

An Introduction to the C Programming Language
and Software Design

Tim Bailey

Preface

This textbook began as a set of lecture notes for a first-year undergraduate software engineering course in 2003. The course was run over a 13-week semester with two lectures a week. The intention of this text is to cover topics on the C programming language and introductory software design in sequence as a 20 lecture course, with the material in Chapters 2, 7, 8, 11, and 13 well served by two lectures apiece. Ample cross-referencing and indexing is provided to make the text a servicable reference, but more complete works are recommended. In particular, for the practicing programmer, the best available tutorial and reference is Kernighan and Ritchie [KR88] and the best in-depth reference is Harbison and Steele [HS95, HS02]. The influence of these two works on this text is readily apparent throughout.

What sets this book apart from most introductory C-programming texts is its strong emphasis on software design. Like other texts, it presents the core language syntax and semantics, but it also addresses aspects of program composition, such as function interfaces (Section 4.5), file modularity (Section 5.7), and object-modular coding style (Section 11.6). It also shows how to design for errors using `assert()` and `exit()` (Section 4.4). Chapter 6 introduces the basics of the software design process—from the requirements and specification, to top-down and bottom-up design, to writing actual code. Chapter 14 shows how to write generic software (i.e., code designed to work with a variety of different data types).

Another aspect that is not common in introductory C texts is an emphasis on bitwise operations. The course for which this textbook was originally written was prerequisite to an embedded systems course, and hence required an introduction to bitwise manipulations suitable for embedded systems programming. Chapter 12 provides a thorough discussion of bitwise programming techniques.

The full source code for all significant programs in this text can be found on the web at the address www.acfr.usyd.edu.au/homepages/academic/tbailey/index.html. Given the volatile nature of the web, this link may change in subsequent years. If the link is broken, please email me at tbailey@acfr.usyd.edu.au and I will attempt to rectify the problem.

This textbook is a work in progress and will be refined and possibly expanded in the future. No doubt there are errors and inconsistencies—both technical and grammatical—although hopefully nothing too seriously misleading. If you find a mistake or have any constructive comments please feel free to send me an email. Also, any interesting or clever code snippets that might be incorporated in future editions are most welcome.

Tim Bailey 2005.

Draft 0.6 (July 12, 2005)

TODO:

- complete Chapter 16
- complete appendices
- complete the index

Contents

Preface	i
Contents	ii
1 Introduction	1
1.1 Programming and Programming Languages	1
1.2 The C Programming Language	2
1.3 A First Program	3
1.4 Variants of Hello World	4
1.5 A Numerical Example	5
1.6 Another Version of the Conversion Table Example	6
1.7 Organisation of the Text	6
2 Types, Operators, and Expressions	8
2.1 Identifiers	8
2.2 Types	8
2.3 Constants	10
2.4 Symbolic Constants	11
2.5 printf Conversion Specifiers	12
2.6 Declarations	13
2.7 Arithmetic Operations	13
2.8 Relational and Logical Operations	14
2.9 Bitwise Operators	15
2.10 Assignment Operators	15
2.11 Type Conversions and Casts	16
3 Branching and Iteration	17
3.1 If-Else	17
3.2 ?: Conditional Expression	19
3.3 Switch	19
3.4 While Loops	20
3.5 Do-While Loops	21
3.6 For Loops	21
3.7 Break and Continue	22
3.8 Goto	23
4 Functions	25
4.1 Function Prototypes	25
4.2 Function Definition	25
4.3 Benefits of Functions	28
4.4 Designing For Errors	29

4.5	Interface Design	31
4.6	The Standard Library	32
5	Scope and Extent	33
5.1	Local Scope and Automatic Extent	33
5.2	External Scope and Static Extent	34
5.3	The <code>static</code> Storage Class Specifier	35
5.4	Scope Resolution and Name Hiding	36
5.5	Summary of Scope and Extent Rules	38
5.6	Header Files	38
5.7	Modular Programming: Multiple File Programs	39
6	Software Design	41
6.1	Requirements and Specification	41
6.2	Program Flow and Data Structures	42
6.3	Top-down and Bottom-up Design	42
6.4	Pseudocode Design	43
6.5	Case Study: A Tic-Tac-Toe Game	44
6.5.1	Requirements	44
6.5.2	Specification	44
6.5.3	Program Flow and Data Structures	45
6.5.4	Bottom-Up Design	45
6.5.5	Top-Down Design	47
6.5.6	Benefits of Modular Design	48
7	Pointers	49
7.1	What is a Pointer?	49
7.2	Pointer Syntax	50
7.3	Pass By Reference	52
7.4	Pointers and Arrays	53
7.5	Pointer Arithmetic	54
7.6	Return Values and Pointers	56
7.7	Pointers to Pointers	57
7.8	Function Pointers	57
8	Arrays and Strings	59
8.1	Array Initialisation	59
8.2	Character Arrays and Strings	60
8.3	Strings and the Standard Library	62
8.4	Arrays of Pointers	63
8.5	Multi-dimensional Arrays	65
9	Dynamic Memory	68
9.1	Different Memory Areas in C	68
9.2	Standard Memory Allocation Functions	69
9.3	Dynamic Memory Management	70
9.4	Example: Matrices	72
9.5	Example: An Expandable Array	75

10 The C Preprocessor	79
10.1 File Inclusion	79
10.2 Symbolic Constants	79
10.3 Macros	80
10.3.1 Macro Basics	81
10.3.2 More Macros	82
10.3.3 More Complex Macros	83
10.4 Conditional Compilation	84
11 Structures and Unions	86
11.1 Structures	86
11.2 Operations on Structures	87
11.3 Arrays of Structures	88
11.4 Self-Referential Structures	89
11.5 Typedefs	91
11.6 Object-Oriented Programming Style	93
11.7 Expandable Array Revisited	94
11.8 Unions	97
12 Bitwise Operations	99
12.1 Binary Representations	99
12.2 Bitwise Operators	100
12.2.1 AND, OR, XOR, and NOT	100
12.2.2 Right Shift and Left Shift	101
12.2.3 Operator Precedence	102
12.3 Common Bitwise Operations	102
12.4 Bit-fields	103
13 Input and Output	105
13.1 Formatted IO	105
13.1.1 Formatted Output: <code>printf()</code>	105
13.1.2 Formatted Input: <code>scanf()</code>	107
13.1.3 String Formatting	109
13.2 File IO	109
13.2.1 Opening and Closing Files	109
13.2.2 Standard IO	110
13.2.3 Sequential File Operations	110
13.2.4 Random Access File Operations	112
13.3 Command-Shell Redirection	113
13.4 Command-Line Arguments	114
14 Generic Programming	115
14.1 Basic Generic Design: Typedefs, Macros, and Unions	115
14.1.1 Typedefs	115
14.1.2 Macros	116
14.1.3 Unions	116
14.2 Advanced Generic Design: <code>void *</code>	117
14.2.1 Case Study: Expandable Array	117
14.2.2 Type Specific Wrapper Functions	121
14.2.3 Case Study: <code>qsort()</code>	123

[Click here to download full PDF material](#)