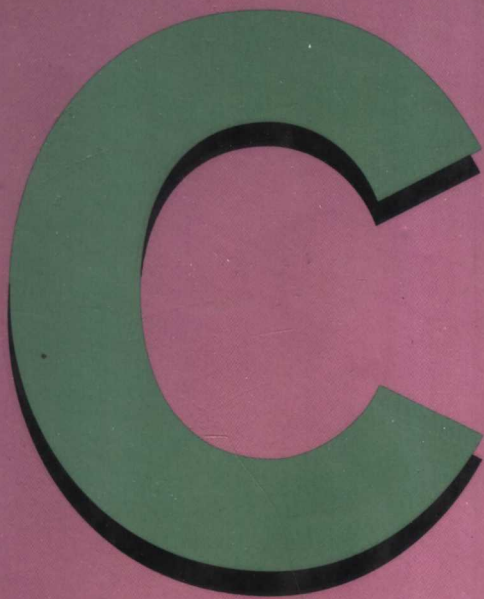


Mastering



Anthony Rudd



Mastering C

Anthony Rudd



A Wiley-QED Publication

John Wiley & Sons, Inc.

New York • Chichester • Brisbane • Toronto • Singapore

Designations used by companies to distinguish their products are often claimed as trademarks. In all instances where John Wiley & Sons, Inc. is aware of a claim, the product names appear in initial capital or all capital letters. Readers, however, should contact the appropriate companies for more complete information regarding trademarks and registration.

This text is printed on acid-free paper.

© 1994 by John Wiley & Sons, Inc.

All rights reserved.

This publication is designed to provide accurate and authoritative information in regard to the subject matter covered. It is sold with the understanding that the publisher is not engaged in rendering legal, accounting, or other professional services. If legal advice or other expert assistance is required, the services of a competent professional person should be sought. FROM A DECLARATION OF PRINCIPLES JOINTLY ADOPTED BY A COMMITTEE OF THE AMERICAN BAR ASSOCIATION AND A COMMITTEE OF PUBLISHERS.

Reproduction or translation of any part of this work beyond that permitted by section 107 or 108 of the 1976 United States Copyright Act without the permission of the copyright owner is unlawful. Requests for permission or further information should be addressed to the Permissions Department, John Wiley & Sons, Inc.

ISBN 0 471-60820-3

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

Preface

I have nothing to offer but blood, toil, tears and sweat.

Winston Churchill

The title of this book, *Mastering C*, with its subtitle, *A Practical Reference and Tutorial for ANSI C*, reveals the purpose I had in writing it: the book should be a concise, complete, practical reference book that can be used to master C. This book is not intended to be an introductory book for programming novices (nonprogrammers); I – and I am not alone here – do not consider C to be a suitable language with which to learn programming (a language like REXX or Pascal is a better introductory programming language). A basic assumption of C is that the programmer knows what he is doing, which cannot be said of someone new to programming. A note to my female readers: I use *he* as a generic third-person pronoun; I do not intend to imply that there are only masculine programmers; a woman, Ada Lovelace, is usually acknowledged to have been the first programmer.

I intend this book to satisfy the requirements of programmers new to C and those C programmers still mystified by the language. I would classify myself as having belonged to this second group before I started to write this book; I had attended a C course and had written several C programs, but I did not feel comfortable with the language. Many of the problems I encountered in learning the C language are mentioned in this book.

This book is divided into three parts:

- Part 1 – the C Language Elements
- Part 2 – the Standard C Library
- Part 3 – C in Practice

It is probably not possible to write a reference book on a complex programming language that can be read sequentially, and I have certainly not succeeded. In any case, research has shown that such books are not normally so read, rather, those parts of

immediate interest are read. I have tried to write this book so that the individual sections are easy to find; to avoid unnecessary jumping between sections, some important information is repeated.

From my personal experience, a major difficulty in learning a new programming language is how the elements are put together. With this in mind, an example is shown for each new concept and a worked example is provided at the end of most chapters. In the interest of keeping the book as compact as possible, the section examples are usually only code fragments, i.e., they are not executable programs. The worked examples are complete, and have been chosen to illustrate the information provided in the chapter but not to be overloaded with unnecessary detail. The solutions I offer in the worked examples are illustrative, and are not the only possible solutions nor even necessarily the best solutions; the reader is invited to produce his own (better) solution.

What is the target objective of this book? It should provide a compact reference to Standard C (ANSI and ISO) and show how the language is used.

This book does not provide information for a particular operating environment – such topics warrant a book in their own right – instead C-specific information is provided for all common hardware platforms (for example, Appendix B contains a table with both ASCII and EBCDIC codes). Similarly, unlike the authors of many programming books, I have not presented complete applications (which are not usually read anyway); rather, I have tried to provide the foundations with which such applications can be built.

At this point I would like to offer my thanks to Elke Berger, Norman Goldberg, Wolfgang Lauterbach, and Syed Mohamed for their assistance and suggestions for improvement.

Contents

Preface		xvii
----------------	-------	------

Part 1: The C Language Elements

1. Introduction

1.1	History	3
1.2	Simple C Program	5
1.3	Preparation Phases of a C Program.....	6
1.4	Execution Environment.....	7

2. C Program Elements

2.1	Introduction	9
2.2	C Character Set	10
2.2.1	Other Character Sets	10
2.3	Token Parsing	10
2.4	Block	11
2.5	Expression	12
2.6	Operators	13
2.6.1	Operator Processing	15
2.6.2	Order of Evaluation	15
2.6.3	Arithmetic Operators	16
2.6.4	Relational Operators	17
2.6.5	Logical Operators	17
2.6.6	Bit Operators	18
2.6.7	NOT Operator.....	20
2.6.8	Address and Indirection Operators.....	20
2.6.9	Reference Operators	21
2.6.10	Conditional Operator	21
2.6.11	Comma Operator	22
2.6.12	Array Subscript Operator	23
2.6.13	Cast Operator	23
2.6.14	Assignment Operators.....	24
2.6.15	sizeof Compile-Time Operator.....	25
2.7	Punctuators	25

2.7.1	Delimiters	26
2.7.2	Comment	26
2.8	Special Characters.....	26
2.8.1	Escape Sequences	26
2.8.2	White-Space Characters.....	27
2.8.3	Trigraphs	28
2.9	Object (Lvalue)	28
2.9.1	Identifier	29
2.10	Data Declarations (Definitions).....	30
2.10.1	Type.....	30
2.10.2	Scope.....	31
2.10.3	Linkage	32
2.10.4	Storage Duration.....	33
2.10.5	Storage Class.....	34
2.10.6	Qualifiers	35
2.10.7	Example.....	36
2.11	Data Types	36
2.11.1	Integer.....	37
2.11.2	Floating-Point	39
2.11.3	Character.....	42
2.12	Constants	43
2.12.1	Numeric Constants.....	43
2.12.2	Character Constant	45
2.12.3	String Constant (Literal).....	46
2.13	Data Aggregates	47
2.13.1	enum – Enumeration.....	47
2.13.2	struct – Structure.....	47
2.13.3	union – Union.....	47
2.13.4	Array.....	47
2.14	Initialization	48
2.15	Conversions	48
2.16	Orthogonality (Use of Expressions Within Expressions)	50

3. Program Statements

3.1	Introduction	53
3.1.1	Conditional Expressions.....	53
3.1.2	Nested Expressions.....	54
3.2	C Statements	54
3.2.1	break – Terminate Control Block.....	54
3.2.2	continue – Terminate Current Iteration of Loop.....	56
3.2.3	do ... while – Loop Tested at Bottom	56
3.2.4	for – Iterative Loop.....	57
3.2.5	goto – Unconditional Branch.....	58
3.2.6	if (else) – Conditional Processing.....	59

3.2.7	return – Explicit Return from a Function	60
3.2.8	switch – Conditional Processing Block	61
3.2.9	while – Loop Tested at Top	63
3.2.10	Expression Statement	64
3.2.11	{ } – Program Block (Compound Statement)	64
3.3	Equivalence of Statements and Expressions	65
3.4	Worked Example	65
3.4.1	Specification	66
3.4.2	Program Code	66

4. Declarations

4.1	Introduction	69
4.1.1	Consistency of Declarations	70
4.1.2	Function Prototypes	70
4.2	Declaration (Definition)	70
4.2.1	Storage Class	71
4.2.2	Type	72
4.2.3	Qualifier	74
4.2.4	Pointer	74
4.2.5	Identifier	75
4.2.6	Function Parameter List	75
4.2.7	Subscript Declarator	75
4.2.8	Initializer	76
4.3	Incomplete Declaration	76
4.4	Parentheses in Declarations	77
4.5	Functions in Declarations	77
4.6	typedef	77
4.7	Interpretation of Declarations	78
4.7.1	Rules for the Interpretation of Declarations	79
4.7.2	Function Declaration	80
4.7.3	Complex Declarations	81
4.8	Creation of Declarations	81
4.8.1	Use of typedef to Simplify Declarations	82
4.9	Worked Example	83
4.9.1	Specification	84
4.9.2	Program Code	84

5. Preprocessor

5.1	Introduction	87
5.1.1	Processing of Preprocessor Directives	87
5.2	Preprocessor Translation Phases	88
5.2.1	Preprocessing Token	89
5.2.2	Comment	90
5.3	Preprocessor Directives	90

5.3.1	# – Null-Directive	91
5.3.2	#define – Define Macro.....	92
5.3.3	#elif – Else-If Directive	95
5.3.4	#else – Introduce Block of Statements to be Processed If the Preceding Conditional Directive Is Not Satisfied	96
5.3.5	#endif – Terminate Block.....	96
5.3.6	#error – Terminate Compilation	97
5.3.7	#if – Test Preprocessor Condition	97
5.3.8	#ifdef – Test Whether Macro-Name Defined	98
5.3.9	#ifndef – Test Whether Macro-Name Not Defined	99
5.3.10	#include – Include Header File	99
5.3.11	#line – Modify Line Number and File Name	100
5.3.12	#pragma – Implementation-Specific Action	100
5.3.13	#undef – Remove Macro Definition	101
5.4	Predefined Macros.....	101
5.4.1	__DATE__ – Compilation Date	102
5.4.2	__FILE__ – Source File Name	102
5.4.3	__LINE__ – Line Number	102
5.4.4	__STDC__ – Standard C.....	103
5.4.5	__TIME__ – Compilation Time.....	103
5.5	Preprocessor Operators	104
5.5.1	# – Convert-to-String Operator	104
5.5.2	## – Concatenation Operator.....	104
5.6	Avoiding Multiple Definitions.....	104
5.7	Testing Preprocessor Directives	105
5.8	Use of Program-Oriented Headers	105
5.9	Worked Example.....	106
5.9.1	Specification.....	106
5.9.2	Macro Example (Version 1).....	106
5.9.3	Macro Example (Version 2).....	107

6. Functions

6.1	Introduction	111
6.1.1	Comparison of Functions and Macros	111
6.2	Function Definition.....	112
6.3	Function Call (Invocation).....	115
6.3.1	Recursive Function	116
6.4	Function Prototype	117
6.4.1	Interpreting Function Prototypes.....	118
6.5	Pointer to Function	118
6.6	Passing Arguments to Functions.....	119
6.6.1	Passing Arrays	121
6.6.2	Argument Promotion.....	123
6.7	Function Termination.....	124

6.8	Function Return Value.....	124
6.9	Function Redefinition.....	125
6.10	Executable Program.....	126
6.10.1	Program Startup.....	126
6.10.2	Program Termination.....	127
6.11	Worked Example.....	128
6.11.1	Specification.....	128
6.11.2	Sample Output.....	128
6.11.3	Program Code.....	128

7. Data Aggregates

7.1	Introduction.....	131
7.2	Array.....	131
7.2.1	Array Declaration.....	133
7.2.2	Array Addressing.....	135
7.2.3	Character Strings.....	138
7.2.4	Pointers to Arrays.....	138
7.2.5	Passing an Array to a Function.....	139
7.3	Structure.....	140
7.3.1	Structure Declaration.....	140
7.3.2	Complex Structures.....	143
7.3.3	Addressing Structure Members.....	144
7.4	Union.....	146
7.4.1	Union Declaration.....	147
7.4.2	Complex Unions.....	149
7.5	Enumeration.....	149
7.6	Worked Example.....	151

8. Pointers

8.1	Introduction.....	153
8.2	Pointer Operators.....	153
8.2.1	Address-of Operator (&).....	153
8.2.2	Indirection Operator (*).....	154
8.2.3	Array Pointers.....	154
8.3	Pointer Variables.....	155
8.3.1	NULL.....	156
8.4	Pointer Arithmetic.....	156
8.4.1	Pointer to void.....	157
8.5	Arrays of Pointers.....	157
8.6	Pointers to Pointers.....	158
8.7	Pointers to Functions.....	159
8.8	Worked Example.....	160
8.8.1	alloc - Simple Version.....	161
8.8.2	alloc - Portable Version.....	164

Part 2: The Standard C Library

9. Standard Library

9.1	Introduction	171
9.1.1	Naming Conventions.....	172
9.1.2	Header Files.....	172
9.1.3	Functions and Macros	173
9.2	Standard Include Files.....	173
9.2.1	Standard Definitions.....	175
9.3	Input/Output (I/O).....	176
9.3.1	Standard Files.....	177
9.3.2	File Management.....	177
9.3.3	File Buffering.....	178
9.3.4	Data Transfer.....	178
9.3.5	Formatted I/O.....	179
9.3.6	File Positioning.....	179
9.3.7	File Processing Summary	180
9.4	String Processing.....	182
9.5	Integer Arithmetic.....	184
9.6	Floating-Point and Trigonometrical Functions.....	184
9.7	String Conversion Functions.....	185
9.8	Character Classification	186
9.9	Time and Date Functions.....	187
9.10	Variable Parameter Lists	187
9.11	Storage Allocation Functions	188
9.12	Nonlocal Jump Functions.....	189
9.13	Error Functions	190
9.14	Signal Processing.....	191
9.15	Internationalization.....	191
9.16	Communication with the Environment	193
9.17	Sorting and Searching Functions.....	193
9.18	Multibyte-Character Functions	194
9.19	Multibyte String Functions.....	194
9.20	Standard Macro Function	194

10. ANSI Library Functions

10.1	Introduction	197
10.2	Function Definitions	197
10.2.1	abort - Terminate Program Immediately.....	201
10.2.2	abs - (Integer) Absolute Value	201
10.2.3	acos - Arc Cosine.....	202
10.2.4	asctime - Convert Time Structure to String.....	202
10.2.5	asin - Arc Sine.....	203
10.2.6	assert - Write Error Information Conditionally to stderr	203

10.2.7	atan – Arc Tangent.....	204
10.2.8	atan2 – Arc Tangent of Quotient.....	204
10.2.9	atexit – Establish Function to be Invoked on Program Termination.....	205
10.2.10	atof – Alphanumeric to Floating-Point Conversion.....	206
10.2.11	atoi – Alphanumeric to Integer Conversion.....	206
10.2.12	atol – Alphanumeric to Long (Integer) Conversion.....	207
10.2.13	bsearch – Binary Search.....	208
10.2.14	calloc – Allocate and Clear Memory.....	209
10.2.15	ceil – Round Floating-Point Number Up.....	209
10.2.16	clearerr – Clear Error.....	210
10.2.17	clock – Get Internal Timer Value.....	210
10.2.18	cos – Cosine.....	211
10.2.19	cosh – Hyperbolic Cosine.....	211
10.2.20	ctime – Convert Time Value to String.....	212
10.2.21	difftime – Determine Difference Between Two Times.....	212
10.2.22	div – Divide.....	213
10.2.23	exit – Terminate Program.....	214
10.2.24	exp – Exponential Function.....	214
10.2.25	fabs – Floating-Point Absolute Value.....	214
10.2.26	fclose – Close File.....	215
10.2.27	feof – Test for End-of-File.....	215
10.2.28	ferror – Test Error Flag.....	216
10.2.29	fflush – Flush Contents of Buffer.....	217
10.2.30	fgetc – Read Character.....	217
10.2.31	fgetpos – Store the Current File Position.....	218
10.2.32	fgets – Read a String.....	218
10.2.33	floor – Round Floating-Point Number Down.....	219
10.2.34	fmod – Floating-Point Remainder.....	219
10.2.35	fopen – Open File.....	220
10.2.36	fprintf – Write Formatted Output to a File.....	221
10.2.37	fputc – Put Character.....	224
10.2.38	fputs – Put String.....	224
10.2.39	fread – Read Records.....	225
10.2.40	free – Release Storage Block.....	226
10.2.41	freopen – Reopen File.....	226
10.2.42	frexp – Convert Floating-Point Value into Exponential Form.....	227
10.2.43	fscanf – Read Formatted Data from File.....	228
10.2.44	fseek – Set Relative File Position.....	230
10.2.45	fsetpos – Set Absolute File Position.....	231
10.2.46	ftell – Obtain the Current File Position.....	232
10.2.47	fwrite – Write Records.....	232
10.2.48	getc – Read Character.....	233
10.2.49	getchar – Read Character from stdin.....	234

10.2.50	getenv – Return Environmental Information.....	234
10.2.51	gets – Read a String from stdin.....	235
10.2.52	gmtime – Convert Time to Greenwich Mean Time (GMT).....	235
10.2.53	isalnum – Test for Alphanumeric Character.....	236
10.2.54	isalpha – Test for Alphabetic Character.....	236
10.2.55	iscntrl – Test for Control Character.....	237
10.2.56	isdigit – Test for Decimal Digit.....	237
10.2.57	isgraph – Test for Graphical (True Printable) Character.....	238
10.2.58	islower – Test for Lowercase Character.....	238
10.2.59	isprint – Test for Printable Character.....	239
10.2.60	ispunct – Test for Punctuation Character.....	239
10.2.61	isspace – Test for White-Space Character.....	240
10.2.62	isupper – Test for Uppercase Character.....	240
10.2.63	isxdigit – Test for Hexadecimal Digit.....	241
10.2.64	labs – Long Absolute Value.....	241
10.2.65	ldexp – Multiply by Power of 2.....	242
10.2.66	ldiv – Long Division.....	242
10.2.67	localeconv – Get Location Conventions.....	243
10.2.68	localtime – Convert Time to Local Time.....	243
10.2.69	log – Natural Logarithm.....	244
10.2.70	log10 – Logarithm.....	244
10.2.71	longjmp – Resume Program Execution at Last Saved Environment.....	245
10.2.72	malloc – Allocate Main-Storage Block.....	246
10.2.73	mblen – Determine Length of Multibyte-Character String.....	246
10.2.74	mbstowcs – Convert Multibyte-Characters to Wide-Characters.....	247
10.2.75	mbtowc – Convert Multibyte-Character to Wide-Character.....	247
10.2.76	memchr – Search Memory for Characters.....	248
10.2.77	memcmp – Compare Memory.....	248
10.2.78	memcpy – Copy Memory.....	249
10.2.79	memmove – Move Memory.....	250
10.2.80	memset – Set Memory to Specified Character.....	250
10.2.81	mktime – Make Time.....	251
10.2.82	modf – Convert Floating-Point Value into Integral and Fractional Form.....	252
10.2.83	perror – Print Error Message.....	252
10.2.84	pow – Compute Power.....	253
10.2.85	printf – Display Formatted Output.....	253
10.2.86	putc – Put Character.....	256
10.2.87	putchar – Display Character.....	256
10.2.88	puts – Display String.....	257
10.2.89	qsort – Quick Sort.....	257
10.2.90	raise – Raise Signal Condition.....	258
10.2.91	rand – Pseudo-Random Number.....	259

10.2.92	realloc – Change Size of Allocated Memory	259
10.2.93	remove – Delete File	260
10.2.94	rename – Rename File	261
10.2.95	rewind – Position File Pointer at Start of File	261
10.2.96	scanf – Read Formatted Data from Standard Input	262
10.2.97	setbuf – Specify Buffer	264
10.2.98	setjmp – Save Environment	265
10.2.99	setlocale – Set Location Information	266
10.2.100	setvbuf – Specify Buffer Mode	267
10.2.101	signal – Specify Interrupt Processing	268
10.2.102	sin – Sine	269
10.2.103	sinh – Hyperbolic Sine	270
10.2.104	sprintf – Store Formatted Output in a Buffer	270
10.2.105	sqrt – Square Root	272
10.2.106	srand – Seed Pseudo-Random Number Process	273
10.2.107	sscanf – Get Formatted Data from Buffer	273
10.2.108	strcat – Concatenate String	276
10.2.109	strchr – Scan String for Character	276
10.2.110	strcmp – Compare Two Strings	277
10.2.111	strcoll – Collate-Oriented String Compare	278
10.2.112	strcpy – Copy String	278
10.2.113	strcspn – Scan String for First Character Contained in Specified String	278
10.2.114	strerror – Get Message Text	279
10.2.115	strftime – Formatted Time Conversion	280
10.2.116	strlen – String Length	281
10.2.117	strncat – Concatenate Bounded Strings	282
10.2.118	strncmp – Compare Two Bounded Strings	283
10.2.119	strncpy – Copy Bounded Strings	283
10.2.120	strpbrk – Scan String for Specified (Break) Character	284
10.2.121	strrchr – Scan String for Last Occurrence of Specified Character	285
10.2.122	strspn – Scan String for First Character Not Contained in Specified String	285
10.2.123	strstr – Search String for Substring	286
10.2.124	strtod – String to double (Floating-Point) Conversion	287
10.2.125	strtok – Get Token from String	288
10.2.126	strtol – String to Long (Integer) Conversion	289
10.2.127	strtoul – String to Unsigned Long Conversion	289
10.2.128	strxfrm – Transform String	291
10.2.129	system – Perform Operating System Command	291
10.2.130	tan – Tangent	292
10.2.131	tanh – Hyperbolic Tangent	292
10.2.132	time – Get Current Time	292

10.2.133 tmpfile – Create a Temporary File293

10.2.134 tmpnam – Generate a Temporary (Unique) File Name294

10.2.135 tolower – Convert Character to Lowercase295

10.2.136 toupper – Convert Character to Uppercase295

10.2.137 ungetc – Return Character to Input Stream.....296

10.2.138 va_arg – Get Next Argument in List297

10.2.139 va_end – Terminate Variable-Length Argument List297

10.2.140 va_start – Start Variable-Length Argument List.....298

10.2.141 vfprintf – Write Formatted Output to a File Using Pointer
to an Argument List.....299

10.2.142 vprintf – Display Formatted Output Using Pointer to an
Argument List.....300

10.2.143 vsprintf – Store Formatted Output in a Buffer Using Pointer
to an Argument List.....301

10.2.144 wctombs – Convert Wide-Characters to Multibyte-Characters.....301

10.2.145 wctomb – Convert Wide-Character to Multibyte-Character302

10.3 Macro Functions302

10.3.1 offsetof – Determine Offset of Member in a Structure302

10.4 Worked Example.....303

10.4.1 Specification.....303

10.4.2 Program Code.....304

Part 3: C in Practice

11. Portability

11.1 Introduction309

11.2 Application Portability309

11.3 C Language Provisions for Portability.....310

11.4 General Considerations.....310

11.5 Worked Example.....311

12. C Culture

12.1 Introduction313

12.2 Coding Style316

12.3 Summary.....317

13. Common Problems in C

13.1 Introduction319

13.2 Arrays.....320

13.2.1 Processing Array Elements.....320

13.2.2 Array Indexing.....320

13.2.3 Assignment Using an Array Name321

13.2.4 Difference Between Arrays and Pointers321

13.3 Operators321

13.3.1	= and -- Operators	321
13.3.2	Precedence	322
13.3.3	Unexpected Conversions	323
13.4	Conditional Statements	323
13.4.1	if, else Statements	323
13.4.2	for Loop	323
13.5	Pointers	324
13.5.1	Dangling Pointers	324
13.5.2	Nonassigned Storage	325
13.5.3	Array Overrun	325
13.5.4	Contiguous Storage	326
13.6	Header Files	327
13.7	Language Inconsistencies	327
13.7.1	Difference Between Initialization and Assignment	327
13.7.2	Comma Operator	327
13.8	Floating-Point Arithmetic	328
13.8.1	Noncommutativity	328
13.8.2	Nonequality	328

14. Worked Example

14.1	Introduction	331
14.2	Specification	331
14.3	Solution	332
14.4	Reader's Exercise	333
14.4.1	One Solution for the Reader's Exercise	334

Appendix A. Syntax Notation	335
Appendix B. Code Table	339
Appendix C. Reserved Words	345
Appendix D. Bibliography	351
Appendix E. Glossary	353
Appendix F. printf/scanf Format Codes	357
Index	361

Part 1

The C Language Elements