Contents

Preface xiii
Acknowledgments xv
Introduction xix
   The $20 Million Bug xx
   Convention xxi
   Some Light Relief—Tuning File Systems xxii

1. C Through the Mists of Time 1
   The Prehistory of C 1
   Early Experiences with C 4
   The Standard I/O Library and C Preprocessor 6
   K&R C 9
   The Present Day: ANSI C 11
   It's Nice, but Is It Standard? 14
   Translation Limits 16
   The Structure of the ANSI C Standard 17
   Reading the ANSI C Standard for Fun, Pleasure, and Profit 22
   How Quiet is a "Quiet Change"? 25
   Some Light Relief—The Implementation-Defined Effects of Pragmas . . . 29

2. It's Not a Bug, It's a Language Feature 31
   Why Language Features Matter—The Way the Fortran Bug Really Happened! 31
   Sins of Commission 33
      Switches Let You Down with Fall Through 33
      Available Hardware Is a Crayon? 39
      Too Much Default Visibility 41
   Sins of Mission 42
      Overloading the Camel's Back 42
      "Some of the Operators Have the Wrong Precedence" 44
      The Early Bug gets() the Internet Worm 48
Sins of Omission  50
  Mail Won’t Go to Users with an “f” in Their Usernames  50
Space—The Final Frontier  53
  A Digression into C++ Comments  55
The Compiler Date Is Corrupted  55
  Lint Should Never Have Been Separated Out  59
Some Light Relief—Some Features Really Are Bugs!  60
   References  62

3. **Unscrambling Declarations in C**  63
Syntax Only a Compiler Could Love  64
  How a Declaration Is Formed  66
    A Word About structs  68
    A Word About unions  71
    A Word About enums  73
The Precedence Rule  74
  Unscrambling C Declarations by Diagram  75
typedef Can Be Your Friend  78
Difference Between typedef int x[10] and
  \#define x int[10]  80
What typedef struct foo { ... foo } foo; Means  81
The Piece of Code that Understandeth All Parsing  83
  Further Reading  86
Some Light Relief—Software to Bite the Wax Tadpole...  86

4. **The Shocking Truth: C Arrays and Pointers Are NOT the Same!**  95
Arrays Are NOT Pointers!  95
  Why Doesn’t My Code Work?  96
What’s a Declaration? What’s a Definition?  97
  How Arrays and Pointers Are Accessed  98
    What Happens When You “Define as Array/Declare as Pointer”  101
  Match Your Declarations to the Definition  102
Other Differences Between Arrays and Pointers  103
Some Light Relief—Fun with Palindromes!  105

5. **Thinking of Linking**  109
Libraries, Linking, and Loading  110
  Where the Linker Is in the Phases of Compilation  110
The Benefits of Dynamic Linking  113
Five Special Secrets of Linking with Libraries 118
Watch Out for Interpositioning 123
Generating Linker Report Files 128
Some Light Relief—Look Who's Talking: Challenging the Turing Test 129
  Eliza 130
  Eliza Meets the VP 130
  Doctor, Meet Doctor 131
  The Prize in Boston 133
  Conclusions 133
  Postscript 135
  Further Reading 135

6. Poetry in Motion: Runtime Data Structures 137
a.out and a.out Folklore 138
Segments 139
  What the OS Does with Your a.out 142
  What the C Runtime Does with Your a.out 145
    The Stack Segment 145
What Happens When a Function Gets Called:
    The Procedure Activation Record 146
The auto and static keywords 151
  A Stack Frame Might Not Be on the Stack 152
Threads of Control 152
setjmp and longjmp 153
The Stack Segment Under UNIX 155
The Stack Segment Under MS-DOS 156
Helpful C Tools 156
Some Light Relief—Programming Puzzles at Princeton 161
For Advanced Students Only 163

7. Thanks for the Memory 165
The Intel 80x86 Family 165
The Intel 808x6 Memory Model and How It Got That Way 170
Virtual Memory 174
Cache Memory 177
The Data Segment and Heap 181
Memory Leaks 183
  How to Check for a Memory Leak 184
Bus Error, Take the Train 187
  Bus Error 188
  Segmentation Fault 189
Some Light Relief—The Thing King and the Paging Game 195

8. Why Programmers Can't Tell Halloween from Christmas Day 201
  The Potrzebie System of Weights and Measures 201
  Making a Glyph from Bit Patterns 203
  Types Changed While You Wait 205
  Prototype Painfulness 207
    Where Prototypes Break Down 209
  Getting a Char Without a Carriage Return 212
  Implementing a Finite State Machine in C 217
  Software Is Harder than Hardware! 219
  How and Why to Cast 223
Some Light Relief—The International Obfuscated C Code Competition 225

9. More about Arrays 239
  When an Array Is a Pointer 239
  Why the Confusion? 240
    Rule 1: An "Array Name in an Expression" Is a Pointer 243
    Rule 2: C Treats Array Subscripts as Pointer Offsets 244
    Rule 3: An "Array Name as a Function Parameter" Is a Pointer 246
  Why C Treats Array Parameters as Pointers 246
    How an Array Parameter Is Referenced 247
  Indexing a Slice 250
Arrays and Pointers Interchangeability Summary 251
C Has Multidimensional Arrays... 251
...But Every Other Language Calls Them "Arrays of Arrays" 251
How Multidimensional Arrays Break into Components 254
How Arrays Are Laid Out in Memory 256
How to Initialize Arrays 257
Some Light Relief—Hardware/Software Trade-Offs 260

10. More About Pointers 263
  The Layout of Multidimensional Arrays 263
  An Array of Pointers Is an "Illiffe Vector" 265
  Using Pointers for Ragged Arrays 269
  Passing a One-Dimensional Array to a Function 273
Using Pointers to Pass a Multidimensional Array to a Function  273
  Attempt 2  275
  Attempt 3  276
  Attempt 4  277

Using Pointers to Return an Array from a Function  277
Using Pointers to Create and Use Dynamic Arrays  280
Some Light Relief—The Limitations of Program Proofs  287
Further Reading  291

11. You Know C, So C++ is Easy!  293
  Allez-OOP!  293
  Abstraction—Extracting Out the Essential Characteristics of a Thing  296
  Encapsulation—Grouping Together Related Types, Data, and Functions  298
  Showing Some Class—Giving User-Defined Types the Same Privileges as
      Predefined Types  299
  Availability  301
  Declarations  301
  How to Call a Method  304
      Constructors and Destructors  305
  Inheritance—Reusing Operations that Are Already Defined  307
  Multiple Inheritance—Deriving from Two or More Base Classes  311
  Overloading—Having One Name for the Same Action on Different Types  312
  How C++ Does Operator Overloading  313
  Input/Output in C++  314
  Polymorphism—Runtime Binding  315
  Explanation  317
  How C++ Does Polymorphism  318
  Fancy Pants Polymorphism  319
  Other Corners of C++  320
  If I Was Going There, I Wouldn’t Start from Here  322
  It May Be Crufty, but It’s the Only Game in Town  325
  Some Light Relief—The Dead Computers Society  328
  Some Final Light Relief—Your Certificate of Merit!  330
  Further Reading  331

Appendix: Secrets of Programmer Job Interviews  333
  Silicon Valley Programmer Interviews  333
  How Can You Detect a Cycle in a Linked List?  334
  What Are the Different C Increment Statements For?  335
How Is a Library Call Different from a System Call? 338
How Is a File Descriptor Different from a File Pointer? 340
Write Some Code to Determine if a Variable Is Signed or Not 341
What Is the Time Complexity of Printing the Values in a Binary Tree? 342
Give Me a String at Random from This File 343
Some Light Relief—How to Measure a Building with a Barometer 344
Further Reading 346

Index 349