
Contents

<i>Preface</i>	<i>vii</i>
<i>Contents in Brief</i>	<i>ix</i>
CHAPTER 1 Introducing Computers and Hardware	1
1.1 The Computer Boom	1
1.2 Computer Basics	2
1.3 The Background	4
1.3.1 Napier, Pascal and Leibniz: The Early Pioneers	4
1.3.2 The Programmable Computer	4
1.3.3 The Programmable Electronic Computer	5
1.4 Computer Generations	6
1.4.1 Vacuum Tubes: The First Generation	6
1.4.2 Transistors: The Second Generation	6
1.4.3 Integrated Circuits: The Third Generation	7
1.4.4 The Microprocessor: The Fourth Generation	7
1.4.5 Artificial Intelligence: The Fifth Generation	8
1.5 Computer Types	8
1.5.1 Supercomputers	9
1.5.2 Mainframes	9
1.5.3 Minicomputers	9
1.5.4 Microcomputers	10
1.5.5 Smartphones and Embedded Computers	10
1.6 Bits, Bytes and Words	11
1.7 Inside the Computer	12
1.8 The Central Processing Unit (CPU)	13
1.9 Primary Memory	14
1.9.1 Random Access Memory (RAM)	14
1.9.2 Read Only Memory (ROM)	15
1.9.3 Cache Memory	16
1.9.4 Registers	16
1.10 Secondary Memory	17
1.10.1 The Hard Disk	17
1.10.2 Magnetic Tape	19

1.10.3	Optical Disks: The CD-ROM, DVD-ROM and Blu-Ray Disk	19
1.10.4	Flash Memory	20
1.10.5	Floppy Diskette	21
1.11	Ports and Connectors	21
1.12	Input Devices	23
1.12.1	The Keyboard	23
1.12.2	Pointing Devices	23
1.12.3	The Scanner	24
1.13	Output Devices	25
1.13.1	The Monitor	25
1.13.2	Impact Printers	26
1.13.3	Non-Impact Printers	27
1.13.4	Plotters	27
1.14	Computers in a Network	28
1.14.1	Network Topology	28
1.14.2	Network Types	29
1.14.3	The Internet and internet	30
1.15	Network Hardware	30
1.15.1	Network Interface Card	30
1.15.2	Hub and Switch	31
1.15.3	Bridge and Router	31

CHAPTER 2 Computer Software

36

2.1	Why Computers Need Software	36
2.2	Software Basics	37
2.3	Software Types	38
2.3.1	System Software	38
2.3.2	Application Software	39
2.4	Basic Input Output System (BIOS)	40
2.5	The Operating System (OS)	41
2.5.1	How Users Interact with an Operating System	42
2.5.2	Classification of Operating Systems	43
2.5.3	The Current Operating Systems	45
2.6	Device Drivers	45
2.7	MSDOS: A CUI Operating System	47
2.7.1	Files and the File System	47
2.7.2	Internal and External Commands	48
2.7.3	Working with Files and Directories	49
2.7.4	Using Wild-Cards with Files	50
2.7.5	The Utilities	50
2.8	Windows: A GUI Operating System	51
2.8.1	The Display and Mouse	51
2.8.2	File Explorer: Manipulating Files and Directories	52
2.8.3	Utilities and Administrative Tools	54

- 2.9 UNIX/Linux: A Programmer's Operating System 55
 - 2.9.1 The UNIX File System 56
 - 2.9.2 Basic Directory and File Handling Commands 56
 - 2.9.3 File Ownership and Permissions 57
 - 2.9.4 **grep** and **find**: The UNIX Search Commands 58
 - 2.9.5 Other Utilities 59
- 2.10 Microsoft Word 60
 - 2.10.1 Creating and Saving a Document 61
 - 2.10.2 Handling Text 62
 - 2.10.3 Font Handling 64
 - 2.10.4 Inserting Tables 64
 - 2.10.5 Inserting Graphics 65
 - 2.10.6 Other Useful Utilities 66
- 2.11 Microsoft Excel 66
 - 2.11.1 Creating, Saving and Printing a Spreadsheet 67
 - 2.11.2 Formatting Cells 68
 - 2.11.3 Computing and Using Formulas 69
 - 2.11.4 Handling Data 70
 - 2.11.5 Creating Charts 73
- 2.12 Microsoft Powerpoint 74
- 2.13 Understanding a TCP/IP Network 75
 - 2.13.1 Hostnames and IP Addresses 75
 - 2.13.2 The TCP/IP Model—The Four Layers 76
 - 2.13.3 DNS: Resolving Domain Names on the Internet 77
- 2.14 Internet Applications 77
 - 2.14.1 Electronic Mail 77
 - 2.14.2 Telnet: Remote Login 78
 - 2.14.3 Ftp: Transferring Files 79
 - 2.14.4 The Secure Shell 79
 - 2.14.5 The World Wide Web 79

CHAPTER 3 Computing and Programming Concepts

85

- 3.1 Numbering Systems 85
- 3.2 The Binary Numbering System 87
 - 3.2.1 Converting from Binary to Decimal 87
 - 3.2.2 Converting from Decimal to Binary 88
 - 3.2.3 Binary Coded Decimal (BCD) 88
- 3.3 Negative Binary Numbers 89
 - 3.3.1 Sign-and-Magnitude Representation 89
 - 3.3.2 One's Complement 89
 - 3.3.3 Two's Complement 90
- 3.4 Binary Arithmetic 91
 - 3.4.1 Binary Addition 91

- 3.4.2 Binary Subtraction (Regular Method) 92
- 3.4.3 Binary Subtraction Using Two's Complement 92
- 3.4.4 Binary Multiplication 93
- 3.4.5 Binary Division 94
- 3.5 The Octal Numbering System (Base-8) 95
 - 3.5.1 Converting from Octal to Decimal 95
 - 3.5.2 Converting from Decimal to Octal 95
 - 3.5.3 Converting from Octal to Binary 96
 - 3.5.4 Converting from Binary to Octal 96
- 3.6 The Hexadecimal Numbering System (Base-16) 96
 - 3.6.1 Converting from Hexadecimal to Decimal 97
 - 3.6.2 Converting from Decimal to Hexadecimal 97
 - 3.6.3 Converting between Hexadecimal and Octal Systems 98
 - 3.6.4 Converting between Hexadecimal and Binary Systems 98
- 3.7 Numbers with a Fractional Component 99
 - 3.7.1 Converting Binary, Octal and Hexadecimal Fractions to Decimal 99
 - 3.7.2 Converting a Decimal Fraction to Binary, Octal and Hexadecimal 100
 - 3.7.3 Converting between Binary, Octal and Hexadecimal Fractions 101
- 3.8 ASCII Codes 103
- 3.9 Logic Gates 103
 - 3.9.1 The AND, OR and XOR Gates 104
 - 3.9.2 The NOT Gate 105
 - 3.9.3 The NAND, NOR and XNOR Gates 105
- 3.10 Programming Methodologies 106
 - 3.10.1 Top-Down Programming 107
 - 3.10.2 Bottom-Up Programming 107
 - 3.10.3 Object-Oriented Programming 108
- 3.11 Structured Programming 108
- 3.12 Algorithms 109
 - 3.12.1 Sequencing 109
 - 3.12.2 Decision Making 110
 - 3.12.3 Repetition 110
- 3.13 Flowcharts 111
 - 3.13.1 Sequencing 111
 - 3.13.2 Decision Making 112
 - 3.13.3 Repetition 112
- 3.14 Classification of Programming Languages 114
 - 3.14.1 Assembly Language (2GL) 114
 - 3.14.2 High-Level Languages (3GL) 115
 - 3.14.3 4GL Languages 116
- 3.15 Compilers, Linkers and Loaders 116
- 3.16 Compiled vs Interpreted Languages 117

CHAPTER 4 A Taste of C **121**

- 4.1 The C Language 121
- 4.2 The Life Cycle of a C Program 122
 - 4.2.1 Header Files and Functions 122
 - 4.2.2 Editing the Program 124
 - 4.2.3 The Three-Phase Compilation Process 124
 - 4.2.4 Executing the Program 125
- 4.3 Know Your C Compiling Software 125
 - 4.3.1 GCC (Linux) 126
 - 4.3.2 Microsoft Visual Studio (Windows) 126
- 4.4 **first_prog.c**: Understanding Our First C Program 127
 - 4.4.1 Program Comments 128
 - 4.4.2 Preprocessor Section 128
 - 4.4.3 Variables and Computation 128
 - 4.4.4 Encounter with a Function: **printf** 129
 - 4.4.5 The **return** Statement 129
- 4.5 Editing, Compiling and Executing **first_prog.c** 130
 - 4.5.1 Using **gcc** (Linux) 130
 - 4.5.2 Using Visual Studio (Windows) 131
- 4.6 Handling Errors and Warnings 132
- 4.7 **second_prog.c**: An Interactive and Decision-Making Program 133
- 4.8 Two Functions: **printf** and **scanf** 135
 - 4.8.1 **printf**: Printing to the Terminal 135
 - 4.8.2 **scanf**: Input from the Keyboard 136
- 4.9 **third_prog.c**: An Interactive Program Featuring Repetition 137
- 4.10 Functions 138
- 4.11 **fourth_prog.c**: A Program Containing a User-Defined Function 139
- 4.12 Features of C 141
- 4.13 Some Programming Tips 142
- 4.14 C89 and C99: The C Standards 143

CHAPTER 5 Data—Variables and Constants **148**

- 5.1 Program Data 148
 - 5.1.1 Variables and Constants 148
 - 5.1.2 Data Types 149
 - 5.1.3 Data Sources 149
- 5.2 Variables 149
 - 5.2.1 Naming Variables 149
 - 5.2.2 Declaring Variables 150
 - 5.2.3 Assigning Variables 151
- 5.3 **intro2variables.c**: Declaring, Assigning and Printing Variables 151
- 5.4 Data Classification 152
 - 5.4.1 The Fundamental Types 153
 - 5.4.2 Type Sizes: The **sizeof** Operator 153

- 5.5 **sizeof.c**: Determining the Size of the Fundamental Data Types 154
- 5.6 The Integer Types 154
 - 5.6.1 The ANSI Stipulation for Integers 156
 - 5.6.2 Signed and Unsigned Integers 156
 - 5.6.3 The Specific Integer Types 157
 - 5.6.4 Octal and Hexadecimal Integers 158
- 5.7 **all_about_int.c**: Understanding Integers 158
- 5.8 The Floating Point Types 160
- 5.9 The Specific Floating Point Types 161
- 5.10 **all_about_real.c**: Understanding Floating Point Numbers 162
- 5.11 **char**: The Character Type 163
 - 5.11.1 **char** as Integer and Character 164
 - 5.11.2 Computation with **char** 165
 - 5.11.3 **char** as Escape Sequence 165
- 5.12 **all_about_char.c**: Understanding the **char** Data Type 166
- 5.13 Data Types of Constants 168
 - 5.13.1 Constants of the Fundamental Types 168
 - 5.13.2 Variables as Constants: The **const** Qualifier 169
 - 5.13.3 Symbolic Constants: Constants with Names 169
- 5.14 **sizeof_constants.c**: Program to Evaluate Size of Constants 170
- 5.15 Arrays and Strings: An Introduction 171
- 5.16 **intro2arrays.c**: Getting Familiar with Arrays 172

CHAPTER 6 Operators and Expressions

178

- 6.1 Expressions 178
- 6.2 Operators 179
- 6.3 **expressions.c**: Evaluating Expressions 180
- 6.4 Arithmetic Operators 181
 - 6.4.1 The +, -, * and /: The Basic Four Operators 182
 - 6.4.2 The %: The Modulus Operator 182
- 6.5 **computation.c**: Making Simple Calculations 182
- 6.6 Automatic or Implicit Type Conversion 184
- 6.7 **implicit_conversion.c**: Program to Demonstrate Implicit Conversion 185
- 6.8 Explicit Type Conversion—The Type Cast 186
- 6.9 **casts.c**: Program to Demonstrate Attributes of a Cast 187
- 6.10 Order of Evaluation 188
 - 6.10.1 Operator Precedence 188
 - 6.10.2 Using Parentheses to Change Default Order 189
 - 6.10.3 Operator Associativity 189
- 6.11 **order_of_evaluation.c**: Precedence and Associativity 190
- 6.12 Assignment Operators 192
 - 6.12.1 The =: The Simple Assignment Operator 192
 - 6.12.2 The Other Assignment Operators 192

- 6.13 ++ and --: Increment and Decrement Operators 193
 - 6.13.1 Side Effect of ++ and -- 193
 - 6.13.2 When Things Can Go Wrong 194
- 6.14 **computation2.c**: Using the Assignment and ++/-- Operators 194
- 6.15 Relational Operators and Expressions 196
 - 6.15.1 Using the Operators 196
 - 6.15.2 Precedence and Associativity 197
- 6.16 The Logical Operators 198
 - 6.16.1 The && & || Operators 198
 - 6.16.2 The ! Operator 199
- 6.17 The Conditional Operator 199
- 6.18 **binary_outcomes.c**: Relational, Logical and Conditional Operators at Work 200
- 6.19 Other Operators 200
 - 6.19.1 The Comma (,) Operator 200
 - 6.19.2 The **sizeof** Operator 202

CHAPTER 7 Control Structures—Decision Making

207

- 7.1 Decision-Making Concepts 207
- 7.2 Decision Making in C 208
 - 7.2.1 The Control Expression 208
 - 7.2.2 Compound Statement or Block 209
- 7.3 The **if** Statement 209
- 7.4 **average_integers.c**: Average Calculating Program 210
- 7.5 **if-else**: Two-Way Branching 212
- 7.6 **leap_year_check.c**: Program to Check Leap Year 212
- 7.7 Multi-Way Branching with **if-else-if ...** 214
- 7.8 **irctc_refund.c**: Computes Train Ticket Cancellation Charges 216
- 7.9 **atm_operation.c**: Checks PIN Before Delivery of Cash 216
- 7.10 Multi-Way Branching with Nested **if (if-if-else)** 218
- 7.11 **right_angle_check.c**: Program to Check Pythagoras' Theorem 219
- 7.12 Pairing Issues with **if-if-else** Nested Constructs 219
- 7.13 **leap_year_check2.c**: Program Using the **if-if-else** Structure 221
- 7.14 The Conditional Expression (?:) 223
- 7.15 The **switch** Statement 224
- 7.16 **calculator.c**: A Basic Calculator Program Using **switch** 226
- 7.17 **mobile_tariffs.c**: Program to Compute Charges for 4G Services 227
- 7.18 **date_validation.c**: A Program to Validate a Date 229
- 7.19 The “Infinitely Abusable” **goto** Statement 230

CHAPTER 8 Control Structures—Loops

236

- 8.1 Looping Basics 236
- 8.2 The **while** Loop 238
 - 8.2.1 **while_intro.c**: An Introductory Program 238

- 8.2.2 The Control Expression 238
- 8.2.3 Updating the Key Variable in the Control Expression 239
- 8.3 Three Programs Using **while** 240
 - 8.3.1 **factorial.c**: Determining the Factorial of a Number 240
 - 8.3.2 **extract_digits.c**: Program to Reverse Digits of an Integer 241
 - 8.3.3 **fibonacci_ite.c**: Printing and Summing the Fibonacci Numbers 242
- 8.4 Loading the Control Expression 243
 - 8.4.1 Merging Entire Loop Body with the Control Expression 243
 - 8.4.2 When You Actually Need a Null Body 244
- 8.5 Nested **while** Loops 244
 - 8.5.1 **nested_while.c**: Printing a Multiplication Table 245
 - 8.5.2 **half_pyramid.c**: Printing a Half-Pyramid with Digits 246
 - 8.5.3 **power.c**: Computing Power of a Number 247
- 8.6 Using **while** with **break** and **continue** 248
- 8.7 **prime_number_check.c**: More of **break** and **continue** 249
- 8.8 The **do-while** Loop 251
- 8.9 **decimal2binary.c**: Collecting Remainders from Repeated Division 252
- 8.10 The **for** Loop 253
 - 8.10.1 **ascii_letters.c**: A Portable ASCII Code Generator 255
 - 8.10.2 **print_terms.c**: Completing a Series of Mathematical Expressions 256
- 8.11 **for**: The Three Expressions (*exp1*, *exp2*, *exp3*) 257
 - 8.11.1 Moving All Initialization to *exp1* 257
 - 8.11.2 Moving Expression Statements in Body to *exp3* 257
 - 8.11.3 Dropping One or More Expressions 258
 - 8.11.4 The Infinite Loop 258
- 8.12 **decimal2binary2.c**: Converting a Decimal Number to Binary 259
- 8.13 Nested **for** Loops 260
- 8.14 Using **for** with **break** and **continue** 261
- 8.15 **all_prime_numbers.c**: Using **break** and **continue** 261

CHAPTER 9 Terminal Input/Output Functions

266

- 9.1 I/O Function Basics 266
- 9.2 Character Input with **getchar** 267
- 9.3 Character Output with **putchar** 269
- 9.4 **retrieve_from_buffer.c**: A Buffer-Related Issue 269
- 9.5 The Standard Files 270
- 9.6 **unix2dos.c**: Program to Convert a UNIX File to MSDOS Format 271
- 9.7 Other Character-I/O Functions 272
 - 9.7.1 The **fgetc** and **fputc** Functions 272
 - 9.7.2 The **getc** and **putc** Macros 272
 - 9.7.3 The **ungetc** Function 273
- 9.8 Formatted Input: The **scanf** Function 274

- 9.9 **scanf**: The Matching Rules 277
 - 9.9.1 Mismatches and Return Value 277
 - 9.9.2 **scanf_retval.c**: Program to Extract Numbers from a String 278
- 9.10 **scanf**: The Major Format Specifiers 279
 - 9.10.1 Handling Numeric Data (%d and %f) 280
 - 9.10.2 Handling Character Data (%c) 280
 - 9.10.3 Handling String Data (%s) 281
- 9.10.4 **scanf_char_string.c**: Characters and Strings as Input 281
- 9.11 **scanf**: Other Features 281
 - 9.11.1 The Scan Set Specifier 281
 - 9.11.2 The * Flag 284
 - 9.11.3 The %p, %n and %% Specifiers 284
- 9.12 Formatted Output: The **printf** Function 285
 - 9.12.1 **printf** and **scanf** Compared 286
 - 9.12.2 Printing Plain Text 286
 - 9.12.3 **printf_special.c**: A Second Look at the Data Types 287
- 9.13 **printf**: Using Field Width and the * Flag 288
- 9.14 **printf**: Using Precision 289
 - 9.14.1 Precision with Floating Point Numbers 289
 - 9.14.2 Precision with Integers 290
 - 9.14.3 Precision with Character Strings 290
- 9.15 **printf**: Using Flags 291
- 9.16 **printf_flags.c**: Using the Flags 292

CHAPTER 10 Arrays

298

-
- 10.1 Array Basics 298
 - 10.2 Declaring and Initializing an Array 299
 - 10.2.1 Initializing During Declaration 300
 - 10.2.2 Initializing After Declaration 300
 - 10.3 **array_init.c**: Initializing and Printing Arrays 301
 - 10.4 **scanf_with_array.c**: Populating an Array with **scanf** 302
 - 10.5 Basic Operations on Arrays 302
 - 10.5.1 **insert_element.c**: Inserting an Element in an Array 303
 - 10.5.2 **delete_element.c**: Deleting an Element from an Array 304
 - 10.6 Reversing an Array 305
 - 10.6.1 Reversing with Two Arrays 305
 - 10.6.2 Reversing with a Single Array 306
 - 10.7 Two Programs Revisited 306
 - 10.7.1 **extract_digits2.c**: Saving Digits of an Integer 306
 - 10.7.2 **decimal2anybase.c**: Converting from Decimal to Any Base 307
 - 10.8 Sorting an Array (Selection) 309
 - 10.9 **array_search.c**: Program to Sequentially Search an Array 311

- 10.10 Binary Search 311
 - 10.10.1 The Binary Search Algorithm 313
 - 10.10.2 **binary_search.c**: Implementation of the Algorithm 313
- 10.11 **count_chars.c**: Frequency Count of Data Items 315
- 10.12 Two-Dimensional (2D) Arrays 316
 - 10.12.1 Full Initialization During Declaration 317
 - 10.12.2 Partial Initialization During Declaration 318
 - 10.12.3 Assignment After Declaration 318
 - 10.12.4 Assignment and Printing Using a Nested Loop 319
- 10.13 **count_numbers.c**: Using a 2D Array as a Counter 320
- 10.14 Multi-Dimensional Arrays 321
- 10.15 Using Arrays as Matrices 322
 - 10.15.1 **matrix_transpose.c**: Transposing a Matrix 322
 - 10.15.2 **matrix_add_subtract.c**: Adding and Subtracting Two Matrices 323
 - 10.15.3 **matrix_multiply.c**: Multiplying Two Matrices 324
- 10.16 Variable Length Arrays (C99) 326

CHAPTER 11 Functions

331

- 11.1 Function Basics 331
- 11.2 **first_func.c**: No Arguments, No Return Value 332
- 11.3 The Anatomy of a Function 334
 - 11.3.1 Declaration, Prototype or Signature 334
 - 11.3.2 Definition or Implementation 335
 - 11.3.3 Invocation or Call 335
- 11.4 **c2f.c**: One Argument and Return Value 336
- 11.5 Arguments, Parameters and Local Variables 337
 - 11.5.1 Parameter Passing: Arguments and Parameters 338
 - 11.5.2 Passing by Value vs Passing by Reference 338
 - 11.5.3 Local Variables 339
 - 11.5.4 **swap_failure.c**: The “Problem” with Local Variables 340
- 11.6 The Return Value and Side Effect 341
- 11.7 **prime_number_check2.c**: Revised Program to Check Prime Numbers 342
- 11.8 Using Arrays in Functions 343
 - 11.8.1 **input2array.c**: Passing an Array as an Argument 345
 - 11.8.2 The **static** keyword: Keeping an Array Alive 346
- 11.9 **merge_arrays.c**: Merging Two Sorted Arrays 346
- 11.10 Passing a Two-Dimensional Array as Argument 348
- 11.11 Calling a Function from Another Function 350
 - 11.11.1 **time_diff.c**: Program to Compute Difference Between Two Times 350
 - 11.11.2 **power_func.c**: Computing the Sum of a Power Series 351
- 11.12 **sort_bubble.c**: Ordering an Array Using Bubble Sort 353
- 11.13 Recursive Functions 355
 - 11.13.1 **factorial_rec.c**: Using a Recursive Function to Compute Factorial 356
 - 11.13.2 Recursion vs Iteration 357

- 11.14 Thinking in Recursive Terms 359
 - 11.14.1 Adding Array Elements Recursively 359
 - 11.14.2 Computing the Power of a Number Recursively 360
 - 11.14.3 Using Recursion To Compute Fibonacci Numbers 360
- 11.15 The **main** Function 361
- 11.16 Variable Scope and Lifetime 362
 - 11.16.1 Local and Global Variables 362
 - 11.16.2 Variables in a Block 362
 - 11.16.3 Variable Hiding 363
- 11.17 The Storage Classes 364
 - 11.17.1 Automatic Variables (**auto**) 364
 - 11.17.2 Static Variables (**static**) 364
 - 11.17.3 External Variables (**extern**) 365
 - 11.17.4 **extern.c**: Using **extern** to Control Variable Visibility 366
 - 11.17.5 Register Variables (**register**) 368

CHAPTER 12 Pointers

373

- 12.1 Memory Access and the Pointer 373
- 12.2 Pointer Basics 374
- 12.3 **intro2pointers.c**: A Simple Demonstration of Pointers 375
- 12.4 Declaring, Initializing and Dereferencing a Pointer 375
- 12.5 **pointers.c**: Using Two Pointers 377
- 12.6 Important Attributes of Pointers 379
- 12.7 Pointers and Functions 382
 - 12.7.1 **swap_success.c**: Making the **swap** Function Work 383
 - 12.7.2 Using Pointers to Return Multiple Values 384
 - 12.7.3 **sphere_calc.c**: Function “Returning” Multiple Values 385
 - 12.7.4 Returning a Pointer 386
- 12.8 Pointers and Arrays 387
 - 12.8.1 Using a Pointer for Browsing an Array 387
 - 12.8.2 Dereferencing the Array Pointer 388
 - 12.8.3 Treating a Pointer as an Array 389
 - 12.8.4 Array vs Pointer Which Points to an Array 389
 - 12.8.5 **pointer_array.c**: Treating an Array as a Pointer 390
- 12.9 Operations on Pointers 391
 - 12.9.1 Assignment 391
 - 12.9.2 Pointer Arithmetic Using **+** and **-** 391
 - 12.9.3 Pointer Arithmetic Using **++** and **--** 392
 - 12.9.4 Comparing Two Pointers 393
- 12.10 **max_min.c**: Using **scanf** and **printf** with Pointer Notation 393
- 12.11 **NULL** and the Null Pointer 393
- 12.12 Pointers, Arrays and Functions Revisited 395
 - 12.12.1 Pointers in Lieu of Array as Function Argument 395

xxii Contents

- 12.12.2 Using **const** to Protect an Array from a Function 397
- 12.12.3 Returning a Pointer to an Array 397
- 12.13 Array of Pointers 398
- 12.14 **sort_selection2.c**: Sorted View of Array Using Array of Pointers 398
- 12.15 Pointer to a Pointer 399
- 12.16 Pointers and Two-Dimensional Arrays 402
- 12.17 The Generic or Void Pointer 404
- 12.18 Using the **const** Qualifier with Pointers 406

CHAPTER 13 Strings

412

- 13.1 String Basics 412
- 13.2 Declaring and Initializing a String 413
 - 13.2.1 Using an Array to Declare a String 413
 - 13.2.2 When an Array is Declared But Not Initialized 413
 - 13.2.3 Using a Pointer to Declare a String 414
 - 13.2.4 When an Array of Characters Is Not a String 415
- 13.3 **intro2strings.c**: Declaring and Initializing Strings 415
- 13.4 Handling Lines as Strings 417
 - 13.4.1 Using **gets/puts**: The Unsafe Way 417
 - 13.4.2 **gets_puts.c**: A Program Using **gets** and **puts** 417
 - 13.4.3 Using **fgets/fputs**: The Safe Way 418
 - 13.4.4 **fgets_fputs.c**: A Program Using **fgets** and **fputs** 419
- 13.5 The **sscanf** and **sprintf** Functions 420
 - 13.5.1 **sscanf**: Formatted Input from a String 420
 - 13.5.2 **sprintf**: Formatted Output to a String 421
 - 13.5.3 **validate_pan.c**: Using **sscanf** and **sprintf** to Validate Data 421
- 13.6 Using Pointers for String Manipulation 422
- 13.7 Common String-Handling Programs 423
 - 13.7.1 **string_palindrome.c**: Program to Check a Palindrome 423
 - 13.7.2 **count_words.c**: Counting Number of Words in a String 424
 - 13.7.3 **sort_characters.c**: Sorting Characters in a String 425
- 13.8 Developing String-Handling Functions 426
 - 13.8.1 **my_strlen**: Function to Evaluate Length of a String 426
 - 13.8.2 **reverse_string**: Function to Reverse a String 427
 - 13.8.3 **my_strcpy**: Function to Copy a String 427
 - 13.8.4 **my_strcat**: Function to Concatenate Two Strings 427
 - 13.8.5 **string_manipulation.c**: Invoking All Four Functions from main 428
 - 13.8.6 **substr.c**: Program Using a Function to Extract a Substring 429
- 13.9 Standard String-Handling Functions 430
 - 13.9.1 The **strlen** Function 430
 - 13.9.2 The **strcpy** Function 430
 - 13.9.3 The **strcat** Function 431
 - 13.9.4 The **strcmp** Function 431

- 13.9.5 The **strchr** and **strrchr** Functions 432
- 13.9.6 The **strstr** Function 432
- 13.10 The Character-Oriented Functions 433
- 13.11 **password_check.c**: Using the Character-Handling Functions 434
- 13.12 Two-Dimensional Array of Strings 435
- 13.13 Array of Pointers to Strings 435
- 13.14 **sort_strings.c**: Sorting a 2D Array of Strings 437
- 13.15 **string_swap.c**: Swapping Two Strings 438
- 13.16 The **main** Function Revisited 439

CHAPTER 14 User-Defined Data Types

445

- 14.1 Structure Basics 445
- 14.2 Declaring and Defining a Structure 446
 - 14.2.1 Accessing Members of a Structure 447
 - 14.2.2 Combining Declaration, Definition and Initialization 448
 - 14.2.3 Declaring without Structure Tag 448
- 14.3 **intro2structures.c**: An Introductory Program 449
- 14.4 Important Attributes of Structures 450
 - 14.4.1 **structure_attributes.c**: Copying and Comparing Structures 451
 - 14.4.2 Abbreviating a Data Type: The **typedef** Feature 452
 - 14.4.3 **structure_typedef.c**: Simplifying Use of Structures 453
- 14.5 Nested Structures 454
 - 14.5.1 Initializing a Nested Structure 455
 - 14.5.2 Accessing Members 456
 - 14.5.3 **structure_nested.c**: Program Using a Nested Structure 456
- 14.6 Arrays as Structure Members 457
- 14.7 Arrays of Structures 458
 - 14.7.1 **array_of_structures.c**: Program for Displaying Batting Averages 459
 - 14.7.2 **structure_sort.c**: Sorting an Array of Structures 460
- 14.8 Structures in Functions 462
 - 14.8.1 **structure_in_func.c**: An Introductory Program 463
 - 14.8.2 **time_difference.c**: Using a Function that Returns a Structure 464
 - 14.8.3 **swap_success2.c**: Swapping Variables Revisited 465
- 14.9 Pointers to Structures 466
 - 14.9.1 **pointer_to_structure.c**: Accessing Structure Members 467
 - 14.9.2 **update_pay.c**: Using a Pointer as a Function Argument 468
 - 14.9.3 **time_addition.c**: Using Pointers as Function Arguments 469
- 14.10 **student_management.c**: A Project 470
- 14.11 Unions 473
 - 14.11.1 Unique Attributes of Unions 473
 - 14.11.2 **intro2unions**: An Introductory Program 474
- 14.12 Bit Fields 475
- 14.13 The Enumerated Type 476

CHAPTER 15 File Handling

484

- 15.1 A Programmer's View of the File 484
- 15.2 File-Handling Basics 485
- 15.3 Opening and Closing Files 486
 - 15.3.1 **fopen**: Opening a File 486
 - 15.3.2 File Opening Modes 486
 - 15.3.3 The Filename 487
 - 15.3.4 Error Handling 488
 - 15.3.5 **fclose**: Closing a File 489
 - 15.3.6 **fopen_fclose.c**: An Introductory Program 489
- 15.4 The File Pointer and File Buffer 490
- 15.5 The File Read/Write Functions 491
 - 15.5.1 **mixing_functions.c**: Manipulating the File Offset Pointer 491
 - 15.5.2 The **fgetc** and **fputc** Functions Revisited 492
 - 15.5.3 **file_copy.c**: A File Copying Program 493
 - 15.5.4 **file_append.c**: A File Appending Program 494
- 15.6 The **fgets** and **fputs** Functions Revisited 495
 - 15.6.1 **fgets_fputs2.c**: Using **fgets** and **fputs** with a Disk File 496
 - 15.6.2 **save_student_data.c**: Writing Data Stored in Array to a File 497
- 15.7 **fscanf** and **fprintf**: The Formatted Functions 498
- 15.8 **fscanf_fprintf.c**: Writing and Reading Lines Containing Fields 498
- 15.9 Filenames from Command-Line Arguments 500
 - 15.9.1 **file_copy2.c**: File Copying Using Command-Line Arguments 500
 - 15.9.2 **validate_records.c**: Detecting Lines Having Wrong Number of Fields 501
- 15.10 **perror** and **errno**: Handling Function Errors 503
- 15.11 **feof**, **ferror** and **clearerr**: EOF and Error Handling 504
- 15.12 Text and Binary Files 505
- 15.13 **fread** and **fwrite**: Reading and Writing Binary Files 506
 - 15.13.1 The **fread** Function 506
 - 15.13.2 The **fwrite** Function 507
 - 15.13.3 **fread_fwrite.c**: Using the Primitive and Derived Data Types 507
 - 15.13.4 **save_structure.c**: Saving and Retrieving a Structure 508
- 15.14 Manipulating the File Position Indicator 510
 - 15.14.1 **fseek**: Positioning the Offset Pointer 510
 - 15.14.2 The **ftell** and **rewind** Functions 510
 - 15.14.3 **reverse_read.c**: Reading a File in Reverse 511
- 15.15 **update_structure.c**: Updating a Structure Stored on Disk 512
- 15.16 The Other File-Handling Functions 514
 - 15.16.1 The **remove** Function 514
 - 15.16.2 The **rename** Function 514
 - 15.16.3 The **tmpfile** Function 515

CHAPTER 16 Dynamic Memory Allocation and Linked Lists 520

- 16.1 Memory Allocation Basics 520
- 16.2 The Functions for Dynamic Memory Allocation 521
 - 16.2.1 The Generic Pointer 522
 - 16.2.2 Error Handling 522
- 16.3 **malloc**: Specifying Memory Requirement in Bytes 523
 - 16.3.1 **malloc.c**: An Introductory Program 524
 - 16.3.2 Error-Checking in **malloc** 524
 - 16.3.3 Using **malloc** to Store an Array 525
 - 16.3.4 **malloc_array.c**: An Array-Handling Program Using **malloc** 525
- 16.4 **free**: Freeing Memory Allocated by **malloc** 526
- 16.5 Memory Mismanagement 527
 - 16.5.1 The Dangling Pointer 527
 - 16.5.2 Memory Leaks 528
- 16.6 **malloc_2Darray.c**: Simulating a 2D Array 529
- 16.7 **malloc_strings.c**: Storing Multiple Strings 531
- 16.8 **calloc**: Allocating Memory for Arrays and Structures 532
- 16.9 **realloc**: Changing Size of Allocated Memory Block 534
- 16.10 The Linked List 536
 - 16.10.1 Creating a Linked List with Variables 538
 - 16.10.2 **create_list.c**: Creating a Linked List Using **malloc** 539
 - 16.10.3 Operations on Linked Lists 540
- 16.11 Adding a Node 541
 - 16.11.1 Adding a Node at Beginning 541
 - 16.11.2 Adding a Node at End 542
- 16.12 Deleting a Node at Beginning and End 543
- 16.13 **head_tail_operations.c**: Adding and Deleting a Single Node 543
- 16.14 **list_manipulation.c**: A List Handling Program 545
 - 16.14.1 The **add_node** Function 547
 - 16.14.2 The **count_nodes** Function 548
 - 16.14.3 The **find_node** Function 548
 - 16.14.4 The **insert_after** Function 549
 - 16.14.5 The **delete_node** Function 549
- 16.15 Types of Linked Lists 550
- 16.16 Abstract Data Types (ADTs) 550
 - 16.16.1 The Stack 551
 - 16.16.2 The Queue 551
 - 16.16.3 The Tree 552

CHAPTER 17 The Preprocessor and Other Features 558

- 17.1 The Preprocessor 558
- 17.2 **#define**: Macros without Arguments 560
 - 17.2.1 Using Numbers and Expressions 561

17.2.2	Why Not Use a Variable?	562
17.2.3	Abbreviating Text	562
17.3	#define : Macros with Arguments	563
17.3.1	When Parentheses Are Required	564
17.3.2	Useful Macros	564
17.3.3	swap_with_macro.c : Swapping Two Numbers Using a Macro	565
17.3.4	Functions vs Macros	566
17.4	Macros and Strings	567
17.4.1	The # Operator	567
17.4.2	The ## Operator	568
17.4.3	tokens.c : Using the # and ## Operators	568
17.5	The #undef Directive	569
17.6	The #include Directive	570
17.7	Conditional Compilation	571
17.7.1	The #ifdef and #ifndef Directives	572
17.7.2	The #if and #elif Directives	573
17.8	Using #ifdef for Debugging Programs	574
17.9	The Bitwise Operators	575
17.9.1	The & Operator: Bitwise AND	576
17.9.2	Using a Mask with &	577
17.9.3	The Operator: Bitwise OR	578
17.9.4	The ^ Operator: Bitwise Exclusive OR (XOR)	578
17.9.5	The ~ Operator: One's Complement (NOT)	579
17.9.6	The << Operator: Bitwise Left-Shift	579
17.9.7	The >> Operator: Bitwise Right-Shift	580
17.10	bitwise_operations.c : Using the Bitwise Operators	580
17.11	Pointers to Functions	582
17.11.1	Function Pointer for Functions Using Arguments	583
17.11.2	func_pointer.c : Using Function Pointers	584
17.11.3	Callback Functions	585
17.12	Functions with Variable Arguments	587
17.13	Multi-Source Program Files	589
17.13.1	A Multi-Source Application	590
17.13.2	Compiling and Linking the Application	592
Appendix A	Selective C Reference	599
Appendix B	The ASCII Character Set	606
Appendix C	Glossary	610
Appendix D	Answers to Objective Questions	628
Appendix E	Bibliography	635
Index		637