

Contents

1 Introduction 1

- 1.1 A brief history of computing 2
 - 1.2 What is computer science? 4
 - 1.3 A brief tour of computer hardware 6
 - The CPU 7, Memory 7, Secondary storage 8, I/O devices 8
 - 1.4 Algorithms 9
 - 1.5 Programming languages and compilation 10
 - 1.6 Programming errors and debugging 12
 - 1.7 Software maintenance 13
 - 1.8 The importance of software engineering 14
 - 1.9 Some thoughts on the C programming language 15
- Summary 16
REVIEW QUESTIONS 17

PART ONE

The Basics of C Programming 19

2 Learning by Example 21

- 2.1 The "Hello world" program 22
 - Comments 22, Library inclusions 23, The main program 24
 - 2.2 A program to add two numbers 26
 - The input phase 28, The computation phase 30, The output phase 30
 - 2.3 Perspectives on the programming process 32
 - 2.4 Data types 33
 - Floating-point data 34, String data 34
 - 2.5 Expressions 37
 - Constants 38, Variables 39, Assignment statements 41, Operators and operands 43, Combining integers and floating-point numbers 44, Integer division and the remainder operator 45, Precedence 45, Applying precedence rules 48, Type conversion 48
- Summary 51
REVIEW QUESTIONS 52
PROGRAMMING EXERCISES 55

3 Problem Solving 59

- 3.1 Programming idioms and paradigms 60**
Shorthand assignment idioms 61, Increment and decrement operators 62
- 3.2 Solving problems on a larger scale 63**
- 3.3 Control statements 65**
The repeat-N-times idiom 65, Iteration and loops 67, Index variables 67, The importance of initialization 69, The read-until-sentinel idiom 69, Building a more practical application 71, Conditional execution and the `if` statement 74
- 3.4 An exercise in debugging 76**
- 3.5 Formatted output 80**
Format codes for `printf` 82, Controlling spacing, alignment, and precision 83
- 3.6 Crafting a program 86**
Programming style 86, Designing for change 88, The `#define` mechanism 89
Summary 89
REVIEW QUESTIONS 91
PROGRAMMING EXERCISES 93

4 Statement Forms 99

- 4.1 Simple statements 100**
Embedded assignments 102, Multiple assignments 102, Blocks 103
- 4.2 Control statements 104**
- 4.3 Boolean data 105**
Relational operators 106, Logical operators 107, Short-circuit evaluation 109, Flags 110, Avoiding redundancy in Boolean expressions 110, An example of Boolean calculation 111
- 4.4 The `if` statement 112**
Single-line `if` statements 114, Multiline `if` statements 114, The `if-else` statement 114, Cascading `if` statements 115, The `?:` operator 115
- 4.5 The `switch` statement 117**
- 4.6 The `while` statement 119**
Using the `while` loop 120, Infinite loops 122
Solving the loop-and-a-half problem 123

- 4.7 The for statement 125**
Nested for loops 126, The relationship between for and while 129, Using for with floating-point data 129
Summary 131
REVIEW QUESTIONS 132
PROGRAMMING EXERCISES 133

5 Functions 137

- 5.1 Using library functions 138**
- 5.2 Function declarations 140**
- 5.3 Writing your own functions 141**
The return statement 142, Putting functions together with main programs 143, Functions involving internal control structures 145, Functions that return nonnumeric values 146, Predicate functions 148, A predicate function to test for string equality 149
- 5.4 Mechanics of the function-calling process 150**
Parameter passing 151, Calling functions from within other functions 156
- 5.5 Procedures 163**
- 5.6 Stepwise refinement 165**
Starting at the top 166, Implementing PrintCalendar 167, Implementing PrintCalendarMonth 167, Completing the final pieces 171
Summary 176
REVIEW QUESTIONS 177
PROGRAMMING EXERCISES 178

6 Algorithms 185

- 6.1 Testing for primality 186**
A simple version of IsPrime 187, Verifying that a strategy represents an algorithm 187, Demonstrating the correctness of the IsPrime algorithm 188, Improving algorithmic efficiency 189, Choosing between alternative implementations 192
- 6.2 Computing the greatest common divisor 193**
Brute-force algorithms 194, Euclid's algorithm 195, Defending the correctness of Euclid's algorithm 195, Comparing the efficiency of the GCD algorithms 197
- 6.3 Numerical algorithms 197**
Successive approximation 198, Reporting errors 200
- 6.4 Series expansion 201**
Zeno's paradox 201, Using a series expansion for the square root function 203, The Taylor series expansion for approximating a square root 204, Implementing the Taylor series approximation 205, Staying within the radius of convergence 207

- 6.5 Specifying the size of numeric types 210**
 - Integer types 210, Unsigned types 211, Floating-point types 212
 - Summary 212
 - REVIEW QUESTIONS 213
 - PROGRAMMING EXERCISES 214

PART TWO

Libraries and Modular Development 219

- 7 Libraries and Interfaces: A Simple Graphics Library 221**
 - 7.1 The concept of an interface 222**
 - Interfaces and header files 224
 - 7.2 An introduction to the graphics library 225**
 - The underlying model for graphics.h 225, The functions in the graphics.h interface 227, Initializing the package 231, Drawing straight lines 231, Drawing circles and arcs 233, Obtaining information about the graphics window 235
 - 7.3 Building your own tools 235**
 - Defining DrawBox 236, Defining DrawCenteredCircle 237, Switching between absolute and relative coordinates 239, The advantages of defining procedures 240
 - 7.4 Solving a larger problem 240**
 - Using stepwise refinement 241, Implementing the DrawHouse procedure 242, Looking for common patterns 243, Finishing the decomposition 245
 - Summary 250
 - REVIEW QUESTIONS 251
 - PROGRAMMING EXERCISES 252
- 8 Designing Interfaces: A Random Number Library 259**
 - 8.1 Interface design 260**
 - The importance of a unifying theme 261, Simplicity and the principle of information hiding 261, Meeting the needs of your clients 263, The advantages of general tools 263, The value of stability 264
 - 8.2 Generating random numbers by computer 265**
 - Deterministic versus nondeterministic behavior 265, Random versus pseudo-random numbers 265, Generating pseudo-random numbers in ANSI C 266, Changing the range of random numbers 267, Generalizing the problem 272
 - 8.3 Saving tools in libraries 274**
 - The contents of an interface 275, Writing the random.h interface 276, The random.c implementation 277, Constructing a client program 278, Initializing the random number generator 280

8.4	Evaluating the design of the <code>random.h</code> interface 283 Generating random real numbers 284, Simulating a probabilistic event 285, Including header files in an interface 286, Completing the implementation of the random-number package 286	
8.5	Using the random-number package 286 Summary 290 REVIEW QUESTIONS 291 PROGRAMMING EXERCISES 292	
9	Strings and Characters	301
9.1	The principle of enumeration 302 Representing enumeration types inside the machine 303, Representing enumeration types as integers 304, Defining new enumeration types 304, Operations on enumeration types 307, Scalar types 307	
9.2	Characters 308 The data type <code>char</code> 308, The ASCII code 308, Character constants 310, Important properties of the ASCII coding scheme 310, Special characters 311, Character arithmetic 312, The <code>ctype.h</code> interface 314, Control statements involving characters 315, Character input and output 316	
9.3	Strings as abstract data 316 Layered abstractions 317, The concept of an abstract type 319	
9.4	The <code>strlib.h</code> interface 319 Determining the length of a string 320, Selecting characters from a string 321, Concatenation 321, Converting characters to strings 322, Extracting parts of a string 323, Comparing one string with another 324, Searching within a string 325, Case conversion 327, Numeric conversion 329, Efficiency and the <code>strlib.h</code> library 330 Summary 331 REVIEW QUESTIONS 332 PROGRAMMING EXERCISES 334	
10	Modular Development	339
10.1	Pig Latin—a case study in modular development 342 Applying top-down design 342, Using pseudocode 343, Implementing <code>TranslateLine</code> 344, Taking spaces and punctuation into account 345, Refining the definition of a word 347, Designing the token scanner 348, Completing the <code>TranslateLine</code> implementation 349, Specifying the scanner module interface 352	
10.2	Maintaining internal state within a module 352 Global variables 355, The dangers of using global variables 356, Keeping variables private to a module 356, Initializing global variables 357, Private functions 359	
10.3	Implementing the scanner abstraction 359 Summary 366 REVIEW QUESTIONS 366 PROGRAMMING EXERCISES 367	

PART THREE

***Compound Data Types* 373**

11 Arrays

375

11.1 Introduction to arrays 376

Array declaration 377, Array selection 378, Example of a simple array 379, Changing the index range 381

11.2 Internal representation of data 382

Bits, bytes, and words 382, Memory addresses 383, The sizeof operator 385, Allocation of memory to variables 385, References to elements outside the array bound 386

11.3 Passing arrays as parameters 388

Generalizing the number of elements 390, The mechanics of array parameter transmission 392, Implementing PrintIntegerArray and GetIntegerArray 394, Implementing ReverseIntegerArray 396, Implementing SwapIntegerElements 397

11.4 Using arrays for tabulation 398

11.5 Static initialization of arrays 404

Automatic determination of array size 408, Determining the size of an initialized array 409, Initialized arrays and scalar types 409

11.6 Multidimensional arrays 410

Passing multidimensional arrays to functions 411, Initializing multidimensional arrays 412

Summary 413

REVIEW QUESTIONS 414

PROGRAMMING EXERCISES 416

12 Searching and Sorting

425

12.1 Searching 426

Searching in an integer array 426, A more sophisticated example of searching 429, Linear search 431, Binary search 433, Relative efficiency of the search algorithms 435

12.2 Sorting 437

Sorting an integer array 437, The selection sort algorithm 438, Evaluating the efficiency of selection sort 442, Measuring the running time of a program 442, Analyzing the selection sort algorithm 445

Summary 446

REVIEW QUESTIONS 446

PROGRAMMING EXERCISES 447

13 Pointers

453

13.1 Using addresses as data values 455

13.2 Pointer manipulation in C 457

Declaring pointer variables in C 457, The fundamental pointer operations 458, The special pointer NULL 461

13.3	Passing parameters by reference 461 Designing a <code>SwapInteger</code> function 465, Using call by reference to return multiple results 466, The danger of overusing call by reference 467	
13.4	Pointers and arrays 468 Pointer arithmetic 469, New capabilities of the <code>++</code> and <code>--</code> operators 473, Incrementing and decrementing pointers 475, The relationship between pointers and arrays 476	
13.5	Dynamic allocation 478 The type <code>void *</code> 478, Dynamic arrays 480, Detecting errors in <code>malloc</code> 481, Freeing memory 481 Summary 482 REVIEW QUESTIONS 483 PROGRAMMING EXERCISES 486	
14	Strings Revisited	491
14.1	Conceptual representations of the type <code>string</code> 492 Strings as arrays 491, Strings as pointers 495, Strings as an abstract type 496, String parameters 497, String variables 497, Differences between pointer and array variables 499, Deciding on a strategy for string representation 502	
14.2	The ANSI string library 502 The <code>strcpy</code> function 504, The <code>strncpy</code> function 507, The <code>strcat</code> and <code>strncat</code> functions 508, The <code>strlen</code> , <code>strcmp</code> , and <code>strncmp</code> functions 509, The <code>strchr</code> , <code>strrchr</code> , and <code>strstr</code> functions 510, An application of the ANSI string functions 510	
14.3	Implementing the <code>strlib</code> library 511 Implementing the pass-through functions 511, Implementing the <code>strlib</code> allocation functions 514 Summary 516 REVIEW QUESTIONS 517 PROGRAMMING EXERCISES 517	
15	Files	523
15.1	Text files 524	
15.2	Using files in C 525 Declaring a <code>FILE *</code> variable 526, Opening a file 526, Performing I/O operations 527, Closing files 528, Standard files 528	
15.3	Character I/O 529 Updating a file 531, Rereading characters in the input file 533	
15.4	Line-oriented I/O 536	

- 15.5 Formatted I/O 539**
The three forms of `printf` 539, The `scanf` functions 539,
Reading strings with `scanf` 541, An example of formatted I/O 543,
Limitations on the use of `scanf` 546
Summary 547
REVIEW QUESTIONS 548
PROGRAMMING EXERCISES 549

16 Records 557

- 16.1 The concept of the data record 558**
16.2 Using records in C 559
Defining a new structure type 560, Declaring structure variables 560,
Record selection 561, Initializing records 561, Simple records 562
16.3 Combining records and arrays 563
16.4 Pointers to records 566
Defining a pointer-to-record type 566, Allocating storage for
record data 568, Manipulating pointers to records 569
16.5 Building a database of records 570
Creating the employee database 570, Using the database 573
16.6 Designing a record-based application 574
The importance of using a database 575, Framing the
problem 575, Designing the internal representation 576,
Designing the external structure 580, Coding the program 581,
The value of a data-driven design 588
Summary 589
REVIEW QUESTIONS 592
PROGRAMMING EXERCISES 593

17 Looking Ahead 601

- 17.1 Recursion 602**
A simple illustration of recursion 603, The Factorial function 604,
The recursive leap of faith 609, The recursive paradigm 610,
Generating permutations 611, Thinking recursively 614
17.2 Abstract data types 614
The queue abstraction 615, Representing types in the queue
abstraction 616, The `queue.h` interface 618, Implementing
the queue abstraction 620, Alternative implementation of the
queue abstraction 622

17.3	Analysis of algorithms 628	
	Evaluating algorithmic efficiency 628, Big-O notation 629, Selection sort revisited 630, Divide-and-conquer strategies 631, Merging two arrays 632, The merge sort algorithm 633, The computational complexity of merge sort 635, Comparing quadratic and N log N performance 636	
	Summary 637	
	REVIEW QUESTIONS 638	
	PROGRAMMING EXERCISES 639	

Appendix A. Summary of C Syntax and Structure 647

A.1	An overview of the compilation process 648
A.2	The C preprocessor 648
A.3	The lexical structure of C programs 650
A.4	Expressions 652
A.5	Statements 653
A.6	Functions 657
A.7	Declarations 658
A.8	Data types 659
A.9	ANSI libraries 662

Appendix B. Library Sources 669

Index 696