

Contents

Appendices E through I are PDF documents posted online at the book's Companion Website (located at www.pearsonhighered.com/deitel).

Preface

xxi

| | | |
|----------|--|-----------|
| 1 | Introduction to Computers, the Internet and the Web | I |
| 1.1 | Introduction | 2 |
| 1.2 | Computers: Hardware and Software | 3 |
| 1.3 | Computer Organization | 4 |
| 1.4 | Personal, Distributed and Client/Server Computing | 5 |
| 1.5 | The Internet and the World Wide Web | 5 |
| 1.6 | Machine Languages, Assembly Languages and High-Level Languages | 6 |
| 1.7 | History of C | 7 |
| 1.8 | C Standard Library | 8 |
| 1.9 | C++ | 9 |
| 1.10 | Java | 9 |
| 1.11 | Fortran, COBOL, Pascal and Ada | 10 |
| 1.12 | BASIC, Visual Basic, Visual C++, C# and .NET | 10 |
| 1.13 | Key Software Trend: Object Technology | 11 |
| 1.14 | Typical C Program Development Environment | 12 |
| 1.15 | Hardware Trends | 14 |
| 1.16 | Notes About C and This Book | 15 |
| 1.17 | Web Resources | 16 |
| 2 | Introduction to C Programming | 23 |
| 2.1 | Introduction | 24 |
| 2.2 | A Simple C Program: Printing a Line of Text | 24 |
| 2.3 | Another Simple C Program: Adding Two Integers | 28 |
| 2.4 | Memory Concepts | 33 |
| 2.5 | Arithmetic in C | 34 |
| 2.6 | Decision Making: Equality and Relational Operators | 38 |
| 3 | Structured Program Development in C | 54 |
| 3.1 | Introduction | 55 |
| 3.2 | Algorithms | 55 |

✘ Contents

| | | |
|------|---|----|
| 3.3 | Pseudocode | 55 |
| 3.4 | Control Structures | 56 |
| 3.5 | The <code>if</code> Selection Statement | 58 |
| 3.6 | The <code>if...else</code> Selection Statement | 59 |
| 3.7 | The <code>while</code> Repetition Statement | 63 |
| 3.8 | Formulating Algorithms Case Study 1: Counter-Controlled Repetition | 64 |
| 3.9 | Formulating Algorithms with Top-Down, Stepwise Refinement Case Study 2: Sentinel-Controlled Repetition | 66 |
| 3.10 | Formulating Algorithms with Top-Down, Stepwise Refinement Case Study 3: Nested Control Structures | 73 |
| 3.11 | Assignment Operators | 77 |
| 3.12 | Increment and Decrement Operators | 78 |

4 C Program Control 97

| | | |
|------|--|-----|
| 4.1 | Introduction | 98 |
| 4.2 | Repetition Essentials | 98 |
| 4.3 | Counter-Controlled Repetition | 99 |
| 4.4 | <code>for</code> Repetition Statement | 100 |
| 4.5 | <code>for</code> Statement: Notes and Observations | 103 |
| 4.6 | Examples Using the <code>for</code> Statement | 103 |
| 4.7 | <code>switch</code> Multiple-Selection Statement | 107 |
| 4.8 | <code>do...while</code> Repetition Statement | 113 |
| 4.9 | <code>break</code> and <code>continue</code> Statements | 114 |
| 4.10 | Logical Operators | 116 |
| 4.11 | Confusing Equality (<code>==</code>) and Assignment (<code>=</code>) Operators | 119 |
| 4.12 | Structured Programming Summary | 121 |

5 C Functions 140

| | | |
|------|---|-----|
| 5.1 | Introduction | 141 |
| 5.2 | Program Modules in C | 141 |
| 5.3 | Math Library Functions | 142 |
| 5.4 | Functions | 144 |
| 5.5 | Function Definitions | 144 |
| 5.6 | Function Prototypes | 148 |
| 5.7 | Function Call Stack and Activation Records | 151 |
| 5.8 | Headers | 151 |
| 5.9 | Calling Functions By Value and By Reference | 152 |
| 5.10 | Random Number Generation | 153 |
| 5.11 | Example: A Game of Chance | 158 |
| 5.12 | Storage Classes | 161 |
| 5.13 | Scope Rules | 164 |
| 5.14 | Recursion | 167 |
| 5.15 | Example Using Recursion: Fibonacci Series | 170 |
| 5.16 | Recursion vs. Iteration | 174 |

| | | |
|----------|--|------------|
| 6 | C Arrays | 195 |
| 6.1 | Introduction | 196 |
| 6.2 | Arrays | 196 |
| 6.3 | Defining Arrays | 198 |
| 6.4 | Array Examples | 198 |
| 6.5 | Passing Arrays to Functions | 212 |
| 6.6 | Sorting Arrays | 216 |
| 6.7 | Case Study: Computing Mean, Median and Mode Using Arrays | 218 |
| 6.8 | Searching Arrays | 223 |
| 6.9 | Multiple-Subscripted Arrays | 229 |
| | | |
| 7 | C Pointers | 253 |
| 7.1 | Introduction | 254 |
| 7.2 | Pointer Variable Definitions and Initialization | 254 |
| 7.3 | Pointer Operators | 255 |
| 7.4 | Passing Arguments to Functions by Reference | 257 |
| 7.5 | Using the <code>const</code> Qualifier with Pointers | 261 |
| 7.6 | Bubble Sort Using Call-by-Reference | 267 |
| 7.7 | <code>sizeof</code> Operator | 270 |
| 7.8 | Pointer Expressions and Pointer Arithmetic | 273 |
| 7.9 | Relationship between Pointers and Arrays | 275 |
| 7.10 | Arrays of Pointers | 280 |
| 7.11 | Case Study: Card Shuffling and Dealing Simulation | 280 |
| 7.12 | Pointers to Functions | 285 |
| | | |
| 8 | C Characters and Strings | 309 |
| 8.1 | Introduction | 310 |
| 8.2 | Fundamentals of Strings and Characters | 310 |
| 8.3 | Character-Handling Library | 312 |
| 8.4 | String-Conversion Functions | 317 |
| 8.5 | Standard Input/Output Library Functions | 322 |
| 8.6 | String-Manipulation Functions of the String-Handling Library | 326 |
| 8.7 | Comparison Functions of the String-Handling Library | 329 |
| 8.8 | Search Functions of the String-Handling Library | 331 |
| 8.9 | Memory Functions of the String-Handling Library | 337 |
| 8.10 | Other Functions of the String-Handling Library | 341 |
| | | |
| 9 | C Formatted Input/Output | 356 |
| 9.1 | Introduction | 357 |
| 9.2 | Streams | 357 |
| 9.3 | Formatting Output with <code>printf</code> | 357 |
| 9.4 | Printing Integers | 358 |
| 9.5 | Printing Floating-Point Numbers | 359 |

| | | |
|------|---|-----|
| 9.6 | Printing Strings and Characters | 361 |
| 9.7 | Other Conversion Specifiers | 362 |
| 9.8 | Printing with Field Widths and Precision | 363 |
| 9.9 | Using Flags in the printf Format Control String | 366 |
| 9.10 | Printing Literals and Escape Sequences | 368 |
| 9.11 | Reading Formatted Input with scanf | 369 |

10 C Structures, Unions, Bit Manipulations and Enumerations 382

| | | |
|-------|---|-----|
| 10.1 | Introduction | 383 |
| 10.2 | Structure Definitions | 383 |
| 10.3 | Initializing Structures | 386 |
| 10.4 | Accessing Structure Members | 386 |
| 10.5 | Using Structures with Functions | 388 |
| 10.6 | typedef | 388 |
| 10.7 | Example: High-Performance Card Shuffling and Dealing Simulation | 389 |
| 10.8 | Unions | 391 |
| 10.9 | Bitwise Operators | 394 |
| 10.10 | Bit Fields | 403 |
| 10.11 | Enumeration Constants | 406 |

11 C File Processing 417

| | | |
|-------|---|-----|
| 11.1 | Introduction | 418 |
| 11.2 | Data Hierarchy | 418 |
| 11.3 | Files and Streams | 420 |
| 11.4 | Creating a Sequential-Access File | 421 |
| 11.5 | Reading Data from a Sequential-Access File | 426 |
| 11.6 | Random-Access Files | 430 |
| 11.7 | Creating a Random-Access File | 431 |
| 11.8 | Writing Data Randomly to a Random-Access File | 433 |
| 11.9 | Reading Data from a Random-Access File | 436 |
| 11.10 | Case Study: Transaction-Processing Program | 437 |

12 C Data Structures 454

| | | |
|------|-----------------------------|-----|
| 12.1 | Introduction | 455 |
| 12.2 | Self-Referential Structures | 456 |
| 12.3 | Dynamic Memory Allocation | 456 |
| 12.4 | Linked Lists | 458 |
| 12.5 | Stacks | 466 |
| 12.6 | Queues | 472 |
| 12.7 | Trees | 478 |

13 C Preprocessor 495

| | | |
|------|--------------|-----|
| 13.1 | Introduction | 496 |
|------|--------------|-----|

| | | |
|-------|--|-----|
| 13.2 | <code>#include</code> Preprocessor Directive | 496 |
| 13.3 | <code>#define</code> Preprocessor Directive: Symbolic Constants | 496 |
| 13.4 | <code>#define</code> Preprocessor Directive: Macros | 497 |
| 13.5 | Conditional Compilation | 499 |
| 13.6 | <code>#error</code> and <code>#pragma</code> Preprocessor Directives | 500 |
| 13.7 | <code>#</code> and <code>##</code> Operators | 500 |
| 13.8 | Line Numbers | 501 |
| 13.9 | Predefined Symbolic Constants | 501 |
| 13.10 | Assertions | 502 |

14 Other C Topics 507

| | | |
|-------|---|-----|
| 14.1 | Introduction | 508 |
| 14.2 | Redirecting I/O | 508 |
| 14.3 | Variable-Length Argument Lists | 509 |
| 14.4 | Using Command-Line Arguments | 511 |
| 14.5 | Notes on Compiling Multiple-Source-File Programs | 512 |
| 14.6 | Program Termination with <code>exit</code> and <code>atexit</code> | 514 |
| 14.7 | <code>volatile</code> Type Qualifier | 515 |
| 14.8 | Suffixes for Integer and Floating-Point Constants | 516 |
| 14.9 | More on Files | 516 |
| 14.10 | Signal Handling | 518 |
| 14.11 | Dynamic Memory Allocation: Functions <code>calloc</code> and <code>realloc</code> | 520 |
| 14.12 | Unconditional Branching with <code>goto</code> | 521 |

15 C++ as a Better C; Introducing Object Technology 528

| | | |
|-------|---|-----|
| 15.1 | Introduction | 529 |
| 15.2 | C++ | 529 |
| 15.3 | A Simple Program: Adding Two Integers | 530 |
| 15.4 | C++ Standard Library | 532 |
| 15.5 | Header Files | 533 |
| 15.6 | Inline Functions | 535 |
| 15.7 | References and Reference Parameters | 537 |
| 15.8 | Empty Parameter Lists | 542 |
| 15.9 | Default Arguments | 542 |
| 15.10 | Unary Scope Resolution Operator | 544 |
| 15.11 | Function Overloading | 545 |
| 15.12 | Function Templates | 548 |
| 15.13 | Introduction to Object Technology and the UML | 551 |
| 15.14 | Wrap-Up | 554 |

16 Introduction to Classes and Objects 560

| | | |
|------|---|-----|
| 16.1 | Introduction | 561 |
| 16.2 | Classes, Objects, Member Functions and Data Members | 561 |

| | | |
|-------|---|-----|
| 16.3 | Defining a Class with a Member Function | 562 |
| 16.4 | Defining a Member Function with a Parameter | 566 |
| 16.5 | Data Members, <i>set</i> Functions and <i>get</i> Functions | 569 |
| 16.6 | Initializing Objects with Constructors | 576 |
| 16.7 | Placing a Class in a Separate File for Reusability | 579 |
| 16.8 | Separating Interface from Implementation | 583 |
| 16.9 | Validating Data with <i>set</i> Functions | 589 |
| 16.10 | Wrap-Up | 594 |

17 **Classes: A Deeper Look, Part 1** **601**

| | | |
|-------|---|-----|
| 17.1 | Introduction | 602 |
| 17.2 | Time Class Case Study | 603 |
| 17.3 | Class Scope and Accessing Class Members | 609 |
| 17.4 | Separating Interface from Implementation | 611 |
| 17.5 | Access Functions and Utility Functions | 612 |
| 17.6 | Time Class Case Study: Constructors with Default Arguments | 615 |
| 17.7 | Destructors | 620 |
| 17.8 | When Constructors and Destructors are Called | 621 |
| 17.9 | Time Class Case Study: A Subtle Trap—Returning a Reference to a private Data Member | 624 |
| 17.10 | Default Memberwise Assignment | 627 |
| 17.11 | Wrap-Up | 629 |

18 **Classes: A Deeper Look, Part 2** **635**

| | | |
|------|---|-----|
| 18.1 | Introduction | 636 |
| 18.2 | <code>const</code> (Constant) Objects and <code>const</code> Member Functions | 636 |
| 18.3 | Composition: Objects as Members of Classes | 645 |
| 18.4 | <code>friend</code> Functions and <code>friend</code> Classes | 651 |
| 18.5 | Using the <code>this</code> Pointer | 654 |
| 18.6 | <code>static</code> Class Members | 659 |
| 18.7 | Data Abstraction and Information Hiding | 664 |
| 18.8 | Wrap-Up | 666 |

19 **Operator Overloading** **672**

| | | |
|-------|--|-----|
| 19.1 | Introduction | 673 |
| 19.2 | Fundamentals of Operator Overloading | 674 |
| 19.3 | Restrictions on Operator Overloading | 675 |
| 19.4 | Operator Functions as Class Members vs. Global Function | 676 |
| 19.5 | Overloading Stream Insertion and Stream Extraction Operators | 678 |
| 19.6 | Overloading Unary Operators | 681 |
| 19.7 | Overloading Binary Operators | 682 |
| 19.8 | Dynamic Memory Management | 682 |
| 19.9 | Case Study: Array Class | 684 |
| 19.10 | Converting between Types | 696 |

| | | |
|-------|-------------------------------|-----|
| 19.11 | Building a String Class | 697 |
| 19.12 | Overloading ++ and -- | 698 |
| 19.13 | Case Study: A Date Class | 700 |
| 19.14 | Standard Library Class string | 704 |
| 19.15 | explicit Constructors | 708 |
| 19.16 | Proxy Classes | 711 |
| 19.17 | Wrap-Up | 715 |

20 Object-Oriented Programming: Inheritance 727

| | | |
|--------|--|-----|
| 20.1 | Introduction | 728 |
| 20.2 | Base Classes and Derived Classes | 729 |
| 20.3 | protected Members | 732 |
| 20.4 | Relationship between Base Classes and Derived Classes | 732 |
| 20.4.1 | Creating and Using a CommissionEmployee Class | 733 |
| 20.4.2 | Creating a BasePlusCommissionEmployee Class Without Using Inheritance | 738 |
| 20.4.3 | Creating a CommissionEmployee–BasePlusCommissionEmployee Inheritance Hierarchy | 743 |
| 20.4.4 | CommissionEmployee–BasePlusCommissionEmployee Inheritance Hierarchy Using protected Data | 748 |
| 20.4.5 | CommissionEmployee–BasePlusCommissionEmployee Inheritance Hierarchy Using private Data | 755 |
| 20.5 | Constructors and Destructors in Derived Classes | 762 |
| 20.6 | public, protected and private Inheritance | 770 |
| 20.7 | Software Engineering with Inheritance | 771 |
| 20.8 | Wrap-Up | 772 |

21 Object-Oriented Programming: Polymorphism 778

| | | |
|--------|--|-----|
| 21.1 | Introduction | 779 |
| 21.2 | Polymorphism Examples | 780 |
| 21.3 | Relationships Among Objects in an Inheritance Hierarchy | 781 |
| 21.3.1 | Invoking Base-Class Functions from Derived-Class Objects | 782 |
| 21.3.2 | Aiming Derived-Class Pointers at Base-Class Objects | 789 |
| 21.3.3 | Derived-Class Member-Function Calls via Base-Class Pointers | 790 |
| 21.3.4 | Virtual Functions | 792 |
| 21.3.5 | Summary of the Allowed Assignments Between Base-Class and Derived-Class Objects and Pointers | 798 |
| 21.4 | Type Fields and switch Statements | 799 |
| 21.5 | Abstract Classes and Pure virtual Functions | 799 |
| 21.6 | Case Study: Payroll System Using Polymorphism | 801 |
| 21.6.1 | Creating Abstract Base Class Employee | 803 |
| 21.6.2 | Creating Concrete Derived Class SalariedEmployee | 806 |
| 21.6.3 | Creating Concrete Derived Class HourlyEmployee | 808 |
| 21.6.4 | Creating Concrete Derived Class CommissionEmployee | 811 |

| | | |
|--------|--|-----|
| 21.6.5 | Creating Indirect Concrete Derived Class BasePlusCommissionEmployee | 813 |
| 21.6.6 | Demonstrating Polymorphic Processing | 814 |
| 21.7 | (Optional) Polymorphism, Virtual Functions and Dynamic Binding “Under the Hood” | 818 |
| 21.8 | Case Study: Payroll System Using Polymorphism and Runtime Type Information with Downcasting, <code>dynamic_cast</code> , <code>typeid</code> and <code>type_info</code> | 822 |
| 21.9 | Virtual Destructors | 826 |
| 21.10 | Wrap-Up | 826 |

22 **Templates** **832**

| | | |
|------|--|-----|
| 22.1 | Introduction | 833 |
| 22.2 | Function Templates | 833 |
| 22.3 | Overloading Function Templates | 837 |
| 22.4 | Class Templates | 837 |
| 22.5 | Nontype Parameters and Default Types for Class Templates | 844 |
| 22.6 | Notes on Templates and Inheritance | 845 |
| 22.7 | Notes on Templates and Friends | 845 |
| 22.8 | Notes on Templates and <code>static</code> Members | 846 |
| 22.9 | Wrap-Up | 846 |

23 **Stream Input/Output** **851**

| | | |
|--------|---|-----|
| 23.1 | Introduction | 852 |
| 23.2 | Streams | 853 |
| 23.2.1 | Classic Streams vs. Standard Streams | 853 |
| 23.2.2 | <code>iostream</code> Library Header Files | 854 |
| 23.2.3 | Stream Input/Output Classes and Objects | 854 |
| 23.3 | Stream Output | 857 |
| 23.3.1 | Output of <code>char *</code> Variables | 857 |
| 23.3.2 | Character Output Using Member Function <code>put</code> | 857 |
| 23.4 | Stream Input | 858 |
| 23.4.1 | <code>get</code> and <code>getline</code> Member Functions | 858 |
| 23.4.2 | <code>istream</code> Member Functions <code>peek</code> , <code>putback</code> and <code>ignore</code> | 861 |
| 23.4.3 | Type-Safe I/O | 861 |
| 23.5 | Unformatted I/O Using <code>read</code> , <code>write</code> and <code>gcount</code> | 861 |
| 23.6 | Introduction to Stream Manipulators | 862 |
| 23.6.1 | Integral Stream Base: <code>dec</code> , <code>oct</code> , <code>hex</code> and <code>setbase</code> | 863 |
| 23.6.2 | Floating-Point Precision (<code>precision</code> , <code>setprecision</code>) | 864 |
| 23.6.3 | Field Width (<code>width</code> , <code>setw</code>) | 865 |
| 23.6.4 | User-Defined Output Stream Manipulators | 866 |
| 23.7 | Stream Format States and Stream Manipulators | 868 |
| 23.7.1 | Trailing Zeros and Decimal Points (<code>showpoint</code>) | 868 |
| 23.7.2 | Justification (<code>left</code> , <code>right</code> and <code>internal</code>) | 869 |
| 23.7.3 | Padding (<code>fill</code> , <code>setfill</code>) | 871 |
| 23.7.4 | Integral Stream Base (<code>dec</code> , <code>oct</code> , <code>hex</code> , <code>showbase</code>) | 872 |

| | | |
|--------|--|-----|
| 23.7.5 | Floating-Point Numbers; Scientific and Fixed Notation (scientific, fixed) | 873 |
| 23.7.6 | Uppercase/Lowercase Control (uppercase) | 874 |
| 23.7.7 | Specifying Boolean Format (boolalpha) | 874 |
| 23.7.8 | Setting and Resetting the Format State via Member Function flags | 875 |
| 23.8 | Stream Error States | 877 |
| 23.9 | Tying an Output Stream to an Input Stream | 879 |
| 23.10 | Wrap-Up | 879 |

24 Exception Handling 889

| | | |
|-------|--|-----|
| 24.1 | Introduction | 890 |
| 24.2 | Exception-Handling Overview | 891 |
| 24.3 | Example: Handling an Attempt to Divide by Zero | 891 |
| 24.4 | When to Use Exception Handling | 897 |
| 24.5 | Rethrowing an Exception | 898 |
| 24.6 | Exception Specifications | 900 |
| 24.7 | Processing Unexpected Exceptions | 901 |
| 24.8 | Stack Unwinding | 901 |
| 24.9 | Constructors, Destructors and Exception Handling | 903 |
| 24.10 | Exceptions and Inheritance | 904 |
| 24.11 | Processing new Failures | 904 |
| 24.12 | Class auto_ptr and Dynamic Memory Allocation | 907 |
| 24.13 | Standard Library Exception Hierarchy | 909 |
| 24.14 | Other Error-Handling Techniques | 911 |
| 24.15 | Wrap-Up | 912 |

A Operator Precedence Charts 919

B ASCII Character Set 923

C Number Systems 924

| | | |
|-----|--|-----|
| C.1 | Introduction | 925 |
| C.2 | Abbreviating Binary Numbers as Octal and Hexadecimal Numbers | 928 |
| C.3 | Converting Octal and Hexadecimal Numbers to Binary Numbers | 929 |
| C.4 | Converting from Binary, Octal or Hexadecimal to Decimal | 929 |
| C.5 | Converting from Decimal to Binary, Octal or Hexadecimal | 930 |
| C.6 | Negative Binary Numbers: Two's Complement Notation | 932 |

D Game Programming: Solving Sudoku 937

| | | |
|-----|-------------------------------|-----|
| D.1 | Introduction | 937 |
| D.2 | Deitel Sudoku Resource Center | 938 |
| D.3 | Solution Strategies | 938 |

| | | |
|-----|-----------------------------------|-----|
| D.4 | Programming Sudoku Puzzle Solvers | 942 |
| D.5 | Generating New Sudoku Puzzles | 943 |
| D.6 | Conclusion | 945 |

Appendices on the Web **946**

Appendices E through I are PDF documents posted online at the book’s Companion Website (located at www.pearsonhighered.com/deitel).

E Game Programming with the Allegro C Library **I**

| | | |
|------|---|--------|
| E.1 | Introduction | II |
| E.2 | Installing Allegro | II |
| E.3 | A Simple Allegro Program | III |
| E.4 | Simple Graphics: Importing Bitmaps and Blitting | IV |
| E.5 | Animation with Double Buffering | IX |
| E.6 | Importing and Playing Sounds | XVI |
| E.7 | Keyboard Input | XX |
| E.8 | Fonts and Displaying Text | XXV |
| E.9 | Implementing the Game of Pong | XXXI |
| E.10 | Timers in Allegro | XXXVII |
| E.11 | The Grabber and Allegro Datafiles | XLII |
| E.12 | Other Allegro Capabilities | LI |
| E.13 | Allegro Resource Center | LII |

F Sorting: A Deeper Look **LVIII**

| | | |
|-----|----------------|-------|
| F.1 | Introduction | LIX |
| F.2 | Big O Notation | LIX |
| F.3 | Selection Sort | LX |
| F.4 | Insertion Sort | LXIV |
| F.5 | Merge Sort | LXVII |

G Introduction to C99 **LXXVIII**

| | | |
|------|--|----------|
| G.1 | Introduction | LXXIX |
| G.2 | Support for C99 | LXXIX |
| G.3 | New C99 Headers | LXXX |
| G.4 | // Comments | LXXX |
| G.5 | Mixing Declarations and Executable Code | LXXXI |
| G.6 | Declaring a Variable in a for Statement Header | LXXXII |
| G.7 | Designated Initializers and Compound Literals | LXXXIV |
| G.8 | Type bool | LXXXVII |
| G.9 | Implicit int in Function Declarations | LXXXVIII |
| G.10 | Complex Numbers | LXXXIX |
| G.11 | Variable-Length Arrays | XC |

| | | |
|------|---|-------|
| G.12 | The <code>sprintf</code> Function: Helping Avoid Hacker Attacks | XCIII |
| G.13 | Additions to the Preprocessor | XCIV |
| G.14 | Other C99 Features | XCVI |
| G.15 | Web Resources | XCIX |

H Using the Visual Studio Debugger CIV

| | | |
|-----|--|------|
| H.1 | Introduction | CV |
| H.2 | Breakpoints and the <code>Continue</code> Command | CV |
| H.3 | <code>Locals</code> and <code>Watch</code> Windows | CIX |
| H.4 | Controlling Execution Using the <code>Step Into</code> , <code>Step Over</code> , <code>Step Out</code> and <code>Continue</code> Commands | CXII |
| H.5 | <code>Autos</code> Window | CXIV |
| H.6 | Wrap-Up | CXVI |

I Using the GNU Debugger CXVIII

| | | |
|-----|--|---------|
| I.1 | Introduction | CXIX |
| I.2 | Breakpoints and the <code>run</code> , <code>stop</code> , <code>continue</code> and <code>print</code> Commands | CXIX |
| I.3 | <code>print</code> and <code>set</code> Commands | CXXIV |
| I.4 | Controlling Execution Using the <code>step</code> , <code>finish</code> and <code>next</code> Commands | CXXVI |
| I.5 | <code>watch</code> Command | CXXVIII |
| I.6 | Wrap-Up | CXXX |

Index 947

