

Programming Abstractions in C++

Eric S. Roberts and Julie Zelenski

This course reader has had an interesting evolutionary history that in some ways mirrors the genesis of the C++ language itself. Just as Bjarne Stroustrup's first version of C++ was implemented on top of a C language base, this reader began its life as Eric Roberts's textbook *Programming Abstractions in C* (Addison-Wesley, 1998). In 2002-03, Julie Zelenski updated it for use with the C++ programming language, which we began using in CS106B and CS106X during that year.

Although the revised text worked fairly well at the outset, CS106B and CS106X have evolved in recent years so that their structure no longer tracks the organization of the book. This year, we're engaged in the process of rewriting the book so that students in these courses can use it as both a tutorial and a reference. As always, that process takes a considerable amount of time, and there are likely to be some problems as we update the reader. At the same time, we're convinced that the material in CS106B and CS106X is tremendously exciting and will be able to carry us through a quarter or two of instability, and we will end up with an even better course in the future.

We want to thank our colleagues at Stanford, several generations of section leaders (with special thanks to Dan Bentley and Keith Schwarz), and so many students over the years—all of whom have helped make it so exciting to teach this wonderful material.

Programming Abstractions in C++

Chapter 1. An Overview of C++ 1

1.1 What is C++? 2

The object-oriented paradigm; The compilation process

1.2 The structure of a C++ program 5

Comments; Library inclusions; Program-level definitions; Function prototypes; The main program; Function definitions

1.3 Variables, values, and types 9

Naming conventions; Local and global variables; The concept of a data type; Integer types; Floating-point types; Text types; Boolean type; Simple input and output

1.4 Expressions 16

Precedence and associativity; Mixing types in an expression; Integer division and the remainder operator; Type casts; The assignment operator; Increment and decrement operators; Boolean operators

1.5 Statements 24

Simple statements; Blocks; The **if** statement; The **switch** statement; The **while** statement; The **for** statement

1.6 Functions 32

Returning results from functions; Function definitions and prototypes; The mechanics of the function-calling process; Passing parameters by reference

Summary 38

Review questions 39

Programming exercises 41

Chapter 2. Data Types in C++ 45

2.1 Enumeration types 46

Internal representation of enumeration types; Scalar types

2.2 Data and memory 49

Bits; bytes; and words; Memory addresses

2.3 Pointers 51

Using addresses as data values; Declaring pointer variables; The fundamental pointer operations

2.4 Arrays 56

Array declaration; Array selection; Effective and allocated sizes; Initialization of arrays; Multidimensional arrays

2.5 Pointers and arrays 64

The relationship between pointers and arrays

2.6 Records 67

Defining a new structure type; Declaring structure variables; Record selection; Initializing records; Pointers to records

2.7 Dynamic allocation 71

Coping with memory limitations; Dynamic arrays; Dynamic records

Summary 74
Review questions 74
Programming exercises 77

Chapter 3. Libraries and Interfaces 85

3.1 The concept of an interface 86

Interfaces and implementations; Packages and abstractions; Principles of good interface design

3.2 A random number interface 89

The structure of the `random.h` interface; Constructing a client program; The ANSI functions for random numbers; The `random.cpp` implementation

3.3 Strings 98

The data type `string`; Operations on the `string` type ; The `strutils.h` interface; An aside about C-style strings

3.4 Standard I/O and file streams 105

Data files; Using file streams in C++; Standard streams; Formatted stream output; Formatted stream input; Single character I/O; Rereading characters from an input file; Line-oriented I/O

3.5 Other ANSI libraries 112

Summary 113
Review questions 113
Programming exercises 116

Chapter 4. Using Abstract Data Types 123

4.1 The `vector` class 125

Specifying the base type of a `vector`; Declaring a new `vector` object; Operations on the `vector` class; Iterating through the elements of a `vector`; Passing a `vector` as a parameter

4.2 The `grid` class 131

4.3 The `stack` class 133

The structure of the `stack` class

4.4 The `queue` class 136

Simulations and models; The waiting-line model; Discrete time; Events in simulated time; Implementing the simulation

4.5 The `map` class 146

The structure of the `map` class; Using maps in an application; Maps as associative arrays

4.6 The `lexicon` class 151

The structure of the `lexicon` class; A simple application of the `lexicon` class; Why are lexicons useful if maps already exist

4.7 The `scanner` class 154

Setting scanner options

4.8 Iterators 156

The standard iterator pattern; Iteration order; A simple iterator example; Computing word frequencies

Summary 163
Review questions 164
Programming exercises 165

Chapter 5. Introduction to recursion 173

5.1 A simple example of recursion 174

5.2 The factorial function 176

The recursive formulation of `fact`; Tracing the recursive process; The recursive leap of faith

5.3 The Fibonacci function 181

Computing terms in the Fibonacci sequence; Gaining confidence in the recursive implementation; Recursion is not to blame

5.4 Other examples of recursion 187

Detecting palindromes; Binary search; Mutual recursion

5.5 Thinking recursively 192

Maintaining a holistic perspective; Avoiding the common pitfalls

Summary 194

Review questions 195

Programming exercises 197

Chapter 6. Recursive procedures 201

6.1 The Tower of Hanoi 202

Framing the problem; Finding a recursive strategy; Validating the strategy; Coding the solution; Tracing the recursive process

6.2 Generating permutations 211

The recursive insight

6.3 Graphical applications of recursion 213

The graphics library; An example from computer art; Fractals

Summary 224

Review questions 225

Programming exercises 226

Chapter 7. Backtracking algorithms 235

7.1 Solving a maze by recursive backtracking 236

The right-hand rule; Finding a recursive approach; Identifying the simple cases; Coding the maze solution algorithm; Convincing yourself that the solution works

7.2 Backtracking and games 245

The game of nim; A generalized program for two-player games; The minimax strategy; Implementing the minimax algorithm; Using the general strategy to solve a specific game

Summary 269

Review questions 270

Programming exercises 271

Chapter 8. Algorithmic analysis 277

8.1 The sorting problem 278

The selection sort algorithm; Empirical measurements of performance; Analyzing the performance of selection sort

8.2 Computational complexity and big-O notation 282

Big-O notation; Standard simplifications of big-O; Predicting computational complexity from code structure; Worst-case versus average-case complexity; A formal definition of big-O

8.3 Recursion to the rescue 288

The power of divide-and-conquer strategies; Merging two vectors; The merge sort algorithm; The computational complexity of merge sort; Comparing N^2 and $N \log N$ performance

8.4 Standard complexity classes 294

8.5 The Quicksort algorithm 296

Partitioning the vector; Analyzing the performance of Quicksort

8.6 Mathematical induction 301

Summary 304

Review questions 305

Programming exercises 307

Chapter 9. Classes and objects 313

9.1 A simple example of a class definition 314

Defining a `Point` class; Implementing methods in a class; Constructors and destructors; The keyword `this`

9.2 Implementing a specialized version of the `stack` class 319

Defining the `CharStack` interface; Representing the stack data; The advantages of object encapsulation; Removing the maximum size limitation; Object copying

9.3 Implementing the `Scanner` class 328

Summary 328

Review questions 334

Programming exercises 335

Chapter 10. Efficiency and Data Representation 339

10.1 The concept of an editor buffer 340

10.2 Defining the buffer abstraction 341

The public interface of the `EditorBuffer` class; Coding the editor application

10.3 Implementing the editor using arrays 345

Defining the private data representation; Implementing the buffer operations; Assessing the computational complexity of the array implementation

10.4 Implementing the editor using stacks 352

Defining the private data representation for the stack-based buffer; Implementing the buffer operations; Comparing computational complexities

[Click here to download full PDF material](#)