

Table of Contents

About the Authorxvii

About the Technical Reviewerxix

Acknowledgmentsxxi

Introductionxxiii

Chapter 1: Getting Started 1

 A simple program..... 1

 Spacing..... 4

 Creating an SSDL project..... 6

 ...in Visual Studio..... 6

 ...with g++ 17

 How not to be miserable 19

 Shapes and the functions that draw them..... 21

 Antibugging 30

 consts and colors..... 31

 Text 35

 sout, escape sequences, and fonts 35

 SSDL_RenderText, SSDL_RenderTextCentered 39

Chapter 2: Images and Sound 43

 Images and changing window characteristics 43

 Antibugging 49

 Multiple images together 51

 Adding transparency with GIMP..... 52

 Sound..... 57

 Antibugging 60

TABLE OF CONTENTS

Chapter 3: Numbers..... 61

 Variables 61

 Constants 63

 When to use constants, not literal values..... 64

 Math operators..... 65

 Integer division..... 65

 Assignment (=) operators..... 66

 A diving board example 67

 The no-worries list for math operators..... 70

 Built-in functions and casting 71

 Antibugging 76

Chapter 4: Mouse, and if..... 79

 Mouse functions 79

 Antibugging 82

 if..... 83

 Coercion and if conditions (int's dirty little secret)..... 86

 Combining conditions with &&, ||, and !..... 86

 Antibugging 87

 Boolean values and variables 90

 A hidden object game 92

Chapter 5: Loops, Input, and char..... 101

 Keyboard input..... 101

 Antibugging 103

 while and do-while 105

 Loops with SSDL..... 107

 break and continue..... 108

 Antibugging 109

 for loops..... 112

 Increment operators 113

 An example: averaging numbers..... 114

 Antibugging 116

chars and ctype.....	118
switch	123
Antibugging	124
Chapter 6: Algorithms and the Development Process	127
Adventures in robotic cooking	127
Writing a program, from start to finish	131
Requirements: What do we want to do?	131
Algorithm: How do we do it?	132
Walkthrough: Will it do it?.....	134
Coding: putting it all into C++ (plus: commenting the lazy way).....	134
Chapter 7: Functions	141
Functions that return values	141
Functions that return nothing	148
Global variables.....	152
Antibugging	153
How to write a function in four easy steps (and call it in one).....	156
Why have functions, anyway?	160
Chapter 8: Functions (Continued)	171
Random numbers.....	171
Making a random number generator.....	171
Using the built-in random number generator	174
Antibugging	177
Boolean functions	179
& parameters	180
Antibugging	185
Identifier scope	186
A final note on algorithms.....	189

Chapter 9: Using the Debugger 191

 A flawed program 191

 Breakpoints and watched variables 196

 Visual Studio..... 196

 ddd 198

 gdb 199

 Fixing the stripes 199

 Going into functions 199

 Visual Studio..... 199

 ddd 201

 gdb 202

 Fixing the stars 202

 Wrap-up 203

 Antibugging 203

 Bottom-up testing 204

 More on antibugging..... 205

Chapter 10: Arrays and enum 207

 Arrays..... 207

 Arrays' dirty little secret: using memory addresses 210

 Antibugging 211

 Arrays as function parameters..... 212

 Array parameters that change, or don't..... 213

 Array parameters and reusability 213

 Antibugging 214

 Enumeration types 215

 enum class 217

 Antibugging 218

 Multidimensional arrays 219

 Displaying the board..... 220

 Arrays of more than two dimensions..... 223

 Antibugging 224

Chapter 11: Animation with structs and Sprites 227

 structs 227

 Cool struct tricks 231

 Making a movie with struct and while 232

 Sprites 239

 Antibugging 244

Chapter 12: Making an Arcade Game: Input, Collisions, and Putting It All Together 245

 Determining input states 245

 Mouse 245

 Keyboard 246

 Antibugging 248

 Events 248

 Cooldowns and lifetimes 251

 Collisions 254

 The big game 255

 Antibugging 269

Chapter 13: Standard I/O and File Operations 273

 Standard I/O programs 273

 Compiling in Microsoft Visual Studio 274

 Compiling with g++ 278

 File I/O (optional) 280

 cin and cout as files 280

 Using file names 287

TABLE OF CONTENTS

Chapter 14: Character Arrays and Dynamic Memory 293

Character arrays 293

 Antibugging 296

Dynamic allocation of arrays 299

 Antibugging 302

Using the * notation 303

 Antibugging 307

Chapter 15: Classes 309

Writing classes..... 309

Constructors..... 312

 Antibugging 316

const objects, const member functions..... 318

 Antibugging 319

...and const parameters..... 319

Multiple constructors 320

 Copy constructors..... 320

 Default constructors 321

 Conversion constructors..... 322

 Summary 323

 Antibugging 323

Default parameters for code reuse 324

Date program (so far)..... 325

Chapter 16: Classes (Continued) 329

inline functions for efficiency..... 329

Access functions..... 331

Separate compilation and include files..... 332

 What happens in separate compilation 333

 Writing your .h file 334

 Backing up a multi-file project 337

 Antibugging 337

Multiple-file projects in Microsoft Visual Studio 339

Multiple-file projects in g++ 340

 Command line: more typing, less thinking 340

 Makefiles: more thinking, less typing 340

 Antibugging 344

Final Date program 344

static members (optional) 350

Chapter 17: Operators 353

 The basic string class 353

 Destructors 355

 Binary and unary operators 356

 Assignment operators and *this..... 358

 Antibugging 360

 Arithmetic operators 361

 [] and () 364

 >> and <<: operators that aren't class members 365

 ++ and -- 368

 Explicit call to constructor 369

 Final String program 370

 #include <string> 376

Chapter 18: Exceptions, Move Constructors, Recursion, and O Notation 377

 Exceptions..... 377

 Move constructors and move = (optional) 382

 Recursion (optional; used in the next section) 384

 Antibugging 388

 Algorithm analysis and O notation (optional) 389

Chapter 19: Inheritance 393

 The basics of inheritance..... 393

 Constructors and destructors 398

 Inheritance as a concept..... 400

TABLE OF CONTENTS

Classes for card games 401

 An inheritance hierarchy 404

 private inheritance 408

 Hiding an inherited member function 410

 A game of Montana 411

Chapter 20: Templates 423

 Function templates 423

 Antibugging 425

 The Vector class 426

 Efficiency (optional) 431

 Making Vector a template 432

 Antibugging 436

 Unusual class templates 437

 #include <vector> 439

Chapter 21: Virtual Functions and Multiple Inheritance..... 441

 Virtual functions 441

 Behind the scenes 446

 Pure virtual functions and abstract base classes 447

 Why virtual functions often mean using pointers 447

 Virtual destructors 452

 Inheritance and move ctor/move = (optional) 454

 Antibugging 455

 Multiple inheritance 457

 Antibugging 459

Chapter 22: Linked Lists 463

 What lists are and why have them..... 463

 List<T>::List () 467

 void List<T>::push_front (const T& newElement); 468

 void List<T>::pop_front() 469

 List<T>::~~List() 472

->: a bit of syntactic sugar..... 472

More friendly syntax: pointers as conditions 472

The linked list template 473

 Antibugging 477

#include <list>..... 478

Chapter 23: The Standard Template Library 479

 Iterators..... 479

 ...with vector, too..... 483

 const and reverse iterators 484

 Antibugging 485

 Getting really lazy: ranges and auto..... 486

 initializer_lists (optional)..... 487

 algorithm (optional)..... 489

 The erase-remove idiom 490

 Antibugging 490

Chapter 24: Building Bigger Projects 493

 Namespaces 493

 Conditional compilation 494

 Libraries 495

 g++ 496

 Microsoft Visual Studio..... 498

Chapter 25: History..... 509

 Simula 67 509

 Smalltalk 509

 What “object-oriented” is 510

 C..... 511

 C++ 511

 Standards..... 512

TABLE OF CONTENTS

Chapter 26: Esoterica (Recommended) 513

 sstream: using strings like cin/cout 513

 iomanip: formatted output 516

 Command-line arguments 522

 Debugging with command-line arguments in Visual Studio 525

 Debugging with command-line arguments in Unix 525

 static_cast et al 527

 Defaulted constructors and = 528

 constexpr and static_assert: moving work to compile time 529

 User-defined literals: automatic conversion between systems of measurement 533

 Lambda functions for one-time use 536

 Lambda captures 537

 Structured bindings and tuples: returning multiple values at once 542

 Smart pointers 546

 unique_ptr 546

 shared_ptr 552

 Antibugging 553

 Bit twiddling: &, |, ~, and 553

 Antibugging 559

Chapter 27: Esoterica (Not So Recommended)..... 561

 protected sections, protected inheritance 561

 Template specialization 565

 friends, and why you shouldn't have any..... 566

 User-defined conversions 571

Chapter 28: C 573

 Compiling C..... 574

 I/O 575

 printf 575

 scanf, and the address-of (&) operator..... 575

 fprintf and fscanf; fopen and fclose 578

 sprintf and sscanf; fputs and fgets..... 580

Summary	583
Antibugging	584
Parameter passing with *	584
Antibugging	588
Dynamic memory	588
Chapter 29: Moving On with SDL	591
Writing code	593
Compiling	597
Further resources	598
Appendix A: SDL/SSDL Setup Issues	599
Unix	599
g++	599
SDL	599
SSDL	600
Making your own Makefiles	600
Antibugging	600
MinGW	601
g++	601
SDL and SSDL	601
Making your own Makefiles	601
Antibugging	601
Microsoft Visual Studio	602
SDL/SSDL	602
Making your own project files	602
Antibugging	603
Sound	604
Appendix B: Operators	605
Associativity	605
Precedence	605
Overloading	606

TABLE OF CONTENTS

Appendix C: ASCII Codes 607

Appendix D: Fundamental Types..... 609

Appendix E: Escape Sequences 611

Appendix F: Basic C Standard Library 613

 cmath 613

 ctype 614

 cstdlib 615

Appendix G: Common Debugger Commands 617

 Microsoft Visual Studio 617

 gdb/ddd..... 618

Appendix H: SSDL Reference 619

 Updating the screen..... 619

 Added types 619

 Clearing the screen 620

 Colors..... 620

 Drawing..... 620

 Images 621

 Mouse, keyboard, and events 622

 Music 623

 Quit messages 624

 Sounds 625

 Sprites..... 626

 Text 628

 Time and synchronization 629

 Window 630

References..... 631

Index..... 633