Programming in C[1]

Bharat Kinariwala
University of Hawai'i

Tep Dobry
University of Hawai'i

January 5, 1993

[1]Copyright ©1993 by B. Kinariwala and T. Dobry. All rights reserved.

Contents

Table of Contents i

List of Figures vii

List of Tables xiii

Preface xvii

1 Introduction 1

1.1 Computer System Organization 2

1.1.1 Computer Hardware 2

1.1.2 Computer Software – The Operating System 3

1.1.3 Utility Programs 4

1.1.4 User Programs and Applications 5

1.2 Representing Data and Program Internally 5

1.2.1 Representing Data 6

1.2.2 Main Memory 15

1.2.3 Representing Programs 16

1.3 Designing Programs and the C Language 19

1.3.1 Designing The Algorithm 19

1.3.2 The C Language 23

1.4 Summary 24

1.5 References 25

1.6 Exercises 26

1.7 Problems 28

2 Basic Concepts **29**

2.1 A Simple C Program 30

2.1.1 Developing the Algorithm 30

2.1.2 Translating the Algorithm to C 31

2.1.3 Running the Program 31

2.2 Organization of C Programs — Simple Statements 33

2.2.1 Comment Statements 33

2.2.2 Defining a Function — `main()` 34

2.2.3 Variable Declarations 35

2.2.4 The Assignment Statement 37

2.2.5 Generating Output 40

2.3 Testing the Program 42

2.3.1 Debugging the Program 43

2.3.2 Documenting the Code 47

2.4 Input: Reading Data 47

2.5 More C Statements 51

2.5.1 Making Decisions with Branches 51

2.5.2 Simple Compiler Directives 55

2.5.3 More on Expressions 57

2.5.4 A Simple Loop — `while` 64

2.5.5 Controlling Loop Termination 70

2.5.6 More Complex Loop Constructs — Nested Loops 74

2.6 Common Errors 78

2.7 Summary . . . 81
2.8 Exercises . . . 83
2.9 Problems . . . 86

3 Designing Programs Top Down 89

3.1 Designing the Algorithm with Functions . . . 90
3.1.1 Implementing the Program with Functions . . . 91
3.2 Defining Functions . . . 94
3.2.1 Passing Data to and from Functions . . . 96
3.2.2 Call by Value and Local Variables . . . 99
3.3 Coding Programs for Readability . . . 105
3.3.1 The C Preprocessor . . . 105
3.3.2 Macros . . . 105
3.3.3 Including Header Files . . . 113
3.3.4 Conditional Compilation . . . 116
3.4 Interacting with the Operating System . . . 119
3.4.1 Standard Files and EOF . . . 119
3.4.2 Standard Files and Redirection . . . 125
3.5 Debugging Guidelines . . . 126
3.6 Common Errors . . . 127
3.7 Summary . . . 128
3.8 Exercises . . . 130
3.9 Problems . . . 132

4 Processing Character Data 135

4.1 A New Data Type: `char` . . . 135
4.1.1 The ASCII Character Set . . . 138

4.1.2 Operations on Characters 140

4.1.3 Character I/O Using `getchar()` and `putchar()` 147

4.1.4 Strings vs Characters 150

4.2 Sample Character Processing Functions 151

4.2.1 Converting Letter Characters 152

4.2.2 Converting Digit Characters to Numbers 154

4.2.3 Counting Words 162

4.2.4 Extracting Words 169

4.3 New Control Constructs 172

4.3.1 The `switch` Statement 172

4.3.2 The `break` Statement 178

4.3.3 The `continue` Statement 182

4.4 Mixing Character and Numeric Input 185

4.5 Menu Driven Programs 194

4.6 Common Errors 196

4.7 Summary 199

4.8 Exercises 201

4.9 Problems 203

5 Numeric Data Types and Expression Evaluation 207

5.1 Representing Numbers 207

5.1.1 Signed and Unsigned Integer Types 209

5.1.2 Single and Double Precision Floating Point Numbers 211

5.2 New Control Constructs 212

5.2.1 The `for` Statement 212

5.2.2 The `do...while` Statement 216

5.3 Scalar Data Types 224

5.3.1 Data Type `void` 224
5.3.2 Enumeration 226
5.3.3 Defining User Types: `typedef` 228
5.4 Operators and Expression Evaluation 229
5.4.1 Precedence and Associativity 232
5.4.2 The Data Type of the Result 234
5.4.3 Some New Operators 236
5.5 Common Errors 244
5.6 Summary 246
5.7 Exercises 248
5.8 Problems 251

6 Pointers **255**

6.1 What is a Pointer? 255
6.1.1 Data vs Address 256
6.1.2 Indirect Access of Values 259
6.2 Passing Pointers to Functions 268
6.2.1 Indirectly Incrementing a Variable 268
6.2.2 Computing the Square and Cube 268
6.2.3 A function to Swap Values 276
6.3 Returning to the Payroll Task with Pointers 276
6.4 Common Errors 287
6.5 Summary 288
6.6 Exercises 290
6.7 Problems 292

7 Arrays **293**

7.1 A Compound Data Type — *array* . . . 294
7.1.1 Declaring Arrays . . . 294
7.1.2 Character Strings as Arrays . . . 298
7.2 Passing Arrays to Functions . . . 302
7.3 Arrays, Pointers, Pointer Arithmetic . . . 306
7.3.1 Pointers: Increment and Decrement . . . 311
7.3.2 Array Names vs Pointer Variables . . . 315
7.4 String Assignment and I/O . . . 319
7.5 Array Initializers . . . 321
7.6 Arrays for Databases . . . 323
7.7 Common Errors . . . 327
7.8 Summary . . . 329
7.9 Exercises . . . 332
7.10 Problems . . . 336

8 Functions and Files **339**

9 Sorting and Searching **341**

10 String Processing **343**

11 Two Dimensional Arrays **345**

12 Structures and Unions **347**

13 File Input/Output **349**

14 Storage Class and Scope **351**

15 Engineering Programming Examples **353**

A C Language Reference **355**

B ANSI C vs Old C **357**

C The C Standard Library **359**

List of Figures

1.1 Computer System Block Diagram . . . 2

1.2 Memory and Addresses . . . 16

1.3 CPU and Memory Configuration . . . 17

1.4 Machine and Assembly Language Program Fragment . . . 19

1.5 Structural Diagram for Payroll Task . . . 20

1.6 Flow Chart for Payroll Task . . . 21

2.1 Code for pay0.c . . . 32

2.2 Allocation of Memory Cells or Objects . . . 36

2.3 Assignment of Values . . . 37

2.4 Computation of `pay` . . . 38

2.5 Program Trace for pay0.c . . . 46

2.6 Keyboard Buffer . . . 49

2.7 Code for pay1.c . . . 50

2.8 If statement control flow . . . 53

2.9 Code for pay2.c . . . 59

2.10 Control Flow for `while` statement . . . 66

2.11 Coding a While Loop . . . 67

2.12 Code for pay3.c . . . 70

2.13 Code for pay4.c . . . 73

2.14 Code for prime.c . . . 77

3.1 Structural Diagram for Payroll Task 90
3.2 Code for pay5.c driver 93
3.3 Function Call to `calc_pay()` 94
3.4 Code for `calc_pay()` 95
3.5 Function Call Trace 97
3.6 Trace for `calc_pay()` 99
3.7 Call by value variable allocation 101
3.8 Local Variables in Blocks 103
3.9 Driver for `niceday.c` 111
3.10 Functions for `niceday.c` 112
3.11 Using Directives in `niceday.c` 121
3.12 End of File Marker 122
3.13 Code for maxabs.c 124

4.1 Code for copy0.c 137
4.2 Code for ASCII Attributes 145
4.3 Printing character representations 146
4.4 Alternate code for attributes program 148
4.5 Using `getchar()` and `putchar()` 150
4.6 Strings 151
4.7 Code for upper case 153
4.8 Code for Character Utilities 155
4.9 Header file for Character Utilities 156
4.10 Code for `getint()` 158
4.11 Revised Character Utility Header File 163
4.12 Revised Character Utility Code 164
4.13 Code for Count Words Driver 166

4.14 Header Files for Word Count 167

4.15 Code for Word Count Utility Functions 168

4.16 Code fore extracting words 171

4.17 Control Flow for switch statement 174

4.18 Code for `vowelp()` Using a `switch` Statement 175

4.19 Code for `encrypt.c` 176

4.20 Implementing `print_next()` Using a `switch` Statement 178

4.21 New Implementation of `print_category` using `switch` 179

4.22 Extracting Words Using `break` 181

4.23 Code for Revised `encrypt.c` 184

4.24 Code for Testing `scanf()` 185

4.25 Revised Code for Reading Integers 188

4.26 Mixing Numeric and Character Data 190

4.27 Revised Code for Mixing Data 192

4.28 A Better Revision for Mixing Data 193

4.29 Code for menu driven program 195

4.30 Menu Driven Functions 197

5.1 Control Flow of `for` Loop 213

5.2 Code for factorial 214

5.3 Control Flow of `do...while` Loop 216

5.4 Code for Square Root 219

5.5 Code for Math Utilities 220

5.6 Modified Square Root Driver 222

5.7 Modified Square Root Utilities 223

5.8 Code for Simple Postfix Calculator 230

5.9 Code for `get_operator()` 231

5.10 Factorial Function Using Composite Operators 239
5.11 Function `maxdbl` Using a Conditional Expression 240
5.12 Revised Fibonacci 241
5.13 Testing `sizeof` Operator 244

6.1 Declaring Pointer Variables 258
6.2 Declaration of Pointer Variables 260
6.3 Assignments of pointers 260
6.4 Effect of Pointer to Pointer Assignment 261
6.5 Effect of Pointer Reassignment 262
6.6 Effect of Indirect Pointer Access and Assignment 263
6.7 Effect of Indirect Assignment 263
6.8 Effect of Indirect Pointer Access and Assignment 264
6.9 Example Code with Direct and Indirect Access 266
6.10 Trace for Direct and Indirect Access 267
6.11 Code for Indirect Access by a Function 269
6.12 Trace for Indirect Access by a Function 270
6.13 Code for Indirectly Returned Values 271
6.14 Trace for `sqcube` 272
6.15 Trace for `sqcube` 273
6.16 Trace for `sqcube` 274
6.17 Trace for `sqcube` 275
6.18 Code for a Function, `swap()` 277
6.19 Trace for `swap()` 278
6.20 Trace for `swap()` 279
6.21 Trace for `swap()` 280
6.22 Trace for `swap()` 281

6.23 Header file `payutil.h` 283

6.24 Code for the Driver for `pay6.c` 284

6.25 Code for `print_data()` and `print_pay()` 285

6.26 Code for `calc_pay()` 286

6.27 Code for `get_data()` 286

7.1 Code for `scores.c` 295

7.2 An Array of size MAX 296

7.3 A String Stored in Memory 299

7.4 Code for `string.c` 300

7.5 Code fore `scores.c` 304

7.6 Code for `string2.c` 305

7.7 Pointer Arithmetic 307

7.8 Array Pointers as Function Parameters 309

7.9 Pointer to a Sub-array 312

7.10 Pointer Variables and Arrays 317

7.11 Pointer Variables and Strings 318

7.12 Data Record Spread Over Several Arrays 323

7.13 Code for `paytab.c` 324

7.14 Code for `payutil.c` 325

List of Tables

1.1 Number Representations 14

2.1 Arithmetic Operators 40

2.2 Relational Operators 55

2.3 Precedence and Associativity of Operators 59

2.4 Logical Operator Symbols in C 62

2.5 Truth Table for Logical Combinations 63

4.1 Escape Sequences 136

4.2 ASCII Table 139

4.3 Escape sequences with Octal & Hexadecimal values 141

5.1 Precedence and Associativity Table 233

5.2 Composite Assignment Operators and Their Equivalents 238

5.3 Space allocation in Bytes for data types 243

7.1 Pointer Arithmetic and Indirect Access 307

Preface

The C language has boomed in popularity and availability since its creation in the 70's. It has largely become the language of choice for systems programming as well as general purpose programming in both the numeric and symbolic realms. As a result, all programmers today should have some working knowledge of C, particularly in engineering.

This book is intended to be a first text in programming in general with emphasis on the C language. It is meant for students with little or no previous programming experience and as such, a primary focus is on the top down *design* of programs, beginning with the development of an algorithm, proceeding to the translation of the algorithm into a programming language (C), and the subsequent testing and debugging of the resulting code. Throughout the text, emphasis is placed on organization and readability of code as well as debugging aids in program development. In addition, understandable and functional user interfaces are described.

As an introductory text on programming, our approach is to motivate the introduction of features of the language through example problems. We start with meaningful but simple tasks, develop an algorithm to solve the task and then introduce the necessary language constructs to implement the algorithm. We then refine the task, adding complexity or desirable features to motivate introduction of new language constructs. As such, the text is not meant to be a C reference manual, but a text on program design utilizing the available language features to implement the design. However, for student's reference, the key constructs introduced are summarized at the end of each chapter. The intent is for the student to be able to design and code programs from the very beginning.

The book is organized as follows:

- Chapter 1 is an introduction to computers and some of the terminology used throughout the text.
- Chapter 2 begins the development of a simple C program and the introduction to the organization and basic statements of the language.
- Chapter 3 stresses the top down approach to design and introduces functions at an early stage to emphasize their relation to algorithms.
- Chapter 4 introduces the character data type and algorithms for processing characters.

- Chapter 5 presents numeric data types and their limitations and discusses the details of C expression evaluation.
- Chapter 6 addresses the important concept of pointers and their use in C in functions.
- Chapter 7 introduces compound data types with arrays and discusses their relation with pointers.
- Chapter 8 describes some of the standard library functions provided in C for character and math processing as well as giving a detailed description of the standard I/O functions printf() and scanf() and their variations for file I/O.
- Chapter 9 presents some standard sorting and searching techniques.
- Chapter 10 describes the powerful string processing utilities in C and the concept of libraries of functions.
- Chapter 11 returns to arrays presenting two dimensional arrays.
- Chapter 12 discusses the remaining compound data type; structures and unions.
- Chapter 13 presents advanced file Input/Output features of the language.
- Chapter 14 describes the memory organization of C programs and discusses the details of storage classes and scope.
- Finally, Chapter 15 provides several examples of algorithms useful in Engineering computation. It makes use of the concepts presented in earlier Chapters and these examples may be discussed with the appropriate Chapter.

In addition, three Appendices are provided as follows:

- Appendix A provides a summary of the C language constructs discussed in the text.
- Appendix B contrasts the language features of ANSI C as presented in this text to "old" C which is still available on many Unix systems.
- Appendix C summarizes the standard library functions available in C.