CD-ROM INCLUDED

Learning to Program in C++

Steve Heller

Contents

Foreword xxxi

Preface xxxiii

Acknowledgements xxxvii

Letter from a Novice xxxix

CHAPTER 1 *Introduction to Programming* *1*

Definitions **3**
How to Write a Program 5
Baby Steps **7**
On with the Show 9

CHAPTER 2 *Hardware Fundamentals* *11*

Definitions **12**
Objectives of This Chapter **13**
Behind the Curtain **14**
Disk 14

1985: A Space Odyssey 17
RAM 18
Return to Sender, Address Unknown 19
The CPU 21
RAM vs. CPU speeds. 23
A memory hierarchy. 26
Caching In 27
Please Register Here 27
Odometer Trouble **30**
The first few numbers. 31
The next few numbers. 32
How many combinations? 34
Back to the Future 35
Over-Hexed 39
Binary to hex conversion table 44
Different representations of the same numbers 45
Exercises **46**
Registering Relief **47**
32 and 16 bit registers, before add ax,1 *49*
32 and 16 bit registers, after add ax,1 *49*
32 and 16 bit registers, after add eax,1 *50*
On a RAMpage 50
Registering Bewilderment 51
32- and 16-bit register codes 52
Slimming the Program 53
A Fetching Tale 60
Instruction execution time, using registers and prefetching 61
Review **61**
Conclusion **63**
Answers to Exercises **63**

CHAPTER 3 *Basics of Programming* ***65***

Definitions **66**
Objectives of This Chapter **66**
Speed Demon **67**
Blaming It on the Computer 67
That Does Not Compute 68
Lost in Translation **69**
A little numeric calculation 70

What's Going on Underneath? **74**
A small section of RAM 75
Who's on First? **78**
One little endian 79
A big endian example 79
Exercises, First Set **79**
Exercise 1 80
Underware? 80
A really little numeric calculation 80
Compiling, part 3 84
Execution Is Everything 90
A Cast of Characters **93**
Some real characters *and* strings *(code\basic00.cc) 95*
Yet another small section of RAM 100
A Byte by Any Other Name... 101
Some Strings Attached 102
Special characters for program text 103
Exercises, Second Set **103**
A small section of RAM 103
In and Out **104**
Some simple output (code\basic01.cc) 104
Some simple input and output (code\basic02.cc) 105
If Only You Knew **106**
Using an if *statement (code\basic03.cc) 107*
While We're on the Subject **108**
Using a while *statement (code\basic04.cc) 109*
Exercises, Third Set **112**
Just up the Block **113**
At the Fair **114**
A C++ Program (code\pump1.cc) 116
Novice Alert 122
Take It for a Spin 124
Exercises, Fourth Set **125**
Review **125**
Conclusion **129**
Answers to Exercises **129**
First dinner party program (code\basic05.cc) 130
Second dinner party program (code\basic06.cc) 131
else if *example 132*
Name and age program (code\basic07.cc) 133
Novice program (code\basic08.cc) 134

Allowance program (code\basic09.cc) 136

CHAPTER 4 *More Basics* **143**

Objectives of This Chapter **143**
Algorithmic Thinking **144**
Finding the top two weights, first try (code\pump1a.cc) 144
A Prize Catch 148
Susan's solution to the bug in the first attempt 149
Using an if *statement with an* else *clause 149*
Finding the top two weights (code\pump2.cc) 150
What a Tangled Web We Weave... 153
You May Already Have Won **154**
Variables, by the Numbers 156
Using a vector *(code\vect1.cc) 158*
Zero Isn't Nothing 161
Index Variables **163**
Using a for *statement (from code\vect1.cc) 164*
Sorting the weights (from code\vect1.cc) 173
Elements vs. values 175
Details, Details 183
Initial situation 184
After the first pass 184
After the second pass 185
Final situation 185
To Err Is Human... 185
A possible error message 187
Sorting the weights, again (from code\vect1.cc) 187
Sorting the weights, with correct initialization (from code\vect2.cc) 192
To Really Foul Things Up Requires a Computer **193**
What, Me Worry? 195
Garbage in, Garbage Out 197
Garbage prevention, first attempt (from code\vect2a.cc) 197
Finding the top three weights using vectors *(code\vect3.cc) 199*
Review **201**
Exercises **203**
Exercise 1 (code\morbas00.cc) 204

Exercise 2 (code\morbas01.cc) 204
Conclusion **205**
Answers to Exercises **206**
Weight requesting program, first try (code\morbas02.cc) 207
Error messages from the erroneous weight program (code\morbas02.cc) 210
The corrected weight program (code\morbas03.cc) 210
The weight totalling program (code\morbas04.cc) 211

CHAPTER 5 *Functional Literacy* ***215***

Definitions **218**
Objectives of This Chapter **219**
Functioning Normally **219**
A sample program with duplicated code (code\nofunc.cc) 220
A function call 222
Above Average 225
A function to average two values 225
Return to Sender 227
For the Sake of Argument 230
Argument passing with one argument (code\birthday.cc) 230
General Delivery 234
Using a Function **235**
Using the Average function (code\func1.cc) 235
A Convincing Argument 243
The Man behind the Curtain **243**
The Object of My Affections 245
Making an executable 246
Operating Systematically 247
Using Your Library Card 248
Automatic Pilot **250**
Stacking the Deck 254
A stack with one entry 255
A stack with two entries 255
A stack with three entries 255
Don't Call Me, I'll Call You 256
How It All Stacks Up 258
An empty stack 258

The stack immediately after the call to Average *259*
The stack after auto variable allocation 261
Scoped Out **263**
Scope vs. storage class 263
Static Cling 264
Using an auto variable and initializing it (code\count1.cc) 266
Using an auto variable and not initializing it (code\count2.cc) 267
Using a local static variable and initializing it (code\count3.cc) 268
Using a local static variable and not initializing it (code\count4.cc) 268
Using a global variable and not initializing it (code\count6.cc) 269
Using a global variable and initializing it (code\count5.cc) 273
Using variables of different scopes and storage classes (code\scopclas.cc) 275
The results of using variables of different scopes and storage classes (code\scopclas.out) 277
Exercises, First Set **278**
Exercise 1a (code\inita.cc) 278
Exercise 1b (code\initb.cc) 279
Exercise 1c (code\initc.cc) 280
Exercise 1d (code\initd.cc) 281
Exercise 1e (code\inite.cc) 282
Exercise 1f (code\initf.cc) 283
Think Globally? **285**
A BASIC Difficulty 285
I Say "Live It, or Live With It" 287
Nesting Instinct **287**
The stack after the initialization of Result *288*
The stack after exiting from Average *288*
Review **290**
Exercises, Second Set **292**
Exercise 2 (code\calc1.cc) 292
Conclusion **293**
Answers to Exercises **294**
The stack immediately before the call to mess *296*
Immediately before executing the first instruction in mess *297*

The stack after mess *has declared the* auto *variable* xyz *297*
The stack after xyz *has been initialized 298*
The stack before counter *is called 298*
Immediately upon entry to counter *299*
Before execution of the first instruction in counter *299*

CHAPTER 6 *Taking Inventory* ***303***

Definitions **305**
Objectives of This Chapter **306**
Pay Some Attention to the Man Behind the Curtain **306**
Taking Stock **313**
The initial sample program for the StockItem class *(code\itemtst1.cc) 314*
More Definitions **315**
Common Behavior **316**
Comparison between native and user-defined types 317
The initial interface of the StockItem class *(code\item1.h) 323*
The default constructor for the StockItem class *(from code\item1.cc) 328*
Another way to write the default StockItem *constructor 331*
Go to the Head of the class **333**
Shop till You Drop **339**
Another constructor for the StockItem class *(from code\item1.cc) 341*
Display *member function for the* StockItem class *(from code\item1.cc) 343*
The initial interface of the StockItem class *(code\item1.h) 346*
The initial implementation of the StockItem class *(code\item1.cc) 347*
Vectoring In 350
Reading and displaying a vector *of* StockItem*s (code\itemtst2.cc) 351*
The Read *function for the* StockItem class *(from code\item2.cc) 353*

The second version of the interface for the StockItem class *(code\item2.h) 354*
References Required **354**
Don't Fence Me In 360
Can I Help You? 364
First attempt to update inventory of StockItem*s (code\itemtst3.cc) 364*
Access Denied 367
An enhanced interface for the StockItem class *(code\item4.h) 370*
The Customer Is Always Right 371
Some new functions for the StockItem class *(from code\item4.cc) 372*
Updating StockItem *inventory (code\itemtst4.cc) 373*
Next Customer, Please? 375
Interface of Inventory class *(code\invent1.h) 376*
Default constructor for Inventory class *(from code\invent1.cc) 379*
LoadInventory *function for* Inventory class *(from code\invent1.cc) 380*
Nothing Ventured, Nothing Gained 381
The implementation of IsNull *(from code\item5.cc) 382*
FindItem *function for* Inventory class *(from code\invent1.cc) 383*
UpdateItem *function for* Inventory class *(from code\invent1.cc) 384*
The implementation of GetUPC *(from code\item5.cc) 385*
The implementation of GetPrice *(from code\item5.cc) 385*
Current interface for Inventory class *(code\invent1.h) 386*
Current implementation for Inventory class *(code\invent1.cc) 386*
Current interface for StockItem class *(code\item5.h) 388*
Current implementation for StockItem class *(code\item5.cc) 389*
Updated inventory application (code\itemtst5.cc) 391
Paging Rosie Scenario 395
Review **396**
Exercises **405**

Conclusion **406**
Answers to Exercises **406**
The Write *member function for the* StockItem class *(from code\item6.cc) 407*
The StoreInventory *member function for the* Inventory class *(from code\invent2.cc) 407*
The changes to the application program (from code\itemtst6.cc) 407

CHAPTER 7 *Stringing Along* *409*

Objectives of This Chapter **409**
Playing out the string **410**
The string class *interface, initial version (code\string1.h) 413*
The initial implementation for the string class *(code\string1.cc) 414*
The default constructor for the string class *(from code\string1.cc) 415*
Passing Along a Few Pointers 416
The Dynamic Duo, new and delete **419**
An empty string *in memory 425*
Our first test program for the string class *(code\strtst1.cc) 425*
Caution: Construction Area **426**
The char* *constructor for the* string class *(from code\string1.cc) 430*
string n *during construction 437*
A simple test program for the string class *(code\strtst1.cc) 438*
Constructive Criticism? **439**
The char* *constructor for the* string class, *again (from code\string1.cc) 439*
Tricky Assignment **440**
string *n in memory 442*
string*s* n *and* s *in memory after compiler-generated = 442*
Assignment of Responsibility **444**
string*s* n *and* s *in memory after custom = 445*
References Required **448**
The declaration of operator = *for the* string class *451*
Hello, operator? **451**

Calling the operator = *implementation 453*
What Is the Meaning of this*? 453*
*The assignment operator (*operator =*) for the* string class *(from code\string1.cc) 454*
Equality Now! **454**
Please delete *Me, Let Me Go 457*
Running on empty 460
The Next Assignment 461
*A hypothetical assignment operator (*operator =*) for the* string class *with explicit* this *466*
The Terminator 467
The destructor for the string class *(from code/string1.cc) 470*
Review **470**
Exercises **472**
Exercise 1 (code\strex1.cc) 472
Exercise 2 (code\strex2.cc) 473
Exercise 3 (code\strex3.cc) 473
Conclusion **474**
Answers to Exercises 474

CHAPTER 8 *Down the Garden Path* ***479***

Objectives of This Chapter **479**
For Reference Only 480
Call by value ("normal argument") using the compiler-generated copy constructor 480
Unfair Copy 481
Call by reference 482
Our first test program for the string class *(code\strtst1.cc) 483*
Temporary Help Wanted 484
Assigning a C string to a string *via* string::string(char*) *485*
Copy Cat **489**
The string class *interface (code\string1.h) 489*
The copy constructor for the string class *490*
Screen Test 491
The string class *interface, with* Display *function (code\string3.h) 492*

The string class *test program, using the* Display *function (code\strtst3.cc) 493*
The string class *implementation of the* Display *function 494*
A Character Study 494
Array of Hope? 496
Dangerous characters *(code\dangchar.cc) 497*
A Slippery Character 498
Overwrought 499
Reaping the whirlwind 500
private Property: Keep Out! **503**
Attempted privacy violation (code\strtst3a.cc) 504
Maintenance Required 505
Yet another version of the string class *interface (code\string4.h) 507*
The string class *implementation of the* GetLength *function (from code\string4.cc) 507*
Using the GetLength *function in the* string class *(code\strtst4.cc) 508*
First Review **509**
A String of Wins **512**
Sorting a vector *of* string*s (code\strsort1.cc) 513*
The updated string class *interface, including comparison and I/O operators (code\string5.h) 516*
Less Than Obvious 517
Down for the Count 519
string*s* x *and* y *in memory 520*
string*s* x *and* y *in memory, with an embedded null byte 521*
Using operator < *for* string*s (code\strtst5x.cc) 524*
The implementation of operator < *for* string*s (from code\string5a.cc) 525*
For Better or Worse? 526
Is our character less than the other one? (from code\string5a.cc) 528
The else *clause in the comparison loop (from code\string5a.cc) 529*
Handling the return value (from code\string5a.cc) 530
A Greater Cause 530
Simple Pleasures 531
Implementing operator < *for* string*s (from code\string5.cc) 531*

Equalization of Opportunity 534
Implementing operator == *for* string*s (from code\string5.cc) 535*
Displaying Expertise 535
Down by the Old cout *Stream 536*
Chaining several operator << *expressions together (code\cout1.cc) 537*
Gently Down the Stream 537
Friends of Global Progress 539
An operator << *function to output a* string *(from code\string5.cc) 540*
Members and Friends Only 541
Why we need a global function for operator << *542*
Reader and Advisor 543
An operator >> *function to input a* string *(from code\string 5.cc) 543*
Initial Here **545**
Error from an uninitialized const *(code\string5x.out) 545*
*Use of a non-*const *array size (code\string5y.cc) 546*
*Trying to compile a program with a non-*const *array size (code\string5y.out) 547*
Pointers and Setters 548
Second Review **552**
Exercises **554**
Exercise 1 (code\strex5.cc) 555
Exercise 2 (code\strex6.cc) 555
Conclusion **556**
Answers to Exercises **557**
The string class *interface file (from code\string6.h) 558*
The string class *implementation of* operator > *559*
The string class *implementation of* operator >= *559*
The string class *implementation of* operator != *(from code\string6.cc) 560*
The string class *implementation of* operator <= *(from code\string6.cc) 561*
The test program for the comparison operators of the string class *(code\strcmp.cc) 561*

A simple stream *example (code\stream1.cc) 622*
An empty ostream *object 622*
An ostream *object with some data 623*
An ostream *object with some more data 623*
An empty ostream *object 624*
We All stream for strstream **624**
A strstream *formatting example (code\stream2.cc) 624*
An empty strstream *object 625*
A strstream *object with some contents 628*
A strstream *object with some more contents 628*
A strstream *object with even more contents 629*
A strstream *object after reading its contents 629*
Use It or Lose It 630
Default formatting example (code\coutdef1.cc) 630
Output of default formatting example (code\coutdef1.out) 631
Output of controlled formatting example (code\coutdef2.out) 632
Manipulative Behavior 633
Controlled formatting example (code\coutdef2.cc) 633
Baseless Accusations? 635
Default constructor for DatedStockItem *(from code\item21.cc) 635*
Specifying the base class *constructor for a derived* class *object 637*
Constructing a default DatedStockItem *object 639*
Normal constructor for DatedStockItem *(from code\item21.cc) 639*
Constructing a DatedStockItem *object 641*
Reordering Priorities 642
Reorder *function for* DatedStockItem *(from code\item21.cc) 642*
Calling Reorder *through a* StockItem *pointer, part 1 645*
Calling Reorder *through a* DatedStockItem *pointer 646*
Calling Reorder *through a* StockItem *pointer, part 2 647*
Function call example (code\nvirtual.cc) 647
Function call example output (code\nvirtual.out) 648
Simplified implementation for StockItem *and* DatedStockItem class*es (code\itema.cc) 649*

Review **651**
Exercises **654**
Conclusion **656**

CHAPTER 10 *Pretty Poly* *657*

Definitions **658**
Objectives of This Chapter **659**
Polymorphism **659**
- virtual *Certainty 662*
- *Dangerous polymorphism: Interfaces of* StockItem *and* DatedStockItem *with* virtual Reorder *function (code\itemb.h) 663*
- virtual *function call example output (code\virtual.out) 664*
- *A simplified* StockItem *object without* virtual *functions 665*
- *Dangerous polymorphism: A simplified* StockItem *object with a* virtual *function 667*
- *Dangerous polymorphism:* **A simplified** DatedStockItem **object with a** virtual **function** *668*
- *Dangerous polymorphism: Calling a* virtual Reorder *function through a* StockItem *pointer to a* StockItem *object 670*
- *Dangerous polymorphism: Calling a* virtual Reorder *function through a* DatedStockItem *pointer to a* DatedStockItem *object 671*
- *Dangerous polymorphism: Calling a* virtual Reorder *function through a* StockItem *pointer to a* DatedStockItem *object 672*
- *Dangerous polymorphism: A simplified* StockItem *object with two* virtual *functions 673*
- *Dangerous polymorphism: A simplified* DatedStockItem *with two* virtual *functions 674*
- *A Pointed Reminder 675*
- *Dangerous polymorphism: Using* operator << *with a* StockItem* *(code\polyioa.cc) 676*
- *Result of using* operator << *with a* StockItem* *(code\polyioa.out) 676*
- *Dangerous polymorphism:* StockItem *interface with* operator << *and* operator >> *(code\itemc.h) 677*

Dangerous polymorphism: StockItem *implementation with* operator << *and* operator >> *(code\itemc.cc) 680*

Dangerous polymorphism: The implementation of operator << *with a* StockItem* *(from code\itemc.cc) 686*

The Old Switcheroo 686

Dangerous polymorphism: StockItem::Write *(from code\itemc.cc) 687*

Dangerous polymorphism: DatedStockItem::Write *(from code\itemc.cc) 687*

It's Not Polite to Point 688

Dangerous polymorphism: Using operator >> *and* operator << *with a* StockItem* *(code\polyiob.cc) 689*

Dangerous polymorphism: **The results of using** operator >> **and** operator << **with a** StockItem* **(code\polyiob.out)** *690*

Dangerous polymorphism: The implementation of operator >> *(from code\itemc.cc) 691*

Exercises, First Set **692**

Pretty Polly Morphic **693**

More Definitions **693**

Paging Miss Management **694**

Dangerous polymorphism: Using operator >> *and* operator << *with a* StockItem* *(code\polyiob.cc) 696*

Safe polymorphism: Using operator >> *and* operator << *with a polymorphic* StockItem *(code\polyioc.cc) 697*

Safe polymorphism: The polymorphic object version of the StockItem *interface (code\itemp.h) 698*

We'll Manage Somehow 699

Safe polymorphism: The UndatedStockItem *and* DatedStockItem *interfaces for the polymorphic version of* StockItem *(code\itempi.h) 700*

Safe polymorphism: The implementation of the UndatedStockItem *and* DatedStockItem class*es (code\itemp.cc) 702*

Safe polymorphism: The implementation of operator << *for a polymorphic* StockItem *(from code\itemp.cc) 710*

Safe polymorphism: A polymorphic StockItem *object with no date 712*

Safe polymorphism: A polymorphic StockItem *object with a date 714*

A simplified version of the structure of a DatedStockItem *object 716*

Setting the Standard **716**

Safe polymorphism: The default constructor for the polymorphic StockItem class *(from code\itemp.cc) 717*

Safe polymorphism: A default-constructed polymorphic StockItem *object 718*

Safe polymorphism: The default constructor for the UndatedStockItem class *(from code\itemp.cc) 719*

Base Instincts **720**

Safe polymorphism: Implementing a special protected *constructor for* StockItem *(from code\itemp.cc) 721*

References Count **723**

Starring Sharon Sharalike **725**

Safe polymorphism: An example program for reference-counting with StockItem*s (code\refcnt1.cc) 725*

Safe polymorphism: A normal constructor to create a StockItem *without a date (from code\itemp.cc) 726*

Safe polymorphism: A polymorphic StockItem *object with an* UndatedStockItem *worker 727*

Safe polymorphism: A normal constructor that constructs a StockItem *having a date (from code\itemp.cc) 728*

Safe polymorphism: A polymorphic StockItem *object with a* DatedStockItem *worker 728*

Safe polymorphism: The copy constructor for StockItem *(from code\itemp.cc) 729*

Safe polymorphism: Two polymorphic StockItem *objects sharing the same* UndatedStockItem *worker object 730*

Safe polymorphism: The assignment operator (operator =) *for* StockItem *(from code\itemp.cc) 731*

Safe polymorphism: Two polymorphic StockItem *objects sharing the same* DatedStockItem *worker object 733*

Safe polymorphism: A polymorphic StockItem *object* 734
The Last Shall Be First 735
Safe polymorphism: The destructor for the StockItem class *(from code\itemp.cc)* 735
Going, Going, Gone 740
Safe polymorphism: The destructor for the StockItem class *(from code\itemp.cc)* 740
For the Benefit of Posterity **743**
Review **744**
Exercises, Second Set **750**
Conclusion **751**

CHAPTER 11 *The Home Inventory Project* *753*

Definitions **753**
Objectives of This Chapter **754**
Homing In **755**
What It Is, Mama! 756
Interface R Us **758**
The initial interface for the HomeItem *manager* class *(code/hmit1.h)* 758
Deja Vu All Over Again 759
The initial interface for the HomeItemBasic *and* HomeItemMusic *worker* class*es (code\hmiti1.h)* 761
What They Don't Know Won't Hurt Them **764**
The initial test program for the HomeItem class*es (code\hmtst1.cc)* 765
Results of running the first HomeItem *test program (code\hmit1.out)* 766
Initial implementation of HomeItem *manager and worker* class*es (code\hmit1.cc)* 767
HomeItem::Write *(from code\hmit1.cc)* 773
The HomeItem *implementation of* operator >> *(from code\hmit1.cc)* 774
The (incorrect) while *loop in the original implementation of* operator >> 777
Making All Local Stops 778
What Have i *Started?* 779
A legal program (code\fortest.cc) 780
Stereo Typing 781

An incorrect default constructor for the HomeItemBasic class *781*
HomeItemBasic::GetType *(from code\hmit1.cc) 782*
HomeItemMusic::GetType *(from code\hmit1.cc) 782*
HomeItemBasic::Write *(from code\hmit1.cc) 782*
Virtual Reality 783
HomeItemMusic::Write *(from code\hmit1.cc) 783*
The initial HomeInventory class *interface (code\hmin2.h) 787*
The initial implementation of HomeInventory *(code\hmin2.cc) 787*
Waste Not, Want Not 791
You Can Get What You Need 792
Another possible implementation of LoadInventory *(from code\hmin2a.cc) 792*
A Smith and Wesson Beats Four Aces **794**
Yet another implementation of LoadInventory *(from code\hmin3.cc) 795*
Everything Is More Complicated Than It Looks 796
Back to the Future **797**
The next interface for the HomeInventory class *(code\hmin4.h) 798*
The AddItem *member function of* HomeInventory *(from code\hmin4.cc) 799*
The new interface for HomeItem *(code\hmit4.h) 800*
The implementation of HomeItem::NewItem() *(from code\hmit4.cc) 802*
The new version of operator >> *(from code\hmit4.cc) 802*
Strong Like bool *805*
HomeItemBasic::FormattedDisplay *(from code\hmit4.cc) 807*
HomeItemMusic::FormattedDisplay *(from code\hmit4.cc) 808*
Leaving Well Enough Alone 809
The test program for adding a HomeItem *interactively (hmtst4.cc) 809*
The next version of the interface for HomeInventory *(code\hmin5.h) 810*
The next version of the HomeInventory *test program (code\hmtst5.cc) 811*

The EditItem *function of* HomeInventory *(from code\hmin5.cc) 813*
'Tis a Gift to Be Simple **814**
The latest version of the HomeItem class *interface (code\hmit5.h) 815*
HomeItem::Edit *(from code\hmit5.cc) 816*
HomeItemBasic::CopyData() *818*
Better Read Than Dead 819
The latest version of operator >> *(from code\hmit5.cc) 819*
For Your Eyes Only 821
The latest version of the interface for the HomeItem *worker* class*es (code\hmiti5.h) 822*
U Pluribus enum *825*
HomeItemBasic::GetFieldName *(from code\hmit5.cc) 827*
Taking a short*cut 829*
HomeItem::Read *(from code\hmit5.cc) 829*
HomeItemBasic::Read *(from code\hmit5.cc) 830*
HomeItemBasic::ReadInteractive *(from code\hmit5.cc) 831*
this *Must Be the Place 834*
HomeItemBasic::ReadFromFile *(from code\hmit5.cc) 836*
HomeItemBasic::Edit *(from code\hmit5.cc) 836*
HomeItemBasic::FormattedDisplay *(from code\hmit5.cc) 837*
HomeItemBasic::EditField *(from code\hmit5.cc) 839*
Facing the Music **841**
HomeItemMusic::FormattedDisplay *(from code\hmit5.cc) 842*
Maintaining Our Position 843
HomeItemMusic::ReadInteractive *(from code\hmit5.cc) 844*
HomeItemMusic::ReadFromFile *(from code\hmit5.cc) 845*
HomeItemMusic::EditField *(from code\hmit5.cc) 846*
Review **849**
Exercises **856**
Conclusion **857**

CHAPTER 12 *Homeward Bound 859*

Definitions **860**

Objectives of This Chapter **861**

Super-string Theory **861**

The new string class *interface (code\string7.h) 862*

Batteries Not #include*d 863*

Construction Ahead 866

Default Is Mine 866

A simplified interface file for a string class *(code\string7x.h) 867*

An alternate string(short Length) *constructor (from code\string7x.cc) 868*

Warning: Explicit Material 868

An explicit *constructor call vs an implicit one (code\strtstx.cc) 869*

The string(short, char) *constructor for the* string class *(from code\string7.cc) 870*

Adding Insult to Injury 871

Using operator + *for* string *concatenation (code\strtst7a.cc) 872*

The implementation of operator + *for the* string class *(from code\string7.cc) 873*

Inner Peace 875

The new implementation of operator >> *(from code\string7.cc) 876*

Location, Location, Location 879

Using string::find_nocase *(code\strtst7b.cc) 879*

The implementation of string::find_nocase *(from code\string7.cc) 881*

The less_nocase *function (from code\string7.cc) 884*

Home, Sweet Home **885**

The latest home inventory application program (code\hmtst6.cc) 885

The latest version of the HomeInventory *interface (hmin6.h) 887*

HomeInventory::FindItemByDescription *(from code\hmin6.cc) 888*

The new version of the HomeItem *interface (code\hmit6.h) 889*

HomeItemBasic::IsNull *(from code\hmit6.cc) 892*

Nothing Ventured, Nothing Gained 892

A slightly odd default constructor for HomeItem *(from code\hmit2.cc) 893*

Error messages triggered by accidental use of 0 to initialize a string (code\hmit6a.err) *894*

Putting It All Together **894**

First Test Session: Change Requests and Problem Reports 896

Second Test Session: Change Requests and Problem Reports 899

Third Test Session: Change Requests and Problem Reports 901

Fourth Test Session: Change Requests and Problem Reports 902

Fifth Test Session: Change Requests and Problem Reports 904

Round and Round We Go **905**

Review **906**

Exercises **911**

Conclusion **911**

CHAPTER 13 *Stealing Home 913*

Definitions **913**

Objectives of This Chapter **914**

The Final Voyage **915**

The main() *function of the final version of the home inventory main program (from code\hmtst8.cc) 915*

The MenuItem enum *(from code\hmtst8.cc) 917*

The GetMenuChoice *function (from code\hmtst8.cc) 917*

ExecuteMenuChoice *(from code\hmtst8.cc) 920*

Utility Room **926**

The HomeUtility *interface (code\hmutil1.h) 926*

HomeUtility::IgnoreTillCR *(from code\hmutil1.cc) 929*

HomeUtility::HandleError *(from code\hmutil1.cc) 930*

HomeUtility::CheckNumericInput *(from code\hmutil1.cc) 931*

Number, Please 932

Making a List, Checking It Twice 933

HomeUtility::CheckDateInput *(from code\hmutil1.cc) 933*

Enter Here 934

HomeUtility::GetNumberOrEnter *(from code\hmutil1.cc) 935*

Secret Decoder Ring Not Required 939

The Rest of the Story 940

I Can See Clearly Now 942

HomeUtility::ClearRestOfScreen *(from code\hmutil1.cc) 942*

The Final Frontier 943

The HomeUtility::SelectItem *function (from code\hmutil1.cc) 943*

Checking the Inventory **950**

The latest header file for the HomeInventory class *(code\hmin8.h) 950*

The latest version of AddItem *(from code\hmin8.cc) 952*

The new version of the EditItem *function (from code\hmin8.cc) 953*

The latest implementation of LocateItemByDescription *(from code\hmin8.cc) 953*

HomeInventory::LocateItemByCategory *(from code\hmin8.cc) 955*

The PrintNames *function (from code\hmin8.cc) 956*

The PrintAll *function (from code\hmin8.cc) 957*

The StoreInventory *function (from code\hmin8.cc) 958*

The DisplayItem *function (from code\hmin8.cc) 959*

A Better Sort of Function 959

The SortInventoryByName *function (from code\hmin8.cc) 960*

The SelectItemByPartialName *function (from code\hmin8.cc) 961*

The SelectItemFromNameList *function (from code\hmin8.cc) 962*

Categorical Imperative 964

The SelectItemFromCategoryList *function (from code\hmin8.cc) 964*

The DeleteItem *function (from code\hmin8.cc) 968*

Homing In **968**

The new operator >> *implementation (from code\hmit8.cc) 969*
The latest version of the HomeItemBasic::Edit *function (from code\hmit8.cc) 971*
The newest version of HomeItemBasic::ReadInteractive *(from code\hmit8.cc) 972*
The new version of the HomeItemBasic::EditItem *function (from code\hmit8.cc) 974*
The latest version of HomeItemMusic::ReadInteractive *(from code\hmit8.cc) 975*
The latest version of HomeItemMusic::EditField *(from code\hmit8.cc) 977*
Are We Having Fun Yet? **978**
Review **978**
Exercises **984**
Conclusion **985**

Appendix A *Tying up Loose Ends 987*
Operator Precedence 987
Another Native Data Type 988
Wrapping Up **989**

Appendix B *Glossary 991*

About the Author 1041

Index 1043