператор <i>Take</i>	492
Оператор <i>TakeWhile</i>	493
Оператор <i>Skip</i>	493
Оператор <i>SkipWhile</i>	494
Заклучение	494
ГЛАВА 19. ССЫЛКИ НА ИНТЕРЕСНЫЕ МЕСТА.....	495
Сайты	495
Блоги	496
Заклучение	497
ОПИСАНИЕ КОМПАКТ-ДИСКА	499
ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ.....	501

ГЛАВА 2



Строки, даты, числа

В этой главе речь пойдет о базовых объектах, работать с которыми приходится практически в любом приложении.

Строки

Строки встречаются практически во всех приложениях, поэтому хорошее знание всех строковых методов и свойств необходимо любому программисту. Рассмотрим несколько советов. Необходимо помнить, что строки представлены классом `System.String` (или `string` в нотации C#) и любые операции, связанные со строками, на самом деле не изменяют саму строку, а создают новую строку. Поэтому во многих случаях лучше использовать класс `StringBuilder`, который в большей степени оптимизирован для манипуляции со строками.

Простейшие операции со строками

Для вставки одной строки в другую используется метод `Insert`, проиллюстрированный в листинге 2.1.

Листинг 2.1. Вставка одной строки в другую строку

```
string partBookTitle = "C#.советы";  
string insertText = "Народные ";  
string bookTitle = partBookTitle.Insert(3, insertText);  
MessageBox.Show(bookTitle);
```

Для удаления подстроки из заданной строки используется метод `Remove`. Если вы хотите удалить подстроку с указанной позиции до конца строки, то достаточно указать индекс нужного символа, как это сделано в листинге 2.2.

Листинг 2.2. Удаление подстроки из заданной строки

```
string bookTitle = "С#.Народные советы";  
// Удаляем подстроку с третьей позиции  
bookTitle = bookTitle.Remove(2);  
MessageBox.Show(bookTitle);
```

Для извлечения части строки из заданной строки используется метод `Substring`, применение которого показано в листинге 2.3.

Листинг 2.3. Извлечение подстроки из заданной строки

```
string bookTitle = "С#.Народные советы";  
// Извлекаем подстроку с шестой позиции с размером в три символа  
bookTitle = bookTitle.Substring(5, 3);  
MessageBox.Show(bookTitle);
```

ПРИМЕЧАНИЕ

Visual Basic.NET имеет в своем арсенале также такие методы и свойства, как `Left`, `Right`, `Len` и другие, унаследованные от Visual Basic 6.0. Вполне возможен вариант, что вам может попасться код, который использует эти устаревшие конструкции. Необходимо избавляться от подобных строчек кода, которым есть достойная альтернатива среди классов .NET Framework.

Входит ли строка в другую строку?

Метод `IndexOf`, проиллюстрированный в листинге 2.4, позволяет определить место вхождения подстроки в заданную строку.

Листинг 2.4. Определение вхождения подстроки в заданной строке

```
string str1 = "око";  
string str2 = "Царь-колокол";  
int i = str2.IndexOf(str1);  
// Проверяем, входит ли строка око в слово Царь-колокол  
if(i >= 0) MessageBox.Show(str1 + " входит в строку " + str2);
```

Также можно воспользоваться регулярным выражением, подключив пространство имен `System.Text.RegularExpressions`. В этом случае используется метод `Regex.IsMatch`. А если вы хотите написать собственный метод для нахождения строки в другой строке (для общего развития), то обратите внимание на статью "Find a string inside another string" на сайте <http://dirtydogstink.com/blog/2007/02/16/how2FindAStringInsideAnotherString.aspx>, чтобы не изобретать заново велосипед.

Преобразование строки в число

Не менее распространенная задача — преобразовать строку в число. В этом случае, как показано в листинге 2.5, можно использовать метод `Parse`.

Листинг 2.5. Преобразование строки в число

```
string tankman = "4";  
string dog = "1";  
textBox1.Text = tankman + dog; // получим 41  
int all = int.Parse(tankman) + int.Parse(dog);  
textBox1.Text += Environment.NewLine + all.ToString(); // получим 5
```

Как видите, сначала мы пытаемся сложить две строки, чтобы узнать численность экипажа танка из фильма "Четыре танкиста и собака". Результат, равный 41, нас явно не устроит. Поэтому сначала преобразуем строки в числа, а затем снова попытаемся сложить две переменные.

Существует еще один способ преобразования строки в число — методы класса `Convert`.

Вставка специального символа

Несмотря на то, что на клавиатуре имеется более 100 клавиш, некоторые символы на клавиатуре отсутствуют. Например, к разряду таких символов можно отнести знаки копирайта, знак зарегистрированной торговой марки, символы некоторых валют и т. д. В стандарте Unicode знак копирайта имеет десятичный код 169. В листинге 2.6 показано, как, используя класс `Convert`, можно преобразовать код нужного символа и получить нужный символ как часть строки, который затем можно вывести в текстовом поле (рис. 2.1).

Листинг 2.6. Вставка специального символа

```
int charCode = 169;  
char specialChar = Convert.ToChar(charCode);  
textBox1.Text = specialChar.ToString();
```

Можно использовать и шестнадцатеричное значение кода Unicode. Например, для вывода знака зарегистрированной торговой марки используем пример, приведенный в листинге 2.7.

Листинг 2.7. Вставка символа торговой марки

```
// Unicode-код для торговой марки  
int charCode = 0x00AE;  
char specialChar = Convert.ToChar(charCode);  
textBox1.Text += specialChar.ToString();
```

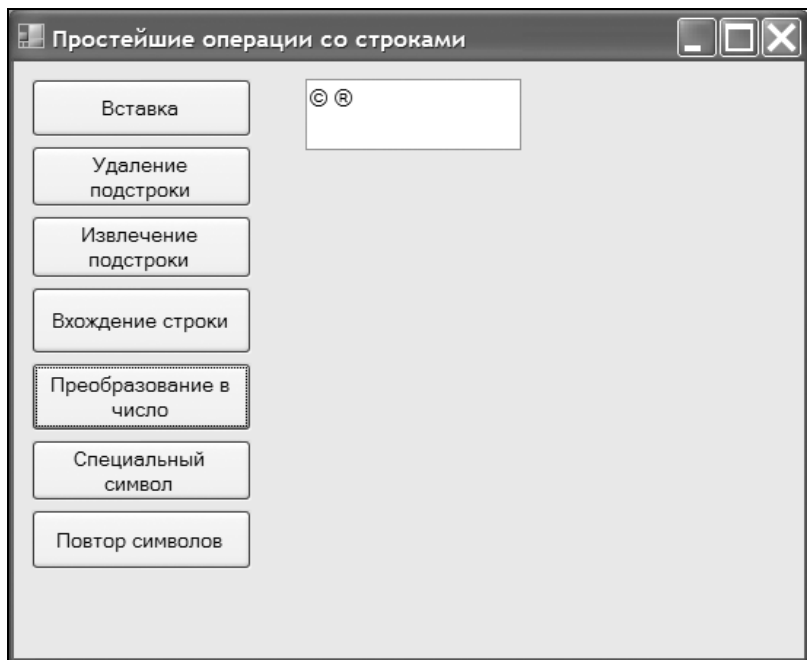


Рис. 2.1. Вывод специальных символов
в текстовом поле

Создание строки из повторяющихся символов

Для создания строки, состоящей из одинаковых символов, достаточно (листинг 2.8) использовать простую конструкцию.

Листинг 2.8. Создание строки из повторяющихся символов

```
// Создаем строку из 5 звездочек
System.String FiveStars = new System.String('*', 5);
textBox1.Text = FiveStars;
```

Все остальные методы для работы со строками вы можете изучить самостоятельно. Комбинируя строковые методы, можно добиться некоторых интересных эффектов.

Метод *String.Format*

Не забывайте о возможности использовать метод `String.Format` при выводе строк для представления текста в нужном виде. Например, чтобы вывести строчку "Стоимость BMW равна 12 000 р.", можно использовать код, показанный в листинге 2.9.

Листинг 2.9. Использование метода `String.Format`

```
private void butFormatStr_Click(object sender, EventArgs e)
{
    string AutoName;
    AutoName = "BMW";
    textBox1.Text =
        String.Format("Стоимость {0} равна {1:0.0;c}", AutoName, 12000);
}
```

Преобразование строки в объект *Color*

Если у вас возникнет необходимость преобразовать строку с названием цвета в сам объект `Color`, то воспользуйтесь классами `TypeDescriptor` и `TypeConverter`, который предоставляет унифицированный способ конвертирования типов значений в другие типы. Например, мы хотим преобразовать цвет `Color.Blue` в строку (или наоборот, преобразовать строку "Green" в объект `Color`). Делается это, как показано в листинге 2.10.

Листинг 2.10. Преобразование строки в объект Color и обратно

```
private void butColorName_Click(object sender, EventArgs e)
{
    // Задаем цвет
    Color clr = Color.Blue;
    // Получим имя выбранного цвета
    textBox1.Text =
        (TypeDescriptor.GetConverter(clr).ConvertToString(clr));

    // Обратная задача. Конвертируем название цвета Green в объект Color
    clr = (Color)TypeDescriptor.GetConverter(
        typeof(Color)).ConvertFromString("Green");
    // Закрашиваем форму в выбранный цвет
    this.BackColor = clr;
}
```

Проверка строки на пустоту

Существует несколько способов проверить строку на пустоту:

- ☐ `if (myString == "")` // пустая строка;
- ☐ `if (myString == String.Empty)` // пустая строка;
- ☐ `if (myString.Length == 0)` // строка с нулевой длиной;
- ☐ `if (String.Equals(myString, String.Empty))`.

Ян Нельсон (Ian Nelson) в своем блоге <http://ianfnelson.com/blog/archive/2004/07/30/171.aspx> не поленился и подсчитал, сколько времени занимает каждый из этих вариантов при обработке 50 миллионов итераций. Результаты вы можете посмотреть на сайте, но самым быстрым вариантом стал третий способ. Но данный способ не учитывает, что строка может принимать значение `null`. Эксперименты проводились в 2004 году. С появлением .NET Framework 2.0 класс `System.String` обзавелся новым статическим методом `IsNullOrEmpty` (листинг 2.11), который позволяет быстро и просто проверить строку — пустая она или имеет значение `null`.

Листинг 2.11. Проверка строки на пустоту

```
public void SayHello(string name)
{
```

```
if ( string.IsNullOrEmpty(name) )
    throw new ArgumentNullException("name");
MessageBox.Show( string.Concat("Hello, ", name) );
}
```

Переворачиваем строку

Иногда встречается задача перевернуть строку наоборот. В Visual Basic 6.0 появилась удобная строковая функция `StrReverse`, которая позволяла быстро и удобно перевернуть строку. Разработчики Visual Studio оставили эту функцию для программистов Visual Basic .NET. Таким образом, вам нужно установить ссылку на пространство имен `Microsoft.VisualBasic` и воспользоваться функцией по своему назначению (листинг 2.12).

Листинг 2.12. Переворачиваем строку при помощи функции Visual Basic

```
using Microsoft.VisualBasic;

private void butReverseVB_Click(object sender, EventArgs e)
{
    // Взять текст из текстового поля
    // Например, А роза упала на лапу Азора
    string myString = textBox1.Text;
    // Используем встроенную функцию Visual Basic
    textBox1.Text = Strings.StrReverse(myString);
}
```

Если вы по каким-то причинам не хотите пользоваться библиотекой Visual Basic, можно написать собственную процедуру переворачивания строки, приведенную в листинге 2.13.

Листинг 2.13. Переворачиваем строку при помощи C#

```
public static string ReverseString(string str)
{
    // Проверка на непустоту строки.
    if(string.IsNullOrEmpty(str))
    {
        return str;
    }
}
```



```

// Создадим объект StringBuilder с нужной длиной.
StringBuilder revStr = new StringBuilder(str.Length);
// Перебираем в цикле все символы
// и присоединяем каждый символ к StringBuilder
for (int count = str.Length - 1; count > -1; count--)
{
    revStr.Append(str[count]);
}
// Возвращаем перевернутую строку
return revStr.ToString();
}

private void butReverseCS_Click(object sender, EventArgs e)
{
    textBox1.Text = ReverseString(textBox1.Text);
}

```

ПРИМЕЧАНИЕ

Примеры с простейшими операциями со строками вы можете найти в папке SimpleStrings на прилагаемом диске.

Сжатие длинных имен файлов

Если вам часто приходится работать со строками, которые представляют собой длинные полные пути к файлам, то возможно вам пригодится следующий пример. С помощью функции Windows API `PathCompactPathEx` можно заменить часть длинного пути к файлу многоточием, например, как это сделано в листинге 2.14.

Листинг 2.14. Замена длинного пути многоточием

```

using System.Runtime.InteropServices;

[DllImport("shlwapi.dll", CharSet = CharSet.Auto, SetLastError = true)]
private static extern bool PathCompactPathEx(
    System.Text.StringBuilder pszOut,
    string pszSrc,
    Int32 cchMax,
    Int32 dwFlags);

```

```
private void button1_Click(object sender, EventArgs e)
{
    // длинный путь к файлу
    string strPathFile =
        "c:/program files/My SuperProgram/skins/sample.txt";
    StringBuilder sb = new StringBuilder(260);

    // оставляем 20 символов, остальное заменяем многоточием
    bool b = PathCompactPathEx(sb, strPathFile, 20+1, 0);

    // Выводим результат в текстовое поле
    textBox1.Text = sb.ToString();
}
```

ПРИМЕЧАНИЕ

Пример со сжатием строки находится в папке CompactPathDemo на прилагаемом диске.

Печатающийся текст

Чтобы создать видимость того, что текст печатается на печатной машинке, нужно использовать таймер, с помощью которого через определенный промежуток времени мы считываем все символы текста с первой буквы до текущей и выводим их на экран. Счетчик текущего количества букв постоянно обновляется (листинг 2.15), что позволяет повторять эффект печатающегося текста бесконечно, пока пользователь не остановит этот процесс.

Листинг 2.15. Эффект печатающегося текста

```
private void button1_Click(object sender, EventArgs e)
{
    if (button1.Text == "Старт")
    {
        timer1.Enabled = true;
        button1.Text = "Стоп";
    }
    else
    {
        timer1.Enabled = false;
    }
}
```

```
        button1.Text = "Старт";
    }
}

public static int counter = 0;

private void timer1_Tick(object sender, EventArgs e)
{
    string typingText = "С#.Народные советы";

    this.Text = typingText.Substring(0, counter);
    counter++;

    if (counter > typingText.Length)
        counter = 0;
}
```

Бегущая строка

Второй распространенный эффект со строками — создание бегущей строки. Алгоритм создания эффекта бегущей строки заключается в следующем — из исходной строки удаляем один символ слева и добавляем его с правой стороны текста.

Листинг 2.16. Создание бегущей строки

```
private void butScroll_Click(object sender, EventArgs e)
{
    timer2.Enabled = true;
}

// Исходная строка. Для большего удобства добавлено несколько пробелов
// в конец строки
private string scrollText = "С#.Народные советы    ";

private void timer2_Tick(object sender, EventArgs e)
{
    // Удаляем один символ слева и прибавляем его с правой стороны
    scrollText = scrollText.Substring(1,
```

```
(scrollText.Length - 1)) + scrollText.Substring(0, 1);  
this.Text = scrollText;  
}
```

ПРИМЕЧАНИЕ

Примеры находятся в проекте StringsFX на прилагаемом диске.

Как соединять строки

Во многих книжках пишут, что для сцепления строк нужно использовать класс `StringBuilder`, как самый эффективный способ.

Листинг 2.17. Соединение строк при помощи `StringBuilder`

```
StringBuilder sb = new StringBuilder();  
sb.Append("строка 1");  
sb.Append("строка 2");  
this.Text = sb.ToString();
```

Однако, на самом деле, это не всегда самый лучший выбор. В одном из выпусков журнала "MSDN Magazine" была статья, в которой говорилось, что существуют случаи, когда использование класса `StringBuilder` для каждой операции сцепления является не самым быстрым решением. Использование класса `StringBuilder` наиболее эффективно, если сцепляется много строк, если число строк неизвестно или если строки длинные. Однако если в вашей программе сцепляются лишь несколько строк, или если они относительно коротки, то лучшим выбором часто является использование стандартной операции сцепления. В этих случаях можно использовать либо стандартную конструкцию сцепления используемого языка (для языка C# это символ `+`), либо метод `String.Concat`. Такие способы обеспечивают сравнимую производительность. Дело в том, что при использовании класса `StringBuilder` происходят дополнительные затраты ресурсов, которые не возникают при использовании стандартных строк. К тому же, класс `StringBuilder`, в конечном счете, необходимо все равно преобразовывать в строку, что опять приводит к дополнительным расходам. Следовательно, если вы занимаетесь оптимизацией вашего приложения, то вам необходимо убедиться в том, что используемые методы сцепления строк себя оправдывают. В указанной статье рассматривается пример сцепления переменного количества строк длиной 25 символов каждая и показывается графики зависимости времени, не-

обходимого для сцепления нескольких двадцатипятисимвольных строк с объектом класса `StringBuilder` и со строкой, от их количества. Результаты показывают, что если число сцеплений больше семи, лучше использовать класс `StringBuilder`, а если сцепляемых строк меньше, обычные способы соединения строк дают лучшую производительность.

ПРИМЕЧАНИЕ

Более подробную информацию вы можете прочитать в статье "CLR Inside Out" в январском номере 2006 года журнала "MSDN Magazine". Существует также русскоязычная онлайн-версия статьи "CLR вдоль и поперек" на сайте <http://msdn.microsoft.com:80/msdnmag/issues/06/01/CLRInsideOut/default.aspx?loc=ru>.

Что лучше: *Parse* или *TryParse*?

В статье из предыдущего примера также сравниваются возможности методов `Parse` и `TryParse`. В .NET Framework 2.0 появился новый метод для базовых типов `TryParse`, который похож на метод `Parse`. Разница в их поведении заключается в том, что метод `Parse` при неудачной проверке данных вызывает исключение, а метод `TryParse` просто переходит к инструкции `else`. Нас интересует, что лучше использовать в программах для повышения производительности. Оказывается, что при успешном анализе значения строки `someString` производительность обоих методов в точности одинакова. Таким образом, использование метода `Parse` не приводит к дополнительным затратам, если точно известно, что имеющаяся строка соответствует тому типу, в который она преобразуется. Однако производительность падает, если вам попадется хотя бы одна строка, которая не может быть успешно преобразована. В этом случае при использовании метода `Parse` управление будет передано в блок обработки исключения. А исключения практически всегда замедляют работу. Если данная ситуация возникает не слишком часто, то это не очень страшно. В противном случае приходится платить значительным ухудшением производительности. В противоположность методу `Parse` метод `TryParse` при такой ситуации использует блок `if-else`. При переходе к блоку `else` производительность не страдает. Метод `TryParse` всегда работает с той же или большей производительностью, что и метод `Parse`, но разница между ними иногда может быть значительной. В статье приводятся такие цифры: в самом крайнем случае, когда ни одну строку не удастся преобразовать в базовый тип, использование метода

`Parse` может занять в 150 раз больше времени, чем использование метода `TryParse`. С другой стороны, если анализируется 100 строк, и только одна из них не может быть успешно преобразована, метод `TryParse` работает всего лишь на 30 процентов быстрее. Таким образом, подтверждается общее правило, что не следует использовать исключения в качестве механизма управления логической последовательностью операций. Хотя использование метода `TryParse` более предпочтительно, если необходимо определить, соответствует ли строка другому типу данных, все же в некоторых ситуациях вместо него следует использовать метод `Parse`. Одной из них является ситуация, при которой ожидается, что анализ будет успешным, и, следовательно, неудачный анализ действительно является исключительным условием, а снижение производительности проблемы не составляет. Другой важный случай — когда необходимо определить конкретные причины неудачного преобразования, например, различить исключительные ситуации, связанные с переполнением и с неверным форматом. Для обоих примеров весьма подходящим является метод `Parse`. Исключение `FormatException`, возникающее в результате неудачи метода `Parse`, означает, что неудача была неожиданной, и случай действительно является исключительным по определению.

Еще раз советую перечитать статью, ссылка на которую дана в предыдущем разделе.

Сравнение и сортировка строк

И, наконец, в указанной выше статье сказано несколько слов о сравнении строк. Строки в среде .NET являются довольно мощным инструментом в том смысле, что их можно сравнивать с учетом и без учета особенностей культурной среды. По умолчанию большинство строковых операций подобные особенности учитывают. Если нет необходимости сравнивать строки с учетом особенностей культурной среды, то для сравнения строк всегда следует использовать сравнение по порядковым номерам. Это же касается и сортировки строк. Экономия при использовании этого способа сортировки или сравнения получается значительной. Для класса `System.String` существуют специальные строковые методы, например, `String.CompareOrdinal`, подчеркивающие ценность такого подхода. Если вам не нужно учитывать регистр строк при сравнении, но при этом важно сохранить производительность, следует воспользоваться методом `StringComparer.OrdinalIgnoreCase`. Он работает намного быстрее, чем метод, учитывающий особенности культурной