

21 世纪高等院校计算机系列教材

影印版

C#程序设计

C# For Students

[英] Douglas Bell Mike Parr 著



中国水利水电出版社
www.waterpub.com.cn

**DOUGLAS BELL
MIKE PARR**

C#

FOR

Students



中国水利水电出版社
www.waterpub.com.cn

Copyright © Pearson Education Limited 2004.

This edition of C# FOR STUDENTS, First Edition is published by arrangement with Pearson Education Limited.

北京市版权局著作权合同登记号: 01-2005-4708

图书在版编目 (CIP) 数据

C#程序设计/(英)贝尔(Bell, D.), (英)帕尔(Parr, M.)著. —影印本. —北京: 中国水利水电出版社, 2006

(21世纪高等院校计算机系列教材)

书名原文: C# FOR Students

ISBN 7-5084-4107-9

I .C… II .①贝…②帕… III .C语言—程序设计
—高等学校—教材—英文 IV .TP312
中国版本图书馆CIP数据核字 (2006) 第116324号

书 名 C#程序设计

作 者 [英] Douglas Bell Mike Parr 著

出版 发行 中国水利水电出版社 (北京市三里河路6号 100044)

网址: www.waterpub.com.cn

E-mail: mchannel@263.net (万水)

sales@waterpub.com.cn

电话: (010) 63202266 (总机)、68331835 (营销中心)、82562819 (万水)

经 售 全国各地新华书店和相关出版物销售网点

排 版 北京万水电子信息有限公司

印 刷 北京市天竺颖华印刷厂

规 格 787mm × 1092mm 16开本 24.25印张 537千字

版 次 2006年10月第1版 2006年10月第1次印刷

印 数 0001—3000册

定 价 36.00元

凡购买我社图书, 如有缺页、倒页、脱页的, 本社营销中心负责调换

版权所有 · 侵权必究

Preface

● This book is for novices

If you have never done any programming before – if you are a complete novice – this book is for you. This book assumes no prior knowledge of programming. It starts from scratch. It is written in a simple, direct style for maximum clarity. It is aimed at first level students at universities and colleges, but it is also suitable for novices studying alone.

● Why C#?

C# is arguably one of the best programming languages to learn and use in the 21st century because:

- C# continues the tradition of the family of languages that includes C, C++ and Java.
- Object-oriented languages are the latest and most successful approach to programming. C# is completely object-oriented from the ground up.
- C# is a completely general-purpose language – anything that Visual Basic, C++ and Java can do, so can C#.
- C# gains most of its functionality from a library of components provided by the .NET framework.

● You will need . . .

To learn to program in C# you need a PC running Windows 2000, NT, XP or above and the software that allows you to prepare and run C# programs in a convenient way. There are two versions of the software provided by Microsoft – Visual C# .NET (for C# alone) and Visual Studio .NET (which supports both C# and other languages). This book comes with CDs containing a full version of Visual Studio .NET.

● The approach of this book

We explain how to use objects early in this book. Our approach is to start with the ideas of variables, assignment and methods, then use objects created from library classes. Next we explain how to use control structures for selection and looping. Then comes the treatment of how to

write your own classes.

We wanted to make sure that the fun element of programming was paramount, so we use graphics right from the start. We think graphics is fun, interesting and clearly demonstrates all the important principles of programming. But we haven't ignored programs that input and output text – they are also included.

The programs we present use many of the features of graphical user interfaces (GUIs), such as buttons and text boxes. But we also explain how to write console programs in C#.

We introduce new ideas carefully, one at a time rather than all at once. So, for example, there is a single chapter on writing methods. We introduce simple ideas early and more sophisticated ideas later on.

What's included?

This book explains the fundamental ideas of programming:

- variables;
- assignment;
- input and output using a graphical user interface (GUI);
- calculation;
- repetition;
- selection between alternatives.

It explains how to use numbers and character strings. Arrays are also described. These are all topics that are fundamental, whatever kind of programming you do. This book also thoroughly explains the object-oriented aspects of programming – using objects, writing classes, methods and properties, and using library classes. We also look at some of the more sophisticated aspects of object-oriented programming including inheritance, polymorphism and interfaces.

What's not included?

This book confines itself to the essentials of C#. It does not explain all the bits and pieces, the bells and whistles. Thus the reader is freed from unnecessary detail and can concentrate on mastering C# and programming in general.

UML

The Unified Modeling Language (UML) is the current mainstream notation for describing programs. We use elements of UML selectively, where appropriate, throughout this book.

Applications

Computers are used in many different applications and this book uses examples from all areas including:

- games;
- information processing;

- scientific calculations.

We have also included a few exercises which look at the exciting idea of artificial life.

The reader can choose to concentrate on those application areas of interest and ignore other areas.

Exercises are good for you

If you were to read this book time and again until you could recite it backwards, you still wouldn't be able to write programs. The practical work of writing programs is vital to becoming fluent and confident at programming.

There are exercises for the reader at the end of each chapter. Please do some of them to enhance your ability to program.

There are also short self-test questions throughout the text, so that you can check you have understood things properly. The answers are given at the end of each chapter.

Have fun

Programming is creative and interesting, particularly in C#. Please have fun!

Visit our website

The website includes:

- the text of all the programs in this book;
- a discussion forum for students;
- a bonus chapter covering the use of C# with databases;
- additional resources for instructors.

Our website can be reached via the Pearson Education website at:

<http://www.mikeparr.info/csharp1st/csabout.html>

<http://www.pearsoned.co.uk/HigherEducation/Booksby/BellParr/>.

Contents

1. The background to C#	1
2. The C# development environment	6
3. Introductory graphics	19
4. Variables and calculations	30
5. Methods and arguments	49
6. Using objects	78
7. Selection	96
8. Repetition	122
9. Debugging	138
10. Writing classes	147
11. Inheritance	166
12. Calculations	179
13. Data structures – list boxes and array lists	193
14. Arrays	203
15. Arrays – two-dimensional	222
16. String manipulation	233
17. Exceptions	251
18. Files	264
19. Console programs	284
20. Object-oriented design	297
21. Program style	318
22. Testing	329
23. Interfaces	341
24. Polymorphism	346
Appendices	357

Detailed contents

1. The background to C#	1
The history of C#	1
The Microsoft .NET framework	2
What is a program?	2
Programming principles	4
Programming pitfalls	4
Summary	4
Exercises	4
Answers to self-test questions	5
 2. The C# development environment	 6
Introduction	6
Installation and configuration	6
Creating a first program	7
Controls at design-time	10
Events and the <code>Button</code> control	11
Opening an existing project	13
Documenting property settings	13
Program errors	14
Editor facilities	14
The message box	15
Help	16
Programming principles	16
Programming pitfalls	16
Grammar spot	16
New language elements	17
New IDE facilities	17
Summary	17
Exercises	17
Answers to self-test questions	18
 3. Introductory graphics	 19
Introduction	19
Objects, methods, properties, classes – an analogy	19

A first drawing	20
Creating the program	21
The graphics coordinate system	21
Explanation of the program	22
Methods for drawing	23
Colours	25
The sequence concept and statements	26
Adding meaning with comments	27
Programming principles	28
Programming pitfalls	28
Grammar spot	28
New language elements	28
New IDE facilities	28
Summary	28
Exercises	28
Answers to self-test questions	29
 4. Variables and calculations	 30
Introduction	30
The nature of <code>int</code>	31
The nature of <code>double</code>	31
Declaring variables	31
The assignment statement	34
Calculations and operators	35
The arithmetic operators	36
The <code>%</code> operator	38
Joining strings with the <code>+</code> operator	39
Converting between strings and numbers	40
Text boxes and labels	41
Converting between numbers	43
The role of expressions	44
Programming principles	45
Programming pitfalls	45
Grammar spot	45
New language elements	46
New IDE facilities	46
Summary	46
Exercises	46
Answers to self-test questions	48
 5. Methods and arguments	 49
Introduction	49
Writing your own methods	49
A first method	50
Calling a method	52
Passing arguments	53
Parameters and arguments	54
A triangle method	54
Local variables	57

Name clashes	57
Event-handling methods	59
return and results	59
Building on methods	62
Passing arguments by reference	63
out and ref arguments	65
out – an example	65
ref – an example	67
A swap method with ref	68
this and objects	69
Overloading	70
Passing objects to methods	71
Programming principles	72
Programming pitfalls	72
Grammar spot	73
New language elements	73
New IDE facilities	73
Summary	74
Exercises	74
Answers to self-test questions	76
 6. Using objects	 78
Introduction	78
Instance variables	78
The form constructor	81
The <code>TrackBar</code> class	83
using and namespaces	85
Members, methods and properties	86
The <code>Random</code> class	87
The <code>Timer</code> class	90
Programming principles	92
Programming pitfalls	92
Grammar spot	93
New language elements	93
New IDE facilities	93
Summary	93
Exercises	93
Answers to self-test questions	95
 7. Selection	 96
Introduction	96
The if statement	96
if...else	98
Comparison operators	100
And, or, not	104
Nested ifs	107
switch	108
Boolean variables	112
Programming principles	115

Programming pitfalls	115
Grammar spot	115
New language elements	116
Summary	116
Exercises	116
Answers to self-test questions	119
8. Repetition	122
Introduction	122
while	122
for	126
And, or, not	127
do...while	129
Nested loops	130
Combining control structures	132
Programming principles	132
Programming pitfalls	132
Grammar spot	133
New language elements	133
Summary	133
Exercises	134
Answers to self-test questions	135
9. Debugging	138
Introduction	138
Using the debugger	139
Case study in debugging	141
Common errors	142
Programming pitfalls	145
New IDE facilities	146
Summary	146
Exercise	146
10. Writing classes	147
Introduction	147
Designing a class	147
private variables	150
public methods	150
Properties	151
Method or property?	154
Constructors	155
Multiple constructors	156
private methods	157
Operations on objects	157
Object destruction	158
static methods and properties	159
Programming principles	160
Programming pitfalls	161

Grammar spot	162
New language elements	163
Summary	163
Exercises	163
Answers to self-test questions	165
11. Inheritance	166
Introduction	166
Using inheritance	166
protected	168
Additional items	169
Overriding	169
Class diagrams	170
Inheritance at work	171
base	171
Constructors	172
Abstract classes	174
Programming principles	175
Programming pitfalls	176
New language elements	176
Summary	177
Exercises	177
Answers to self-test questions	178
12. Calculations	179
Introduction	179
Formatting numbers	179
Library mathematical functions and constants	182
Constants	182
Case study – money	183
Case study – iteration	185
Graphs	185
Exceptions	188
Programming principles	189
Programming pitfalls	189
Summary	189
Exercises	190
Answers to self-test questions	192
13. Data structures – list boxes and array lists	193
Introduction	193
Array lists	193
Adding items to a list	194
The length of a list	195
Indices	195
Removing items from a list	196
Inserting items within a list	197
Lookup	197

Arithmetic on a list box	198
Searching	200
Programming principles	201
Programming pitfalls	201
New language elements	201
Summary	201
Exercises	202
Answers to self-test questions	202
14. Arrays	203
Introduction	203
Creating an array	204
Indices	205
The length of an array	207
Passing arrays as parameters	207
Using constants	208
Initializing an array	209
A sample program	209
Lookup	211
Searching	212
Arrays of objects	213
Programming principles	213
Programming pitfalls	215
Grammar spot	216
Summary	216
Exercises	216
Answers to self-test questions	220
15. Arrays – two-dimensional	222
Introduction	222
Declaring an array	223
Indices	223
The size of an array	224
Passing arrays as parameters	225
Constants	225
Initializing an array	226
A sample program	227
Programming principles	228
Programming pitfalls	229
Summary	229
Exercises	229
Answers to self-test questions	232
16. String manipulation	233
Introduction	233
Using strings – a recap	233
String indexing	234
The characters within strings	235
A note on the <code>char</code> type	235

The <code>String</code> class methods and properties	236
Comparing strings	237
Amending strings	237
Examining strings	239
Regular expressions	242
An example of string processing	244
Case study – Frasier	245
Programming principles	247
Programming pitfalls	247
Grammar spot	248
New language elements	248
New IDE facilities	248
Summary	248
Exercises	249
Answers to self-test questions	250
17. Exceptions	251
Introduction	251
The jargon of exceptions	253
A <code>try-catch</code> example	253
Using the exception object	255
Classifying exceptions	256
Multiple catch blocks	257
The search for a catcher	257
Throwing – an introduction	259
Handling – some possibilities	260
finally	260
Programming principles	261
Programming pitfalls	261
Grammar spot	261
New language elements	262
New IDE facilities	262
Summary	262
Exercises	262
Answers to self-test questions	263
18. Files	264
Introduction	264
The essentials of streams	264
The <code>StreamReader</code> and <code>StreamWriter</code> classes	265
File output	265
File input	267
File searching	269
Files and exceptions	271
Message boxes and dialogs	272
Using file dialogs	274
Creating a menu	275
The <code>Directory</code> class	278

Programming principles	280
Programming pitfalls	280
Grammar spot	280
New language elements	281
New IDE facilities	281
Summary	281
Exercises	281
Answers to self-test questions	282

19. Console programs	284
Introduction	284
A first console program	284
The command prompt: <code>cd</code> and <code>dir</code>	286
Ways of running programs	288
Classes in console applications	289
Command-line arguments	289
Scripting and output redirection	291
Scripting and batch files	292
<i>Programming principles</i>	293
Programming pitfalls	293
Grammar spot	293
New language elements	293
New IDE facilities	293
Summary	294
Exercises	294
Answers to self-test questions	295

20. Object-oriented design	297
Introduction	297
The design problem	298
Identifying objects, methods and properties	298
Case study in design	302
Looking for reuse	309
Composition or inheritance?	309
Guidelines for class design	313
Summary	314
Exercises	315
Answers to self-test questions	316

21. Program style	318
Introduction	318
Program layout	319
Comments	320
Using constants	320
Classes	321
Nested <code>ifs</code>	322
Nested loops	324
Complex conditions	325

Documentation	327
Programming pitfalls	327
Summary	328
Exercises	328
22. Testing	329
Introduction	329
Program specifications	330
Exhaustive testing	330
Black box (functional) testing	331
White box (structural) testing	333
Inspections and walkthroughs	355
Stepping through code	336
Formal verification	336
Incremental development	336
Programming principles	337
Summary	337
Exercises	337
Answers to self-test questions	339
23. Interfaces	341
Introduction	341
Interfaces for design	341
Interfaces and interoperability	343
Programming principles	344
Programming pitfalls	344
New language elements	344
Summary	345
Exercises	345
24. Polymorphism	346
Introduction	346
Polymorphism in action	346
Casting	349
Rules for casting	350
Programming principles	352
Programming pitfalls	353
New language elements	353
Summary	354
Exercises	354
Answer to self-test question	355
Appendices	356
A Selected library components	356
B Keywords	371

The background to C#

This chapter explains:

- how and why C# came into being;
- Microsoft's .NET framework;
- the introductory concepts of programming.

● The history of C#

A computer program is a series of instructions that are obeyed by a computer. The point of the instructions is to carry out a task – e.g. play a game, send an e-mail, etc. The instructions are written in a particular style: they must conform to the rules of the programming language we choose. There are hundreds of programming languages, but only a few have made an impact and become widely used. The history of programming languages is a form of evolution, and here we will look at the roots of C# ('C Sharp'). The names of the older languages are not important, but we provide them for completeness.

Around 1960, a programming language named Algol 60 was created. ('Algol' from the term 'algorithm' – a series of steps that can be performed to solve a problem.) This was popular in academic circles, but its ideas persisted longer than its use. At this time, other languages were more popular: COBOL for data processing, and Fortran for scientific work. In the UK, an extended version of Algol 60 was created (CPL – combined programming language), which was soon simplified into basic CPL, or BCPL.

We then move to Bell Laboratories USA, where Dennis Ritchie and others transformed BCPL into a language named B, which was then enhanced to become C, around 1970. C was tremendously popular. It was used to write the UNIX operating system, and much later, Linus Torvalds used it to write a version of UNIX – named LINUX – for PCs.

The next step came when C++ ('C plus-plus') was created around 1980 by Stroustrup, also at Bell Labs. This made possible the creation and reuse of separate sections of code, in a style known as 'object-oriented programming'. (In C, you could use ++ to add one to an item – hence C++ is one up from C.)

C++ is still popular, but hard to use; it takes a lot of study. Around 1995, Sun Microsystems