Preface

1 Introduction to Computers and C++

2 Introduction to C++ Programming
## Contents

### 3 Introduction to Classes, Objects and Strings

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Introduction</td>
<td>65</td>
</tr>
<tr>
<td>3.2 Defining a Class with a Member Function</td>
<td>65</td>
</tr>
<tr>
<td>3.3 Defining a Member Function with a Parameter</td>
<td>68</td>
</tr>
<tr>
<td>3.4 Data Members, set Functions and get Functions</td>
<td>71</td>
</tr>
<tr>
<td>3.5 Initializing Objects with Constructors</td>
<td>77</td>
</tr>
<tr>
<td>3.6 Placing a Class in a Separate File for Reusability</td>
<td>81</td>
</tr>
<tr>
<td>3.7 Separating Interface from Implementation</td>
<td>84</td>
</tr>
<tr>
<td>3.8 Validating Data with set Functions</td>
<td>90</td>
</tr>
<tr>
<td>3.9 Wrap-Up</td>
<td>95</td>
</tr>
</tbody>
</table>

### 4 Control Statements: Part 1

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1 Introduction</td>
<td>102</td>
</tr>
<tr>
<td>4.2 Algorithms</td>
<td>102</td>
</tr>
<tr>
<td>4.3 Pseudocode</td>
<td>103</td>
</tr>
<tr>
<td>4.4 Control Structures</td>
<td>104</td>
</tr>
<tr>
<td>4.5 if Selection Statement</td>
<td>107</td>
</tr>
<tr>
<td>4.6 if…else Double-Selection Statement</td>
<td>108</td>
</tr>
<tr>
<td>4.7 while Repetition Statement</td>
<td>113</td>
</tr>
<tr>
<td>4.8 Formulating Algorithms: Counter-Controlled Repetition</td>
<td>114</td>
</tr>
<tr>
<td>4.9 Formulating Algorithms: Sentinel-Controlled Repetition</td>
<td>120</td>
</tr>
<tr>
<td>4.10 Formulating Algorithms: Nested Control Statements</td>
<td>130</td>
</tr>
<tr>
<td>4.11 Assignment Operators</td>
<td>134</td>
</tr>
<tr>
<td>4.12 Increment and Decrement Operators</td>
<td>135</td>
</tr>
<tr>
<td>4.13 Wrap-Up</td>
<td>138</td>
</tr>
</tbody>
</table>

### 5 Control Statements: Part 2

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1 Introduction</td>
<td>153</td>
</tr>
<tr>
<td>5.2 Essentials of Counter-Controlled Repetition</td>
<td>153</td>
</tr>
<tr>
<td>5.3 for Repetition Statement</td>
<td>155</td>
</tr>
<tr>
<td>5.4 Examples Using the for Statement</td>
<td>158</td>
</tr>
<tr>
<td>5.5 do…while Repetition Statement</td>
<td>162</td>
</tr>
<tr>
<td>5.6 switch Multiple-Selection Statement</td>
<td>164</td>
</tr>
<tr>
<td>5.7 break and continue Statements</td>
<td>173</td>
</tr>
<tr>
<td>5.8 Logical Operators</td>
<td>174</td>
</tr>
<tr>
<td>5.9 Confusing the Equality (==) and Assignment (=) Operators</td>
<td>179</td>
</tr>
<tr>
<td>5.10 Structured Programming Summary</td>
<td>180</td>
</tr>
<tr>
<td>5.11 Wrap-Up</td>
<td>185</td>
</tr>
</tbody>
</table>
10.2 const (Constant) Objects and const Member Functions 415
10.3 Composition: Objects as Members of Classes 423
10.4 friend Functions and friend Classes 429
10.5 Using the this Pointer 431
10.6 static Class Members 436
10.7 Proxy Classes 441
10.8 Wrap-Up 445

11 Operator Overloading: Class string 451
11.1 Introduction 452
11.2 Using the Overloaded Operators of Standard Library Class string 453
11.3 Fundamentals of Operator Overloading 456
11.4 Overloading Binary Operators 457
11.5 Overloading the Binary Stream Insertion and Stream Extraction Operators 458
11.6 Overloading Unary Operators 462
11.7 Overloading the Unary Prefix and Postfix ++ and -- Operators 463
11.8 Case Study: A Date Class 464
11.9 Dynamic Memory Management 469
11.10 Case Study: Array Class
11.10.1 Using the Array Class 472
11.10.2 Array Class Definition 475
11.11 Operators as Member Functions vs. Non-Member Functions 483
11.12 Converting between Types 483
11.13 explicit Constructors 485
11.14 Building a String Class 487
11.15 Wrap-Up 488

12 Object-Oriented Programming: Inheritance 499
12.1 Introduction 500
12.2 Base Classes and Derived Classes 500
12.3 protected Members 503
12.4 Relationship between Base Classes and Derived Classes 503
12.4.1 Creating and Using a CommissionEmployee Class 504
12.4.2 Creating a BasePlusCommissionEmployee Class Without Using Inheritance 508
12.4.3 Creating a CommissionEmployee–BasePlusCommissionEmployee Inheritance Hierarchy 514
12.4.4 CommissionEmployee–BasePlusCommissionEmployee Inheritance Hierarchy Using protected Data 519
## Contents

12.4.5 *CommissionEmployee–BasePlusCommissionEmployee*
  Inheritance Hierarchy Using *private* Data  

12.5 *Constructors and Destructors in Derived Classes*  
12.6 *public, protected and private Inheritance*  
12.7 *Software Engineering with Inheritance*  
12.8 *Wrap-Up*  

### 13 Object-Oriented Programming: Polymorphism  534

13.1 *Introduction*  
13.2 *Introduction to Polymorphism: Polymorphic Video Game*  
13.3 *Relationships Among Objects in an Inheritance Hierarchy*  
  13.3.1 *Invoking Base-Class Functions from Derived-Class Objects*  
  13.3.2 *Aiming Derived-Class Pointers at Base-Class Objects*  
  13.3.3 *Derived-Class Member-Function Calls via Base-Class Pointers*  
  13.3.4 *Virtual Functions*  
13.4 *Type Fields and *switch* Statements*  
13.5 *Abstract Classes and Pure *virtual* Functions*  
13.6 *Case Study: Payroll System Using Polymorphism*  
  13.6.1 *Creating Abstract Base Class Employee*  
  13.6.2 *Creating Concrete Derived Class SalariedEmployee*  
  13.6.3 *Creating Concrete Derived Class CommissionEmployee*  
  13.6.4 *Creating Indirect Concrete Derived Class BasePlusCommissionEmployee*  
  13.6.5 *Demonstrating Polymorphic Processing*  
13.7 *(Optional) Polymorphism, Virtual Functions and Dynamic Binding “Under the Hood”*  
13.8 *Case Study: Payroll System Using Polymorphism and Runtime Type Information with Downcasting, *dynamic_cast*, *typeid* and *type_info*  
13.9 *Virtual Destructors*  
13.10 *Wrap-Up*  

### 14 Templates  579

14.1 *Introduction*  
14.2 *Function Templates*  
14.3 *Overloading Function Templates*  
14.4 *Class Templates*
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.5</td>
<td>Nontype Parameters and Default Types for Class Templates</td>
<td>590</td>
</tr>
<tr>
<td>14.6</td>
<td>Wrap-Up</td>
<td>591</td>
</tr>
<tr>
<td>15</td>
<td>Stream Input/Output</td>
<td>595</td>
</tr>
<tr>
<td>15.1</td>
<td>Introduction</td>
<td>596</td>
</tr>
<tr>
<td>15.2</td>
<td>Streams</td>
<td>597</td>
</tr>
<tr>
<td>15.2.1</td>
<td>Classic Streams vs. Standard Streams</td>
<td>597</td>
</tr>
<tr>
<td>15.2.2</td>
<td>iostream Library Headers</td>
<td>598</td>
</tr>
<tr>
<td>15.2.3</td>
<td>Stream Input/Output Classes and Objects</td>
<td>598</td>
</tr>
<tr>
<td>15.3</td>
<td>Stream Output</td>
<td>601</td>
</tr>
<tr>
<td>15.3.1</td>
<td>Output of char * Variables</td>
<td>601</td>
</tr>
<tr>
<td>15.3.2</td>
<td>Character Output Using Member Function put</td>
<td>601</td>
</tr>
<tr>
<td>15.4</td>
<td>Stream Input</td>
<td>602</td>
</tr>
<tr>
<td>15.4.1</td>
<td>get and getline Member Functions</td>
<td>602</td>
</tr>
<tr>
<td>15.4.2</td>
<td>istream Member Functions peek, putback and ignore</td>
<td>605</td>
</tr>
<tr>
<td>15.4.3</td>
<td>Type-Safe I/O</td>
<td>605</td>
</tr>
<tr>
<td>15.5</td>
<td>Unformatted I/O Using read, write and gcount</td>
<td>605</td>
</tr>
<tr>
<td>15.6</td>
<td>Introduction to Stream Manipulators</td>
<td>606</td>
</tr>
<tr>
<td>15.6.1</td>
<td>Integral Stream Base: dec, oct, hex and setbase</td>
<td>607</td>
</tr>
<tr>
<td>15.6.2</td>
<td>Floating-Point Precision (precision, setprecision)</td>
<td>607</td>
</tr>
<tr>
<td>15.6.3</td>
<td>Field Width (width, setw)</td>
<td>609</td>
</tr>
<tr>
<td>15.6.4</td>
<td>User-Defined Output Stream Manipulators</td>
<td>610</td>
</tr>
<tr>
<td>15.7</td>
<td>Stream Format States and Stream Manipulators</td>
<td>612</td>
</tr>
<tr>
<td>15.7.1</td>
<td>Trailing Zeros and Decimal Points (showpoint)</td>
<td>612</td>
</tr>
<tr>
<td>15.7.2</td>
<td>Justification (left, right and internal)</td>
<td>613</td>
</tr>
<tr>
<td>15.7.3</td>
<td>Padding (fill, setfill)</td>
<td>615</td>
</tr>
<tr>
<td>15.7.4</td>
<td>Integral Stream Base (dec, oct, hex, showbase)</td>
<td>616</td>
</tr>
<tr>
<td>15.7.5</td>
<td>Floating-Point Numbers; Scientific and Fixed Notation (scientific, fixed)</td>
<td>617</td>
</tr>
<tr>
<td>15.7.6</td>
<td>Uppercase/Lowercase Control (uppercase)</td>
<td>618</td>
</tr>
<tr>
<td>15.7.7</td>
<td>Specifying Boolean Format (boolalpha)</td>
<td>618</td>
</tr>
<tr>
<td>15.7.8</td>
<td>Setting and Resetting the Format State via Member Function flags</td>
<td>619</td>
</tr>
<tr>
<td>15.8</td>
<td>Stream Error States</td>
<td>620</td>
</tr>
<tr>
<td>15.9</td>
<td>Tying an Output Stream to an Input Stream</td>
<td>622</td>
</tr>
<tr>
<td>15.10</td>
<td>Wrap-Up</td>
<td>623</td>
</tr>
<tr>
<td>16</td>
<td>Exception Handling: A Deeper Look</td>
<td>632</td>
</tr>
<tr>
<td>16.1</td>
<td>Introduction</td>
<td>633</td>
</tr>
</tbody>
</table>
16.2 Example: Handling an Attempt to Divide by Zero 633
16.3 When to Use Exception Handling 639
16.4 Rethrowing an Exception 640
16.5 Exception Specifications 641
16.6 Processing Unexpected Exceptions 642
16.7 Stack Unwinding 642
16.8 Constructors, Destructors and Exception Handling 644
16.9 Exceptions and Inheritance 645
16.10 Processing new Failures 645
16.11 Class unique_ptr and Dynamic Memory Allocation 648
16.12 Standard Library Exception Hierarchy 650
16.13 Wrap-Up 652

17 File Processing 658
17.1 Introduction 659
17.2 Files and Streams 659
17.3 Creating a Sequential File 660
17.4 Reading Data from a Sequential File 664
17.5 Updating Sequential Files 669
17.6 Random-Access Files 670
17.7 Creating a Random-Access File 671
17.8 Writing Data Randomly to a Random-Access File 675
17.9 Reading from a Random-Access File Sequentially 677
17.10 Case Study: A Transaction-Processing Program 679
17.11 Object Serialization 686
17.12 Wrap-Up 686

18 Class string and String Stream Processing 696
18.1 Introduction 697
18.2 string Assignment and Concatenation 698
18.3 Comparing strings 700
18.4 Substrings 703
18.5 Swapping strings 703
18.6 string Characteristics 704
18.7 Finding Substrings and Characters in a string 706
18.8 Replacing Characters in a string 708
18.9 Inserting Characters into a string 710
18.10 Conversion to C-Style Pointer-Based char * Strings 711
18.11 Iterators 713
19 Searching and Sorting  724
19.1 Introduction  725
19.2 Searching Algorithms  725
  19.2.1 Efficiency of Linear Search  726
  19.2.2 Binary Search  727
19.3 Sorting Algorithms  732
  19.3.1 Efficiency of Selection Sort  733
  19.3.2 Efficiency of Insertion Sort  733
  19.3.3 Merge Sort (A Recursive Implementation)  733
19.4 Wrap-Up  740

20 Custom Templatized Data Structures  746
20.1 Introduction  747
20.2 Self-Referential Classes  748
20.3 Dynamic Memory Allocation and Data Structures  749
20.4 Linked Lists  749
20.5 Stacks  764
20.6 Queues  768
20.7 Trees  772
20.8 Wrap-Up  780

21 Bits, Characters, C Strings and structs  791
21.1 Introduction  792
21.2 Structure Definitions  792
21.3 typedef  794
21.4 Example: Card Shuffling and Dealing Simulation  794
21.5 Bitwise Operators  797
21.6 Bit Fields  806
21.7 Character-Handling Library  810
21.8 Pointer-Based String Manipulation Functions  815
21.9 Pointer-Based String-Conversion Functions  822
21.10 Search Functions of the Pointer-Based String-Handling Library  827
21.11 Memory Functions of the Pointer-Based String-Handling Library  831
21.12 Wrap-Up  835
22 Standard Template Library (STL)  850
  22.1 Introduction to the Standard Template Library (STL)  851
  22.2 Introduction to Containers  853
  22.3 Introduction to Iterators  856
  22.4 Introduction to Algorithms  861
  22.5 Sequence Containers  863
    22.5.1 vector Sequence Container  864
    22.5.2 list Sequence Container  871
    22.5.3 deque Sequence Container  875
  22.6 Associative Containers  877
    22.6.1 multiset Associative Container  877
    22.6.2 set Associative Container  880
    22.6.3 multimap Associative Container  881
    22.6.4 map Associative Container  883
  22.7 Container Adapters  885
    22.7.1 stack Adapter  885
    22.7.2 queue Adapter  887
    22.7.3 priority_queue Adapter  888
  22.8 Algorithms  890
    22.8.1 fill, fill_n, generate and generate_n  890
    22.8.2 equal, mismatch and lexicographical_compare  892
    22.8.3 remove, remove_if, remove_copy and remove_copy_if  895
    22.8.4 replace, replace_if, replace_copy and replace_copy_if  897
    22.8.5 Mathematical Algorithms  900
    22.8.6 Basic Searching and Sorting Algorithms  903
    22.8.7 swap, iter_swap and swap_ranges  905
    22.8.8 copy_backward, merge, unique and reverse  906
    22.8.9 inplace_merge, unique_copy and reverse_copy  909
    22.8.10 Set Operations  910
    22.8.11 lower_bound, upper_bound and equal_range  913
    22.8.12 Heapsort  915
    22.8.13 min and max  918
    22.8.14 STL Algorithms Not Covered in This Chapter  919
  22.9 Class bitset  920
  22.10 Function Objects  924
  22.11 Wrap-Up  927

23 Boost Libraries, Technical Report 1 and C++0x  936
  23.1 Introduction  937
## Contents

23.2 Deitel Online C++ and Related Resource Centers 937  
23.3 Boost Libraries 937  
23.4 Boost Libraries Overview 938  
23.5 Regular Expressions with the `regex` Library 941  
  23.5.1 Regular Expression Example 942  
  23.5.2 Validating User Input with Regular Expressions 944  
  23.5.3 Replacing and Splitting Strings 947  
23.6 Smart Pointers 950  
  23.6.1 Reference Counted `shared_ptr` 950  
  23.6.2 `weak_ptr`: `shared_ptr` Observer 954  
23.7 Technical Report 1 960  
23.8 C++0x 961  
23.9 Core Language Changes 962  
23.10 Wrap-Up 967  

24 Other Topics 974  
24.1 Introduction 975  
24.2 `const_cast` Operator 975  
24.3 `mutable` Class Members 977  
24.4 `namespace`s 979  
24.5 Operator Keywords 982  
24.6 Pointers to Class Members (`.*` and `->*`) 984  
24.7 Multiple Inheritance 986  
24.8 Multiple Inheritance and `virtual` Base Classes 991  
24.9 Wrap-Up 996  

Chapters on the Web 1001  

A Operator Precedence and Associativity 1002  

B ASCII Character Set 1004  

C Fundamental Types 1005  

D Number Systems 1007  
  D.1 Introduction 1008
26 ATM Case Study, Part 2: Implementing an Object-Oriented Design 26-1

26.1 Introduction 26-2
26.2 Starting to Program the Classes of the ATM System 26-2
26.3 Incorporating Inheritance into the ATM System 26-8
26.4 ATM Case Study Implementation 26-15
   26.4.1 Class ATM 26-16
   26.4.2 Class Screen 26-23
   26.4.3 Class Keypad 26-25
   26.4.4 Class CashDispenser 26-26
   26.4.5 Class DepositSlot 26-28
   26.4.6 Class Account 26-29
   26.4.7 Class BankDatabase 26-31
   26.4.8 Class Transaction 26-35
   26.4.9 Class BalanceInquiry 26-37
   26.4.10 Class Withdrawal 26-39
   26.4.11 Class Deposit 26-44
   26.4.12 Test Program ATMCaseStudy.cpp 26-47
26.5 Wrap-Up 26-47

F C Legacy Code Topics F-1

F.1 Introduction F-2
F.2 Redirecting Input/Output on UNIX/Linux/Mac OS X and Windows Systems F-2
F.3 Variable-Length Argument Lists F-3
F.4 Using Command-Line Arguments F-5
F.5 Notes on Compiling Multiple-Source-File Programs F-7
F.6 Program Termination with exit and atexit F-9
F.7 Type Qualifier volatile F-10
F.8 Suffixes for Integer and Floating-Point Constants F-10
F.9 Signal Handling F-11
F.10 Dynamic Memory Allocation with calloc and realloc F-13
F.11 Unconditional Branch: goto F-14
F.12 Unions F-15
F.13 Linkage Specifications F-18
F.14 Wrap-Up F-19

G UML 2: Additional Diagram Types G-1

G.1 Introduction G-1
G.2 Additional Diagram Types G-2
## Contents

### H Using the Visual Studio Debugger
- **H.1** Introduction  
- **H.2** Breakpoints and the *Continue* Command  
- **H.3** *Locals* and *Watch* Windows  
- **H.4** Controlling Execution Using the *Step Into*, *Step Over*, *Step Out* and *Continue* Commands  
- **H.5** *Autos* Window  
- **H.6** Wrap-Up

### I Using the GNU C++ Debugger
- **I.1** Introduction  
- **I.2** Breakpoints and the *run*, *stop*, *continue* and *print* Commands  
- **I.3** *print* and *set* Commands  
- **I.4** Controlling Execution Using the *step*, *finish* and *next* Commands  
- **I.5** *watch* Command  
- **I.6** Wrap-Up