A Natural Introduction to Computer Programming with C#

Kari Laitinen

www.naturalprogramming.com

Table of contents

INTRODUCTORY PAGES

Acknowledgments . . . iv
Some reasons to choose this book . . . v
Notes for teachers and other experienced programmers . . . v
'Some advice for studying . . . vii
The structure of this book . . . ix
The structure of program descriptions . . . x
Table of contents . . . xii
C# programs in this book . . . xvi
Pages where exercises can be found . . . xix

PART I: THE WORLD OF COMPUTERS . . . 1

1 COMPUTING: SOME CONCEPTS AND TERMINOLOGY . . . 3

1.1 Hardware and software . . . 4
1.2 Operating systems, main memory, and memory devices . . . 4
1.3 Source programs and executable programs . . . 8
1.4 Programs, applications, and systems . . . 9

2 A FIRST LOOK AT C# SOURCE PROGRAMS . . . 11

2.1 A program that can print text to the screen . . . 12
2.2 A program that can read from the keyboard and calculate . . . 14
2.3 Getting a C# compiler to your computer . . . 20
2.4 Installing the electronic material of this book . . . 23
2.5 Compiling with the C# compiler . . . 25
2.6 Finding an editor tool for writing computer programs . . . 26
2.7 Modifying and recompiling programs . . . 27

3 HOW INFORMATION IS STORED IN THE MEMORY OF A COMPUTER . . . 29

3.1 Numerical information: numbering systems . . . 30
3.2 Numerical information: the binary world of computers . . . 38
3.3 Textual information: character coding systems . . . 43
3.4 More information: pictures, sound, and moving pictures . . . 46

4 LOGICAL OPERATING PRINCIPLES OF COMPUTERS . . . 47

4.1 How does the main memory operate? . . . 48
4.2 The components of an imaginary computer . . . 52
4.3 Inside the imaginary processor . . . 54
4.4 Machine instructions . . . 57
4.5 The steps and states of program execution . . . 64
4.6 Programs to print text "Hello!" . . . 66
4.7 Programming language 1ML and compilation . . . 71
4.8 IC8 and ICOM - simulator programs for the imaginary computer . . . 78
4.9 A program that contains a loop . . . 80
4.10 Subroutine calls and stack operations . . . 82
4.11 Programs that use the keyboard, memory area, and stack . . . 85
4.12 Chapter summary - towards high-level programming . . . 92

PART II: FUNDAMENTALS OF PROGRAMMING 95

5 VARIABLES AND OTHER BASIC ELEMENTS IN C# PROGRAMS 97

5.1 Integer variables (int, short, long, byte, uint, ushort, ulong, sbyte, char) 98
5.2 Keywords, names, spaces, andnewlines 106
5.3 Floating-point variables 108
5.4 Operators, assignments, and literal constants Ill
5.5 Reading data from the keyboard - a first look at strings 116
5.6 The double role of operator+ 119
5.7 Formatting the output on the screen 123
5.8 Chapter summary 128

6 DECISIONS AND REPETITIONS: BASIC ACTIVITIES IN PROGRAMS 129

6.1 Making decisions with keywords if and else 130
6.2 Making decisions with switch-case constructs 143
6.3 while loops enable repetition 146
6.4 for loops repeat a known number of times 152
6.5 do-while loops execute at least once 158
6.6 The block structure of C# programs 161
6.7 try-catch constructs handle exceptions 165
6.8 Truth values and variables of type bool 168
6.9 Chapter summary 170

7 ARRAYS: SETS OF SIMILAR DATA ITEMS 171

7.1 Creating arrays and referring to array elements 172
7.2 Array declaration vs. array creation 183
7.3 Initialized arrays 185
7.4 Multidimensional arrays 190
7.5 Chapter summary 191

8 STRINGS STORE SEQUENCES OF CHARACTER CODES 193

8.1 "Variables" of type string 194
8.2 String literals 197
8.3 Accessing individual characters of a string 198
8.4 String methods 202
8.5 Class StringBuilder - mutable strings 222
8.6 Arrays of strings 227
8.7 Chapter summary 232

9 METHODS - LOGICAL PERFORMING UNITS IN PROGRAMS 235

9.1 Simple static methods and the concept of calling 236
9.2 Methods that take parameters 240
9.3 Methods that return data to the caller 250
9.4 Calling static methods of another class 260
9.5 The role of the stack in method calls 267
9.6 Scope of variables 273
9.7 Parameters for the method Main() 276
9.8 Overloading method names 280
9.9 Declaring and using namespaces 284
9.10 Chapter summary 286

PART III: OBJECT-ORIENTED PROGRAMMING 287

10 CLASSES AND OBJECTS 289

10.1 Classes, fields, and instance methods 290
10.2 Constructors are methods that build objects 300
10.3 Several constructors in a class 306
10.4 Arrays containing references to objects 310
10.5 Value types vs. reference types 324
10.6 When objects become garbage 325
10.7 A stack that grows dynamically 326
10.8 Chapter summary 333

11 MORE ADVANCED CLASSES 335

11.1 Class Date - an example of a larger class 336
11.2 The this keyword 356
11.3 Graphical UML class diagrams 359
11.4 "Objects inside objects" 360
11.5 Properties can replace accessor methods 373
11.6 Indexers provide array-like access 378
11.7 Chapter summary 386

12 INHERITANCE AND CLASS HIERARCHIES 387

12.1 Base classes and derived classes 388
12.2 Larger class hierarchies 404
12.3 Polymorphism -redefining methods in derived classes 418
12.4 Structs are class-like constructs that cannot be inherited 432
12.5 Chapter summary 437

13 SOME STANDARD C# CLASSES AND STRUCTS 439

13.1 Structs Byte, Intl6, Int32, Int64, Char, Single, Double, SByte, UIntl6, etc 440
13.2 Object: the class above all classes 442
13.3 Exception classes 446
13.4 Class Array: the base class for all arrays 452
13.5 Chapter summary 456

14 STORING INFORMATION IN FILES 457

14.1 Classes to read and write files 458
14.2 Reading and writing text files 460
14.3 Handling files as binary files 477
14.4 A larger program that uses a binary file 486
14.5 Chapter summary 504

15 MORE STANDARD C# TYPES 509

15.1 ArrayList class 510
15.2 IComparable and other interfaces 522
15.3 DateTime struct 529
15.4 Chapter summary 536

16 GOING CLOSER TO THE MACHINE 537

16.1 Bit operators &, |, ^{A}, ~. », and « 538
16.2 Playing with the time in programs - introduction to threads 546
16.3 Chapter summary 556

APPENDIX A: SUMMARY OF IMPORTANT C# FEATURES **561**

A - 1: Literals 561
A - 2: Variables, constants, and arrays of basic types 562
A - 3: String objects, other objects, and arrays of objects 563
A - 4: Expressions 564
A - 5: Assignments and left side expressions 564
A - 6: The most important C# operators in order of precedence 565
A - 7: Control structures to make decisions (selections) 566
A - 8: Control structures to perform repetitions (iterations) 567
A - 9: The basic C# method structures 568
A - 10: String methods 569
A - 11: Mechanisms for keyboard input and screen output 570
A - 12: Input/output from/to files 570
A - 13: Data conversions 571
A - 14: C# class declaration 572

APPENDIX B: C# KEYWORDS (RESERVED WORDS) **573**

APPENDIX C: PRACTICAL ADVICE FOR PROGRAMMING EXERCISES 577

C - **1**: Starting an exercise 577
C - 2: Writing your program 577
C - 3: Compiling and executing your program 578
C - 4: Correcting compilation errors 579
C - 5: Searching for errors that compilers do not detect 580
C - 6: Programs that do not terminate 581
C - 7: Incremental program development 582
C - 8: Printing your program on paper 583
C - 9: Program versions and backups 583

INDEX **585**

USEFUL TABLES **599**