Programming - Errata

Errata merged July 22, 2012 by Nick Maclaren.

This is errata for the 1st printing of Programming: Principles and Practice using C++ giving the 4th printing; see that Web site for the separate, original errata.

Comments, improvements, bug reports, etc. are welcome.

I list both corrections and clarifications. There is essentially no problem that I consider too small to list, so you might find many pieces of errata too minor to bother with (I know because I have received email to that effect), but at this stage, I'd rather not make decisions about what an unknown reader will find bothersome. Please note that the number of errata does *not* equal the number of errors. I say this because I have seen the book condemned *unread* because the number of errata.

Big issues, such as "why don't you use XXX for your GUI?", "please add two more chapters on the STL", and "please use C++0x features to simplify the code" are not errata but considerations for future work and will not appear here.

Different people have different preferences for sorting errata: in chronological order, in page order, each printing separate, all printings merged, etc. However, I can't manage multiple organizations, so what you get is what seems to be most useful for most people.

At the request of repeat visitors to the page, I have started to add dates of posting of individual errata. If an errata is changed, so is its date.

As an abbreviation I use, s/before/after/ to mean replace "before" with "after".

Table of contents

- (+5/15/09) pg vi: s/delete repeated words/detect repeated words/

Chapter 0

- (+8/16/2010) pg 10: s/Software engineering principles first/"Software engineering principles first"/
- (+) pg 11: s/Appendix E/Appendix C/

Chapter 1

- (+) pg 22: s/animators,// (need not to be mentioned twice in the same sentence)
- (+12/8/09) pg 32: s/though web interfaces/through web interfaces/
- (+2/24/09) pg 34: s/www.research.att/~bs/applications.html/www.research.att.com/~bs/applications.html/

Chapter 2

- (+4/4/09) pg 48: s/You'll find that computers/You'll find that compilers/
- (+) pg 52: s/Appendix D/Appendix C/
- (+) pg 53: s/Appendix D/Appendix C/
- (+) pg 55: s/repeating exercise 4/repeating exercise 5/

Chapter 3

- (+5/20/09) pg 60: s/delete repeated/detect repeated/
- (+5/15/09) pg 61: s/read characters into name/read characters into first_name/
- (+4/11/2010) pg 64: s/**22** followed by some random number/**22** followed by **(age** followed by some random number/
- (+2/24/09) pg 65: s/Hello 22/Hello, 22/
- (+4/2/09) pg 65: s/and the character literal '\n'//
- (+4/15/09) pg 65: s/string literals "Hello, " and " (age "./string literals "Hello, ", " (age ", and ")\n"./
- (+4/11/2010) pg 65: s/5.5/five-and-a-half/
- (+) pg 67: s/3.7/3.9.1/
- (+4/11/2010) pg 67: s/another name for newline ("end of line")/another name for newline ("end of line") in output/
- (+5/15/09) pg 71: s/delete repeated words/detect repeated words/
- (+7/19/09) pg 73: s/count repeated words/find repeated words/

- (+12/25/09) pg 73: replace '/= and %= are referred to as "scaling"' by '*= and /= are referred to as "scaling"'
- (+4/11/2010) pg 74: s/"repeated: "<</"repeated: " <</
- (+12/25/09) pg 75: s/String s =/STRING s =/
- (+4/11/2010) pg 77: s/don't use/don't use the initial capital letter style/
- (+4/11/2010) pg 77: s/McDonald/MacDonald/
- (+4/25/2010) pg 80: s/error("d1 is negative");/cout << "d1 is negative";/
- (+4/19/2010) pg 81: s/double d =0;/double d = 0;/
- (+4/2/09) pg 82: s/[0,127]/[0:127]/

- (+4/19/2010) pg 82: s/(always rounds down)/(always rounds down; towards zero)/
- (+4/25/2010) pg 83: s/**error("you're kidding!")**./**simple_error("you're kidding!")** using **simple_error()** from **std_lib_facilities.h**./
- (+4/19/2010) pg 86: s/You may ...; it can be done.//

Chapter 4

- (+4/25/2010) pg 90: s/4.6 Vector/4.6 vector/
- (+4/18/2010) pg 96: replace the definition of v with

```
double c = 2*pi*r;      // OK: we just read pi; we don't try to change it
```

- (+4/18/2010) pg 96: s/other constants embedded/other literals embedded/
- (+2/24/09) pg 99: s/converted into a double/changed into a double/
- (+3/11/09) pg 99: s/syntax error: missing semicolons/syntax error: missing semicolon/
- (+) pg 102: s/int length = 1;/double length = 1;/
- (+) pg 103: s/int length = 1;/double length = 1;/ *(quite reasonably, people want to input fractions of centimeters)*
- (+) pg 104: s/int length = 1;/double length = 1;/
- (+) pg 107: s/int length = 1;/double length = 1;/
- (+8/16/2010) pg 107: s/with a break/with a **break**/.
- (+4/18/2010) pg 108: s/++i ;/++i;/
- (+) pg 111: s/int = 0;/ int i = 0;/
- (+2/24/09) pg 112: s/get a table/write out a table/
- (+5/7/2010) pg 113: s/(but if we didn't want the results, why would we call it?)//
- (+2/24/09) pg 114: s/Why didn't we use that version/Why didn't we use the version/
- (+6/13/2010) pg 114: s/you'd soon tire of repeating their definition./you'd soon tire of repeating equivalent code./
- (+2/24/09) pg 115: s/piece a paper/piece of paper/
- (+4/25/2010) pg 116: s/4.6 Vector/4.6 vector/
- (+4/3/09) pg 117: add a line after the example near the bottom of the page using "" in a comment:

```
The string with no characters "" is called the empty string.
```

- (+8/16/2010) pg 117: s/philosopher [/philosopher[/ (4 times)
- (+4/18/2010) pg 118: s/<<v[i]/<< v[i]/
- (+4/18/2010) pg 120: s/i< temps.size()/i<temps.size()/
- (+8/16/2010) pg 120: s/elements into sum,/elements into **sum**,/
- (+4/18/2010) pg 121: s/i< temps.size()/i<temps.size()/
- (+4/18/2010) pg 122: s/i< words.size()/i<words.size()/
- (+6/8/09) pg 123: In (words[i-1]!=words[i]) the (round) parentheses should be black (not code font)
- (+5/5/09) pg 124: in exercise 5: s/10000000/100/
- (+8/31/09) pg 124: in exercise 7: add: You may consider 12 m (with space between the number and the unit) equivalent to 12m (without a space).
- (+4/19/2010) pg 124: s/When you see the final '|'/When the loop ends/
- (+4/19/2010) pg 125: s/vector<char>alphabet(26);/vector<char> alphabet(26);/
- (+4/3/09) pg 126: s/five basic/four basic/
- (+3/26/09) pg 126: s/divide, and modulus (remainder)/and divide/
- (+4/19/2010) pg 126: s/"the number for which ... after it,"/"a number so that exactly as many elements come before it in the sequence as come after it,"/
- (+) pg 127: in Exercise 9: s/Observe what happens when the number gets too large to represent as an int and as a double/ Observe what happens when the number gets too large to represent exactly as an int and as a double/ Also add, Please note that you need to use many more than 64 squares before you exhaust a double's ability to give an approximate answer.
- (+)pg 128: the 2 in ax2+bx+c=0 should be a superscript (x squared)
- (+2/24/09) pg 128: s/scores[7]==18/scores[7]==17/
- (+3/16/09) pg 128: s/Terminate input ... (...)/Terminate the input with **NoName 0**./ *(Using the technique*

originally suggested could easily get you into problems in exercise 20).

Chapter 5

- (+4/2/09) pg 135: s/int s1 = area(7)/int s2 = area(7)/
- (+2/24/09) pg 137: s/OK,/OK:/
- (+10/30/2010) pg 138: s/the argument type)/the argument types)/
- (+5/7/2010) pg 138: s/exactly on its type./exactly on its type. For more details see §8.2-3./
- (+5/7/2010) pg 140: s/most common answer./most common approach./
- (+10/30/2010) pg 140: s/This suffices/This approach sufficies/
- (+4/2/09) pg 141: s/frame_area/framed_area/ twice
- (+5/22/09) pg 141: s/2nd area() argument/argument for area()/
- (+2/24/09) pg 142: s/frame_area/framed_area/
- (+4/2/09) pg 143: s/5.9/5.10/
- (+3/9/09) pg 145: s/width <=0/width<=0/
- (+2/24/09) pg 146: s/a vector ints/a vector of ints/ *(in the comment)*
- (+2/24/09) pg 147: s/a vector ints/a vector of ints/ *(in the comment)*
- (+) pg 147: s/out_of_range_error/out_of_range/
- (+5/7/2010) pg 149: s/a few exceptions/a few types of exceptions/
- (+3/9/09) pg 151: s/it produces a new value corresponding to its operand of the required type./ it produces a new value (of the type specified in the < ... >) that corresponds to its operand value./
- (+5/7/2010) pg 152: s/control statements/if-statements/
- (+3/20/09) pg 155: s/9.65685/10.3923/
- (+) pg 157: s/terminal/computer/
- (+8/16/2010) pg 159: s/after the 0/after the **0**/
- (+5/7/2010) pg 162: s/no comments documented/no comments/
- (+4/5/09) pg 168, exercise 7: remove redundant advice: s/and have it throw ... if there is an error.// *(The reader can decide what to do about errors).*
- (+9/23/09) pg 168, exercise 8: change to simplify exercise: s/Write a program ... 48."/

```
Write a program that reads and stores a series of integers
and then computes the sum of the first N integers.
First ask for N, then read the values into a vector,
then calculate the sum of the first N values. For example:

        Please enter the number of values you want to sum:
                3
        Please enter some integers (press '|' to stop):
                12 23 13 24 15 |
        The sum of the first 3 numbers ( 12 23 13 ) is 48
```

- (+7/7/2010) pg 168: s/, starting with the first//

Chapter 6

- (+6/22/09) pg 178: Add an output line and a test to the example:

```
cout << "Please enter expression (we can handle +, -, *, and /)\n";
cout << "add an x to end expression (e.g., 1+2*3x): ";
int lval = 0;
int rval;
char op;
cin>>lval;               // read leftmost operand
if (!cin) error("no first operand");
while (cin>>op) {        // read operator and right-hand operand repeatedly
        if (op!='x') cin>>rval;
if (!cin) error("no second operand");
```

- (+5/17/2010) pg 179: s/among digits and plusses/among digits, plusses, minuses, and parentheses/
- (+) pg 183: s/We can read out input/We can read input/
- (+) pg 183: s/read token from cin/function to read token from cin/
- (+) pg 189: s/(again leaving/(leaving/
- (+) pg 190: s/Expression*Term rule/Expression+Term rule/
- (+5/9/2010) pg 197: s/to avoid/to apply/
- (+) pg 198: s/6.5.3.1/6.5.2.1/
- (+5/24/09) pg 199: /question 4 above/question 5 in sec6.3.5/
- (+) pg 202: s/5+6+7/5+6/ twice *(if you use 5+6+7 the output is far harder to understand with than that from plain 5+6, and the solution harder to spot).*
- (+) pg 203: s/5+6+7/5+6/ three times
- (+) pg 203: s/18/11/ twice

- (+4/2/09) pg 207: s/6.5.1/6.3.3/
- (+5/17/2010) pg 208: s/in the definition of Token above./in the definition of Token in §6.3.3./
- (+4/2/09) pg 209, end of page: empty shouldn't be in code font
- (+6/13/2010) pg 209: s/ get() is defined elsewhere/ get() is defined in §6.8.2/
- (+) pg 210: s/6.5.1/6.3.3/
- (+) pg 211: the opening { of the Token::get() function is indented -- it should not be. *(A compiler can't tell the difference, but we can and code should be aestetically pleasing).*
- (+5/25/09) pg 211: s/ case '%'://
- (+5/17/2010) pg 213: s"// deal with *, / and %"// deal with * and /" *(using " as the substitution separator)*
- (+) pg 215: Questions 1 and 6 should be merged into one question 1.
- (+2/23/2010) pg 216: Add a hint to exercise 3: Hint: The calculator functions deal with **double**s, but factorial is defined only for **int**s, so just for **x!** assign the **x** to an **int** and calculate the factorial of that **int**.
- (+) pg 217: s/P(a-b)/P(a,b)/

Chapter 7

- (+5/22/2010) pg 221: s/If we added/If we used/
- (+5/22/2010) pg 225: s/be the next character to be read/be a character to be read/
- (+4/2/09) pg 226: s/The first thing that expression does is to look for a **primary()**,and now it finds **q**. / The first thing that **expression()** does is to call **term()**, which first calls **primary()**, which finds **q**./
- (+5/22/2010) pg 226: s/isn't a primary/isn't a **Primary**/
- (+4/20/09) pg 229: s/= 0/= 2/
- (+12/26/09) pg 229: s/double d = term();/double d = primary();/
- (+3/22/09) pg 230: s/left *= term();/left *= primary();/
- (+3/22/09) pg 230: s/double d = term();/double d = primary();/ twice
- (+) pg 230: in the bottom code fragment: s/f (i2==0)/if (i2==0)/
- (+5/23/2010) pg 231: s/defining a number/defining an object/
- (+5/22/2010) pg 231: s/**number ='0'**/For example, an assignment **number ='0'**/
- (+) pg 232: s/cin.putback()/cin.putback(ch)/
- (+5/22/2010) pg 234: delete the comment "//quit"
- (+5/23/2010) pg 236: s/repeat an action/explain something/
- (+) pg 237: properly align "- Primary" and "+Primary"
- (+9/20/09) pg 238 s/**Tokenstream**'s buffer/the **Tokenstream**'s and **cin**'s buffers/
- (+5/22/2010) pg 238: delete the comment "//quit"
- (+4/11/09) pg 247: s/struct Token {/class Token { public:/ with the "public:" on its own line, like this

  ```
  class Token {
  public:
  ```

  (to exactly match pg 182).
- (+4/2/09) pg 247: s/7.8.2/7.8.1/
- (+7/9/2010) pg 251: Replace review #18 by: "18. How do you chose names for variables and functions? List possible reasons."
- (+5/22/2010) pg 251: s/calculator's names/calculator's variable names/
- (+) pg 252: s/the H key./the H key (both upper and lower case)./
- (+4/2/09) pg 252: s/After analyzing its output/After analyzing its input/

- (+12/26/09) pg 252: exercise 4: s/declare_name()/define_name()/
- (+5/23/2010) pg 252: exercise 4: s/global variable/variable/
- (+12/26/09) pg 252: exercise 9: s/declare()/define()/

Chapter 8

- (+7/9/2010) pg 257: s/ each must be matched/if the entity it refers to is used each must be matched/
- (+3/9/09) p259 s/hundred of thousands/hundreds of thousands/
- (+7/9/2010) pg 261: s/we have defined string and vector to be initialized/string and vector are defined so that variables of those types are initialized/
- (+5/23/2010) pg 261: s/A global variable/A global variable (§8.4)/
- (+2/24/09) pg 262: s/f.cpp/user.cpp/
- (+2/24/09) pg 263: s/f.cpp/user.cpp/ twice. *(And really that example would have been better if I had also provided a f.cpp in which f() was defined).*
- (+5/23/2010) pg 267: Improve the comments in the first example:

  ```
  int x;      // local variable, hides the global x
  ...
  int x = y;  // local x initialized by global y, hides the previous local x
  ```

- (+8/16/2010) pg 271: s/return statement/**return**-statement/
- (+7/9/2010) pg 272: Start a new paragraph after "value return is a form of initialization."
- (+6/13/2010) pg 272: s/const vector v, const string/vector v, string/
- (+2/24/09) pg 275: s/or (int i=0;/for (int i=0;/
- (+4/11/09) pg 275: in the label "Refer to vd2 in 2nd call" vd2 should be in **code** font
- (+4/2/09) pg 277: s/(did not use pass-by-const-value)/(did not use pass-by-const-reference)/
- (+3/15/09) pg 278: s/copy d2's value to d2/copy d2's value to d1/ (in the comment)
- (+5/23/2010) pg 280: s/For manipulating containers (e.g., vector)"/For manipulating containers (e.g., vector) and other large objects"/
- (+1/20/2010) pg 280: s/if (v1.size()!=v2.size()/if (v1.size()!=v2.size())/
- (+8/18/09) pg 281: replace the line **double x(y);** with

```
double x(y);    // initialize x with y (see sec8.2.2)
```

- (+3/9/09) pg 287 s/first in, first out./last in, first out./
- (+5/23/2010) pg 289: s/the default Date/the default date/
- (+4/2/09) pg 290: s/to prevent the called function/to prevent the calling function/
- (+6/13/2010) pg 290: s/A static local/The static local/
- (+8/16/2010) pg 292: s/namespace directive/**using** directive/
- (+10/28/2010) pg 293: Drill 1. Add at end: "Hint: you need to **#include<iostream>** to use **cin**."
- (+6/13/2010) pg 294: s/Which calls compiled, and why?/Which functions and calls compiled, and why?/
- (+4/2/09) pg 296, exercise 1: s/and istream& parameter/an istream& parameter/
- (+4/2/09) pg 296, exercise 3: s/we get 1, 2, 3, 6, 9, 15, 24/we get 1, 2, 3, 5, 8, 13, 21/ *(amazing mistake!)*
- (+8/6/09) pg 296, exercise 7: s/Hint: Before sorting **age**, take a copy and use that to make a copy of **age** in the right order after sorting **age**/Hint: Before sorting **name**, take a copy and use that to make a copy of age in the right order after sorting **name**/

- (+6/14/2010) pg 296: s/should make such a series/should make such a sequence/

Chapter 9

- (+2/24/09) pg 307: The dot in /initializer syntax./ is blue (code font); it should be black (text font).
- (+6/13/2010) pg 307: Replace 12/24/2007/ with **(12,24,2007)**
- (+3/25/2010) pg 308: s/Ouch! invalid date/Ouch! invalid date (birthday.d==32 makes today invalid)/
- (+3/25/2010) pg 308: s/Ouch! invalid date/Ouch! invalid date (today.m==14 makes today invalid)/
- (+11/16/09) pg 309: s/Georgian/Gregorian/ twice
- (+6/13/2010) pg 310: s/the order of class members/the order of class function and data members/
- (+6/13/2010) pg 310: s/The :y(yy), m(mm), d(dd) notation is how we initialize members. /The :y(yy), m(mm), d(dd) notation is how we initialize members. It is called a (member) initializer list. /
- (+4/2/09) pg 312, last paragraph: s/print out /return the value of/
- (+6/13/2010) pg 312: s/can refer to another member/ can refer to a function or data member/
- (+6/13/2010) pg 312: s/ The function will be ... ./The function will be *inline*; that is, the compiler will try to generate code for the function at each point of call rather than using function call instructions to use common code./
- (+6/13/2010) pg 312: s/, say five lines of code,/, say five or more lines of code,/
- (+3/29/09) pg 314: s/and add_date() will have/and add_day() will have/
- (+4/2/09) pg 317: s/Month m = sep;/Month m = Sep;/
- (+6/13/2010) pg 320: s/(x<min || max<x)/ (x<min || max<=x)/
- (+6/13/2010) pg 321: s/if you copy a **Month**/if you copy a **Date**/
- (+2/24/09) pg 323: s/for (int i=0, i<s.size(), ++i)/for (int i=0; i<s.size(); ++i)/
- (+6/13/2010) pg 323: change the toupper line:

```
s[i] = toupper(s[i]); // oops: read and write a random memory location
```

- (+6/13/2010) pg 324: s/Date& default_date()/const Date& default_date()/
- (+3/7/2010)pg 330: add after the first comment line:

```
#include "Chrono.h"
```

- (+10/30/2010) pg 330: s/Date& default_date()/const Date& default_date()/
- (+6/13/2010) pg 331: add a test of **m** after the test of **d**:

```
If (m<Date::jan || Date::dec<m) return false;
```

- (+02/11/2010) pg 332: Add before the last return statement in **operator>>()**:

```
dd = Date(y,Date::Month(m),d);     // update dd
```

- (+3/11/09) pg 334: s/operator<</operator <</ with only the "<<" in code font

- (+3/11/09) pg 334: s/have functions that access these methods/have functions that access this data/
- (+3/11/09) pg 334: s/Have a helper method/Have a helper function/
- (+2/5/2010) pg 334: s/the title author, and ISBN/the title, author, and ISBN/ (in excercise #6)
- (+3/11/09) pg 335: s/Also create a method/Also write a function/
- (+3/11/09) pg 335: s/a set of useful helper function/a set of useful helper functions/

- (+7/5/2010) pg 335: s/**long**/**long int**/ four times.

Chapter 10

- (+5/7/2010) pg 340: s/and then access/and programs then access/
- (+7/9/2010) pg 345: s/check that the file/checks if the file/ twice
- (+5/8/2010) pg 345: s/file named name/file named oname/
- (+8/16/2010) pg 345: s/ost <</ ost <</ (indent correctly)
- (+6/12/09) pg 346: s/if(fs)/if(!fs)/
- (+8/16/2010) pg 346: s/ifs is already open/fs is already open/ (in comment)
- (+7/9/2010) pg 347: s/(hour of day,/(hour,/
- (+3/13/09) pg 351: s/see sec10.6 and // (This *is* section 10.6)
- (+2/24/09) pg 351: s/and Langer, Standard C++ IOStreams and Locales./and Standard C++ IOStreams and Locales by Langer./

- (+5/8/2010) pg 351: s/simplify all input loops by/simplify all input loops on **ist** by/
- (+5/16/2010) pg 355: s/isdigit(ch)/isdigit(ch) || ch=='-'/

Chapter 11

- (+) pg 377: s/A.1.2.1/A.2.1.1/
- (+6/16/2010) pg 380: s/similar to that of decimal values/similar to that of integer values/
- (+5/16/2010) pg 382: s/round down/round down (towards zero)/
- (+5/16/2010) pg 382: s/round up/round up (away from zero)/
- (+3/23/09) pg 386: s/ofstream ifs/ifstream ifs/
- (+) pg 388: s/as_byte()/as_bytes()/
- (+3/9/09) pg 389: put a 'y' in the second character of the file *(now the figure represent the state of the program after the code has executed).*
- (+6/12/09) pg 389: adjust example to match figure (note that the put() and get() increment their respective pointers):

```
fs.seekg(5);    // move reading position ("g" for "get") to 5 (the 6th character)
char ch;
fs>>ch; // read and increment reading position
cout << "character[5] is " << ch << '(' << int(ch) << ")\n";

fs.seekp(1);    // move writing position ("p" for "put") to 1
fs<<'y';        // write and increment writing position
```

- (+5/16/2010) pg 390: s/stringstream/istringstream/ three times in the middle paragraph
- (+7/5/2010) pg 390: s/ a string stream really is a kind of **istream**/an **istringstream** really is a kind of **istream**/
- (+6/12/09) pg 391: s/A simple use of an **ostream**/A simple use of an **ostringstream**/
- (+6/13/2010) pg 393: s/**c isalpha()|isdigit()|ispunct()**/**isalpha(c)** or **isdigit(c)** or **ispunct(c)**/
- (+3/20/09) pg 395: in code listing: // missing before "by a space"
- (+3/20/09) pg 397: s/stringstream/istringstream/
- (+3/20/09) pg 399: s/We read a line into buffer/We read a line into line/
- (+3/20/09) pg 399: s/stringstream/istringstream/
- (+) pg 402: s/istreambuf/streambuf/

Chapter 12

- (+2/26/09) pg 409: s/HTTP/HTML/
- (+3/20/09) pg 409: s/il/li/ twice on the last line
- (+)pg 410: s/we have added to our window/we have attached to our window/
- (+6/12/09) pg 412: Remove the paragraph starting "On our screen".
- (+3/11/09) pg 415: in the "coordinate diagram" s/100,200/200,100/
- (+7/13/2011) pg 416: s/1450-by-1050/1400-by-1050/
- (+5/9/09) pg 418: Add as the last sentence of 12.7.1 after the code example: "For this **main()** to compile, we need to have **exception** defined. We get that if we include **std_lib_facilities.h** as usual or we could start to deal directly with standard headers and include **<stdexcept>**."
- (+3/21/09) pg 425: the quotes in "Canvas #6" should be normal quotes
- (+) pg 425: The second Point(100,50) should be Point(200,50).

- (+3/26/2010) pg 425: add to the end of the example at the bottom of the page **win.attach(poly_rect);**.
- (+) pg 427: add a line to the code example (before the set_label call):

```
poly_rect.set_fill_color(Color::green);
```

- (+5/16/2010) pg 431: s/stringstream/ostringstream/
- (+7/9/2010) pg 433: s/, **Window.h**, and **GUI.h**/ and **Simple_window.h**/
- (+3/26/2010) pg 434: pg 434: s/HTTP/HTML/
- (+12/8/09) pg 434: Change exercise 4 to: Draw a 3-by-3 tic-tac-toe board of alternating white and red squares.
- (+3/26/2010) pg 435: in 13: s/the superellipse shapes/the lines/
- (+10/5/09) pg 435: in 12: s/the number of points connect to/the number of other points to connect a point to/

Chapter 13

- (+7/13/2011) pg 440: s/Window.ccp/Window.cpp/
- (+6/23/09) pg 441: s/max_x()/x_max()/
- (+6/23/09) pg 441: s/max_y()/y_max()/
- (+5/16/2010) pg 442: s/to add to/to add a **Point** to/
- (+2/05/2010) pg 444: s/we get/we get:/
- (+) pg 445: s/fl_draw()/fl_line()/
- (+5/16/2010) pg 447: s/Transparency::/Color::/ twice
- (+8/16/2010) pg 448: s/**-..-..**/-..-../
- (+6/12/09) pg 454: s/Shape::add(p)/Closed_polyline::add(p)/
- (+5/17/2010) pg 454: s/existing lines/existing lines (code not shown)/
- (+7/23/09) pg 455: s/Rectangle(Point xy, int hh, int ww)/Rectangle(Point xy, int ww, int hh)/
- (+3/24/09) pg 461: s/8-by-8/16-by-16/
- (+5/17/2010) pg 461: s/write a simple color chart/draw a simple color chart/
- (+5/16/2010) pg 461: s/set_fill_color(i*16+j);/set_fill_color(Color(i*16+j));/
- (+7/20/09) pg 463: s/return Font(fnt);/return fnt;/
- (+11/5/2010) pg 464: Change the definitions of set_radius():

```
void set_radius(int rr)
{
        set_point(0,Point(center().x-rr,center().y-rr));        // maintain the center
        r = rr;
}
```

- (+5/17/2010) pg 466: s/write parts of a circle/draw parts of a circle/
- (+11/5/2010) pg 466: Change the definitions of set_major() and set_minor():

```
void set_major(int ww)
{
        set_point(0,Point(center().x-ww,center().y-h)); // maintain the center
        w = ww;
}

void set_minor(int hh)
{
        set_point(0,Point(center().x-w,center().y-hh)); // maintain the center
        h = hh;
}
```

- (+5/16/2010) pg 467: A better definition of focus1():

```
Point focus1() const
{
        if (h<=w)       // foci are on the x-axis:
                return Point(center().x+int(sqrt(double(w*w-h*h))),center().y);
        else            // foci are on the y-axis:
                return Point(center().x,center().y+int(sqrt(double(h*h-w*w))));
}
```

- (+5/16/2010) pg 469: improve **Marked_polyline**'s constructor:

```
Marked_polyline(const string& m) :mark(m) { if (m=="") mark = "*"; }
```

- (+3/9/09) pg 470: Add a paragraph just before section "13.16 mark": "The :Marked_polyline(m) notation is used to initialize the Marked_polyline part of a Marks object. This notation is a variant of the syntax used to initialize members (9.4.4)."
- (+6/12/09) pg473: s/path's point (50,600)/path's point (50,250)/

- (+5/17/2010) pg 474: simplify the character literal '\"' to '"'
- (+5/17/2010) pg 475: simplify the character literal '\"' to '"'

- (+5/16/2010) pg 475: s/, but effective,//
- (+6/16/2010) pg 475 s/map (see chapter 18)/map (see §21.6)/
- (+4/25/2010) pg 476: s/RBG/RGB/
- (+5/16/2010) pg 477: s/(e.g. see www....)/(e.g. search the web for "RGB color chart")/
- (+5/16/2010) pg 477: s/Hexagon/Regular_hexagon/ twice
- (+5/16/2010) pg 477: s/(a hexagon is a regular six-sided polygon)/(a regular hexagon is a six-sided polygon with all sides of equal length)/

Chapter 14

- (+3/20/09) pg 489: s/points start out/points starts out/
- (+3/20/09) pg 489: s/to the members to all members/to all members/
- (+5/16/2010) pg 489: s/making any member of a class public/making any data member of a class public/
- (+5/16/2010) pg 490: s/set_points()/set_point()/
- (+5/16/2010) pg 490: s/is there,/is provided/
- (+5/16/2010) pg 490: s/size of the program/size of the generated code/
- (+9/13/2010) pg 491: s/line_color/lcolor/
- (+7/5/2010) pg 491: Clarification of the second paragraph of 14.2.3:

```
Shape�s most basic job is to draw shapes. We could remove all
other functionality from Shape or leave it with no data of its own
without doing major conceptual harm (see �14.4), but drawing is Shape�s
essential business. It does so using FLTK and the operating system�s
basic machinery, but from a user�s point of view, it provides just two
functions:"
```

- (+3/31/09) pg 492: s/draw_line()/draw_lines()/ twice
- (+9/13/2010) pg 492: s/tries to restore the color and shape/tries to restore color and style/
- (+7/13/2011) pg 494: s/const Open_polyline&/Open_polyline&/
- (+6/12/09) pg 495 s/The Window would know nothing about a copy, so a /For example, if a Window held only a copy of a Shape, rather than a reference to the Shape, changes to the original would not affect the copy. So if we changed the Shape's color, the Window would not notice the change and would display its copy with the unchanged color. A/
- (+9/15/2010) pg 495 (in the figure): s/line_color/lcolor) twice and add "fcolor"
- (+5/16/2010) pg 496: s/for a Circle/for a Shape that is a Circle/
- (+9/15/2010) pg 497 (in the figure): s/line_color/lcolor) twice and add "fcolor"
- (+6/4/09) pg 498: s/vpt/vptr/ twice in the figure
- (+6/4/09) pg 498: s/:/::/ three time in the figure
- (+9/15/2010) pg 498 (in the figure): s/line_color/lcolor) twice and add "fcolor"
- (+5/16/2010) pg 502: (the list on the middle of the page): s/If a base is public, its name can be used by all functions./If a base is public, its public member names can be used by all functions./
- (+5/16/2010) pg 503: s/overridden in a derived class/overridden in some derived class/
- (+6/12/09) pg 504: s/we could override **Shape::draw()**/we could override **Shape::draw_lines()**/

- (+5/16/2010) pg 508: s/double*/double* (see Chapter 17)/
- (+4/25/2010) pg 508: s/runtime_exception/runtime_error/

Chapter 15

- (+6/16/2010) pg 512: s/magic numbers/magic constants/
- (+6/19/09) pg 515: s/orig, r_min, r_max/r_min, r_max, orig/
- (+6/19/09) pg 516: s/orig, r_min, r_max/r_min, r_max, orig/
- (+5/17/2010) pg 516: s/default arguments for trailing argument/default arguments for trailing parameter/
- (+5/17/2010) pg 516: s/If an argument has a default argument, all subsequent arguments/If a parameter has a default argument, all subsequent parameters/
- (+11/5/2010) pg 517: s/20,20/30,30/ twice
- (+3/22/09) pg 518: s/result is/result is:/
- (+5/17/2010) pg 519: s/(1<n)/(0<n)/ twice
- (+5/17/2010) pg 523: s/ss.str().c_str()/ss.str()/
- (+3/31/09) pg 525: The lower image should be that of the iteration with ten terms: ten.
- (+10/4/2010) pg 526: Replace the first paragraph with "Remember, the computer�s arithmetic is not pure math. Floating-point numbers are simply as good an approximation to real numbers as we can get with a fixed number of bits. An **int** overflows if you try to place a too large integer in it, whereas a **double** stores an approximation. When I saw the strange output for larger numbers of terms, I fist suspected that our calculation started to produce values that couldn�t be represented as **double**s so that our results started to diverge from the mathematically correct answers. Later, I realized that **fac()** was producing values that couldn�t be stored in an **int**. Modifying **fac()** to produce a **double** solved the problem. For more information, see exercise 11 of

Chapter 5 and �24.2."
- (+3/31/09) pg 531: last line: s/xy/ys/
- (+5/17/2010) pg 531: Add a comment:

```
int operator()(int v) const { return cbase + (v-vbase)*scale; } // see §21.4
```

- (+5/6/09) pg 532: s/current_line/current_year/
- (+4/3/09) pg 533: s/xy/ys/
- (+4/3/09) pg 537: s/11- .../11+ .../
- (+3/22/09) pg 537: s/about 2.54cm/2.54cm/ *(e.g., see definitions).*

Chapter 16

- (+4/3/09) pg 540: s/or XML//
- (+6/12/09) pg 544: s/wait_for_button();/void wait_for_button();/
- (+3/24/09) pg 547: s/becomes unobscured/becomes visible/
- (+6/14/2010) pg 547: Change the definition of botton_pushed to

```
        bool button_pushed; // initialized to false in the constructor
```

- (+3/24/09) pg 548: indent ``virtual void move(int dx,int dy);''
- (+5/17/2010) pg 551: s/unsigned //
- (+4/6/09) pg 552: There should be no color button on the screen shot.
- (+4/6/09) pg 553: There should be no color buttonon the screen shot.
- (+5/17/2010) pg 554: s/idiom for delete window/idiom to delete window/
- (+5/17/2010) pg 555: /stringstream/ostringstream/
- (+5/17/2010) pg 555: /a stringstream/an ostringstream/
- (+5/17/2010) pg 555: /stringstream/ostringstream/
- (+6/12/09) pg 561: The member initializers should be in declaration order, that is:

```
Lines_window::Lines_window(Point xy, int w, int h, const string& title)
    :Window(xy,w,h,title),
    next_button(Point(x_max()-150,0), 70, 20, "Next point", cb_next),
    quit_button(Point(x_max()-70,0), 70, 20, "Quit", cb_quit),
    next_x(Point(x_max()-310,0), 50, 20, "next x:"),
    next_y(Point(x_max()-210,0), 50, 20, "next y:"),
    xy_out(Point(100,0), 100, 20, "current (x,y):"),
    color_menu(Point(x_max()-70,30),70,20,Menu::vertical,"color"),
    menu_button(Point(x_max()-80,30), 80, 20, "color menu", cb_menu)
{
```

- (+3/24/09) pg 562: s/line for line/line by line/

Chapter 17

- (+) pg 574: In the 2nd figure on the page is the number 4099 (in bold); that should be 4096 (of course).
- (+3/12/09) pg 580: s/8.9/8.8/
- (+5/31/09) pg 589: s/the **Text** object/the **Text** (13.11) object/
- (+5/31/09) pg 590: s/delete p/delete q/ four times
- (+5/31/09) pg 590: s/delete looks at p'/delete looks at q's/
- (+) pg 599: s/norse_gods = new Link("Odin",norse_gods,0);/norse_gods = new Link("Odin",0,norse_gods);/
- (+) pg 599: s/norse_gods = new Link("Freia",norse_gods,0);/norse_gods = new Link("Freia",0,norse_gods);/ *The code as written on page 599 will crash. Actually, the code posted on www.stroustrup.com/Programming says it will crash, but the text does not. This became a far better than intended example of why I (as stated) dislike this kind of code. Oops!*
- (+4/16/09) pg 604: delete the second version of the find() function (the const ... const version). *overloading on const is not explained until later*.
- (+3/31/09) pg 604: s/We left the value/We left value/ with "value" in code font

Chapter 18

- (+6/6/2010) pg 613: s/elem(new double[s]) { }/elem(new double[s]) { /* ... */ }/
- (+7/5/2010) pg 614: add the value 0.0 twice (for v[0] and v[1]) in the figure for v.
- (+) pg 617: s/copy(a);/for(int i=0; i<a.sz; ++i) p[i]=a.elem[i];/
- (+) pg 618: s/copy(a);/for(int i=0; i<a.sz; ++i) p[i]=a.elem[i];/
- (+4/10/09) pg 621, second line: the "and" before vector<int>() should not be blue.
- (+6/23/09) pg 622: s/<double>// five times
- (+3/21/09) pg 623: The definition of X prints only the old value of an X; it would be more useful if both the old and the new value was printed. Replace the definition of X by:

```
struct X {        // simple test class
    int val;

    void out(const string& s, int nv)
        { cerr << this <<  "->" << s << ": " << val << " (" << nv << ")\n"; }

    X(){ out("X()",0); val=0; }                     // default constructor
    X(int v) { out( "X(int)",v); val=v; }
    X(const X& x){ out("X(X&)",x.val); val=x.val; }  // copy constructor
    X& operator=(const X& a)                // copy assignment
            { out("X::operator=()",a.val); val=a.val; return *this; }
    ~X() { out("~X()",0); }                 // destructor
};
```

- (+) pg 624: s/delete pp;/delete[] pp;/
- (+4/14/09) pg 627: s/const double &/const double&/
- (+6/6/2010) pg 627: s/Use **vector**/Use **std::vector**/
- pg 630: *Expert-level comment: the "easy to get wrong" backwards loop actually has a subtle error: It forms a pointer to the (non-existent) element one before the array and compares that to &ad[0]. That's not standards conforming and a very aggressive optimizer may generate bad code for that. However, that's unlikely because there is a lot of such code "out there". A correct loop:*

  ```
  for (double* p = &ad[10]; p>&ad[0]; --p) cout << *(p-1) << '\n';
  ```

  However, you can't just substitute in that version because then the text would be wrong.
- pg 630: *Expert-level comment: the "easy to get wrong" backwards loop actually has a subtle error: It forms a pointer to the (non-existent) element one before the array and compares that to &ad[0]. That's not standards conforming and a very aggressive optimizer may generate bad code for that. However, that's unlikely because there is a lot of such code "out there". A correct loop:*

  ```
  for (double* p = &ad[10]; p>&ad[0]; --p) cout << *(p-1) << '\n';
  ```

  However, you can't just substitute in that version because then the text would be wrong.
- (+7/5/2010) pg 633: s/something like **vector**/something like the standard library **vector**/
- (+5/31/09) pg 639: s/last character read into p/last character read into buffer/
- (+6/6/2010) pg 641: s/return is_palindrome(++first,��last);/return is_palindrome(first+1,last-1);/
- (+4/15/09) pg 642: s/argument array/argument vector/ (with "vector" in code font)

- (+8/14/2010) pg 643: exercise 9: s/Consider the memory layout in �17.3/Consider the memory layout in �17.4/

Chapter 19

- (+) pg 648: s/vector<double> d;/vector<double> vd;/
- (+4/15/09) pg 648: s/be not be/not be/
- (+6/2/2010) pg 650: s/**space** is 0/**space==sz**; that is, there is no "free space'':/
- (+6/2/2010) pg 654: s/self-reference (e.g., **v=v**)/self-assignment (e.g., **v=v**)/
- (+3/13/09) pg 655: s/vector(int s)/explicit vector(int s)/
- (+6/2/2010) pg 655: s/for 0<=n<sz elem[n] is element n/if 0<=n<sz, elem[n] is element n/
- (+3/13/09) pg 657: s/vector(int s)/explicit vector(int s)/
- (+3/13/09) pg 657: s/vector_char(int s)/explicit vector_char(int s)/
- (+6/2/2010) pg 657: s/size+free_space/size + free space/ twice
- (+6/14/2010) pg 659: s/for a give argument type,/for given object and argument types,/
- (+3/14/09) pg 663 in "(the size parameter N)", the "N" should be in the blue code font
- (+5/5/09) pg 664: s/**array** doesn't have those problems/like **vector**, **array** doesn't have those problems/
- (+7/9/2010) pg 664: Add a comment to the definition of printout:

  ```
  template<class C> void printout(const C& c)     // function template
  ```

- (+6/7/2010) pg 665: s/add 200/add 100/
- (+6/7/2010) pg 665: s/add 300/add 100/
- (+8/27/09) pg 666 s/tries to make 100 No_default()s/tries to make 200 No_default()s/
- (+6/7/2010) pg 666: s/make 200/add 100/
- (+6/2/2010) pg 666: s/four fundamental operators/four fundamental operations/
- (+6/2/2010) pg 667: s/how a vector can deal/how a **vector** can deal/
- (+6/2/2010) pg 669: replace

  ```
  cin>>i;
  while (i!=-1) try {
  ```

  ```
  while(cin>>i && i!=-1)
  try {
  ```

- (+6/2/2010) pg 671: s/unsigned int/size_type/ twice
- (+6/2/2010) pg 677: s/some_function()/make_vec()/
- (+6/2/2010) pg 677: s/(vector::at())/(e.g., vector::at())/
- (+8/27/09) pg 678 s/with the object we got from new./with the pointer we got from new./
- (+3/13/09) pg 671: s/Vector(size_type n)/explicit Vector(size_type n)/
- (+4/5/09) pg 672: s/ Fine handles/File handles/
- (+3/17/09) pg 678: s/19.3.5./19.3.6./
- (+12/28/2010) pg 679: s/elem(a.allocate(n))/elem(alloc.allocate(n))/
- (+) pg 680: Replace

```
        template<class T, class A>
        void vector<T,A>::reserve(int newalloc)
        {
                if (newalloc<=space) return;    // never decrease allocation
                vector_base<T,A> b(alloc,newalloc);     // allocate new space
                for (int i=0; i<sz; ++i) alloc.construct(&b.elem[i],elem[i]); // copy
                for (int i=0; i<sz; ++i) alloc.destroy(&telem[i]); // destroy old
                swap< vector_base<T,A> >(*this,b); // swap representations
        }
```

  with

```
        template<class T, class A>
        void vector<T,A>::reserve(int newalloc)
        {
                if (newalloc<=this->space) return;      // never decrease allocation
                vector_base<T,A> b(this->alloc,newalloc);       // allocate new space
                for (int i=0; i<this->sz; ++i) this->alloc.construct(&b.elem[i],this->elem[i]); // copy
                for (int i=0; i<this->sz; ++i) this->alloc.destroy(&this->elem[i]); // destroy old
                swap< vector_base<T,A> >(*this,b); // swap representations
        }
```

  (I forgot - and forgot to mention - that "this->" is required when accessing a member of a template base class of a template class. Not all compilers enforce this rule. When you re-test, re-test on every compiler you use.)
- (+4/19/09) pg 680: add after the last sentence (ending "we wanted."): Similarly, we have to explicitly use **this->** when we refer to a member of the base class **vector_base<T,A>** from a member of the derived class **vector<T,A>**, such as **base<T,A>::reserve()**.
- (+10/22/2010) pg 680: replace

```
        for (int i=0; i<this->sz; ++i)
                this->alloc.construct(&b.elem[i],this->elem[i]); // copy
```

  with

```
        uninitialized_copy(b.elem,&b.elem[this->sz],this->elem); // copy
```

- (+6/2/2010) pg 680: Replace the "When we exit **reserve()** ..." with "We use the standard-library function **uninitialized_copy** to construct copies of the elements from **b** because it correctly handles throws from an element copy constructor and because calling a function is simpler than writing a loop. When we exit **reserve()**, the old allocation is automatically freed by **vector_base**'s destructor if the copy operation succeeded. If instead that exit is caused by the copy operation throwing an exception, the new allocation is freed."
- (+6/2/2010) pg 680: s/such as **vector_base<T,A>::reserve()**./such as **vector<T,A>::reserve()**./
- (+3/29/09) pg 681: s/operator[]./operator[]()./ twice.
- (+6/12/09) pg 681: s/Define template<class T> ostream& operator<<(ostream&, vector<T>&)/Define template<class T> istream& operator>>(istream&, vector<T>&)/
- (+1/16/2010) pg 681: Revised chapter 19 drill:

```
        1-9 unchanged
        10. Replace set() with an S<T>::operator=(const T&). Hint: Much simpler than §19.2.5.
        11. Provide const and non-const versions of get();
        12-13 unchanged
        14: Bonus: Define input and output operators (>> and <<) for vector<T>'s.
                For both input and output use a { val, val, val } format.
                That will allow read_val() to also handle the S< vector<int> > variable.
```

- (+4/20/09) pg 682: Replace exercise 1 with this formulation: "Write a template function **f()** that adds the elements of one **vector<T>** to the elements of another; for example, **f(v1,v2)** should do **v1[i]+=v2[i]** for each element of **v1**."

• (+7/9/2010) pg 683: Add to exercise 8: Hint: look up "placement new" and "explicit call of destructor" in a complete C++ reference.
• (+7/5/2010) pg 683: exercise 11: s/Give the **counted_ptr** an initial value for the **T**/Let **counted_ptr**'s constructor take an argument to be used as the initial value of the **T** elements/

Chapter 20

- (+3/11/09) pg 687: replace

  ```
  if (h<jack_data[i])
          jack_high = &jack_data [i]; //save address of largest element
  ```

  by

  ```
  if (h<jack_data[i]) {
          jack_high = &jack_data[i];      //save address of largest element
          h = jack_data[i];               //update "largest element"
  }
  ```

- (+3/11/09) pg 687: replace

  ```
  if (h<(*jill_data)[i])
          jill_high = &(*jill_data)[i]; //save address of largest element
  ```

  by

  ```
  if (h<(*jill_data)[i]){
          jill_high = &(*jill_data)[i]; //save address of largest element
          h = (*jill_data)[i];            //update "largest element"
  }
  ```

- (+5/9/09) pg 687: s/The function from_jill()/The function get_from_jill()/
- (+3/11/09) pg 688: replace

  ```
  if (h<v[i]) jill_high = &v[i];
  ```

  by

  ```
  if (h<v[i]) {
          jill_high = &v[i];
          h = v[i];
  }
  ```

- (+4/19/09) pg 688: s/high = p;/{ high = p; h = *p; }/
- (+3/9/09) pg 690: s/high = &v[i];/{ high = &v[i]; h = v[i]; }/
- (+7/9/2010) pg 696: s/my code/my algorithms/
- (+8/27/09) pg 699 in the picture it should read val instead of Elem
- (+3/31/09) pg 700: s/is central in/is central to/
- (+) pg 701: s/{ return val; }/{ return curr->val; }/
- (+7/9/2010) pg 701: s/Basically, the **List iterator**/Basically, the **list iterator**/
- (+11/15/2010) pg 701: s/iterator(Link* p)/iterator(Link<Elem>* p)/
- (+7/9/2010) pg 703: s/We use this kind of test systematically with STL algorithms./We use testing the return value against end() - indicating ``not found'' - systematically with STL algorithms./
- (+6/3/2010) pg 705: In the diagram add a . (dot) after "This is the start of a very long document"
- (+3/31/09) pg 706: s/String and processing data/Storing and processing data/
- (+6/3/2010) pg 706: In the diagram add a . (dot) after "This is the start of a very long document" and "This is a new line".
- (+7/22/09) pg 707 s/while (is>>ch)/while (is.get(ch))/
- (+7/25/09) pg 707: Add before "return is;":

  ```
  if (d.line.back().size()) d.line.push_back(Line());   // add final empty line
  ```

- (+6/3/2010) pg 707: delete the comment "// line[i] is the ith line"
- (+7/25/09) pg 708: s/an empty **Document** to have one/a **Document** to end with a/
- (+7/25/09) pg 708: Improved ++:

  ```
  Text_iterator& Text_iterator::operator++()
  {
          ++pos;                          // proceed to next character
          if (pos==(*ln).end()) {
                  ++ln;                           // proceed to next line
                  pos = (*ln).begin();    // bad if ln==line.end(); so make sure it isn't
          }
          return *this;
  }
  ```

- (+) pg 709: replace

  ```
  Text_iterator end() // one beyond the last line
          { return Text_iterator(line.end(), (*line.end()).end()); }
  ```

  with

```
        Text_iterator end()       // one beyond the last line
        {
                list<Line>::iterator last = line.end();
                --last; // we know that the document is not empty
                return Text_iterator(last, (*last).end());
        }
```

- (+7/25/09) pg 709: s/The call **advance(n)** moves an iterator **n** elements forward/A call **advance(p,n)** moves an iterator **p n** elements forward/
- (+6/2/2010) pg 709: s/one beyond the last line/one beyond the last character of the last line/
- (+6/2/2010) pg 709: To match std::advance(), replace the erase_line() definition with

```
        void erase_line(Document& d, int n)
        {
                if (n<0 || d.line.size()-1<=n) return;
                list<Line>::iterator p = d.line.begin();
                advance(p,n);
                d.line.erase(p);
        }
```

- (+6/2/2010) pg 710: To match std::advance(), replace the advance() definition with

```
        template<class Iter> void advance(Iter& p, int n)
        {
                while (0<n) {++p; --n; }
        }
```

- (+6/2/2010) pg 710: Replace the second sentence of the first paragraph with: "In fact, for a **vector** called **v**, **p=v.begin; advance(p,n); *p=x** is roughly equivalent to **v[n]=x**."
- (+7/25/09) pg 711: Add a **++first;** to **find_text()**:

```
        Text_iterator find_text(Text_iterator first, Text_iterator last, const string& s)
        {
                if (s.size()==0) return last;    // can't find an empty string
                char first_char = s[0];
                while (true) {
                        Text_iterator p = find(first,last,first_char);
                        if (p==last || match(p,last,s)) return p;
                        ++first;        // look at the next character
                }
        }
```

- (+10/19/2010) pg 711: s/++first;/first = ++p;/
- (+6/3/2010) pg 712: vector: s/involve moving characters/involve moving elements/
- (+6/3/2010) pg 712: string: s/are not guaranteed to be contiguous/are guaranteed to be contiguous/
- (+7/13/2011) pg 712:s/delete()/erase()/
- (+4/20/09) pg 714: s/3rd element/4th element/ twice
- (+4/20/09) pg 714: s/4th element/5th element/ twice
- (+8/27/09) pg 716 s/we implement vector<T>::erase()/we implement vector<T,A>::erase()/
- (+6/20/09) pg 716: s/destroy(&*pos)/destroy(&*(end()-1)))
- (+8/17/2010) pg 716: s/Elem*/T*/ (in the comment)
- (+) pg 717: replace

```
    if (size()==capacity()) reserve(2*size()); // make sure we have space
```

  with

```
    if (size()==capacity()) reserve(size()==0?8:2*size()); // make sure we have space
```

- (+) pg 717: Replace

```
        *(begin()+offset) = val;     // "insert" val
        return pos;
```

  with

```
        *(begin()+index) = val;     // "insert" val
        return pp;
```

  As I said: always re-test even after the slightest change - in this case I renamed local variables, but didn't re-test until later.
- (+) pg 717: s/elem+space/elem+sz/ in the text; the code is correct
- (+10/4/2010) pg 717: s/the index of the element to be erased/the index at which the element is to be inserted/
- (+11/5/2010) pg 718 : s/typedef T* const_iterator/typedef const T* const_iterator/
- (+) pg 721: s/The C++ Programming Library/The C++ Programming Language/
- (+11/2/2010) pg 722: s/**built-in array**/built-in array/

- (+11/4/2010) pg 723: s/randomaccess/random-access/
- (+4/26/09) pg 724: s/copy (Iter f1, Iter1 e1, Iter2 f2)/Iter2 copy (Iter1 f1, Iter1 e1, Iter2 f2)/
- (+4/26/09) pg 724: s/just like the standard library copy./and returns **f2+(e1-f1)** just like the standard library copy./
- (+5/5/09) pg 726: Add as the second to last sentence of the postscript: "In fact, this underestimates the saved effort because many algorithms take two pairs of iterators and the pairs need not be of the same type (e.g. see exercise 6)."

- (+6/3/2010) pg 726 s/19. Define a range-checked vector for list/19. Define a range-checked iterator for list/

Chapter 21

- (+5/5/09) pg 733: s/finds the first odd value./finds the first odd value. Note that when you pass a function as an argument, you don't add () to its name because doing so would call it./
- (+7/19/09) pg 733: s/larger_than_42(int x)/larger_than_42(double x)/
- (+7/19/09) pg 734: s/int v_val;/double v_val;/
- (+7/19/09) pg 734: s/larger_than_v(int x)/larger_than_v(double x)/
- (+6/4/2010) pg 739: s/a+b+c+d/i+a+b+c+d/
- (+6/4/2010) pg 739: s/a*e+b*f+c*g+d*h/i+a*e+b*f+c*g+d*h/
- (+4/25/09) pg 742: s/all of those sums/all of those products/
- (+6/4/2010) pg 745: s/21.6.1 Maps/21.6.1 **map**/
- (+8/17/2010) pg 749: s/probably a pointer to/similar to a pointer to/ (in the comment)
- (+6/4/2010) pg 749: s/be a **Node***/be similar to a **Node***/
- (+8/27/09) pg 750 s/(Grape,100) (Kiwi,2345)/(Grape,2345) (Kiwi,100)/
- (+6/4/2010) pg 753: s/implementing large **map**s./implementing large maps./
- (+5/5/09) pg 754: The dotted line in the unordered_map lookup picture should go throught the 3rd from last hash table element.
- (+5/5/09) pg 755: s/If you really need unordered_map/If your C++ implementation doesn't provide unordered_map/
- (+6/4/2010) pg 755: s/21.6.5 Sets/21.6.5 **set**/
- (+2/23/2010) pg 756: add a comment to the definition of **inventory**:

```
        set<Fruit,Fruit_order> inventory;  // use Fruit_order(x,y) to compare Fruits
```

- (+7/22/09) pg 762: add a comment: struct No_case { // is lowercase(x) < lowercase(y)?
- (+7/22/09) pg 763: add a test for ==:

```
                        if (yy<xx) return false;                // y<x
                }
                if (x.length()==y.length()) return false;       // x==y
                return true; // x<y (fewer characters in x)
        }
};
```

- (+6/4/2010) pg 763: s/search()/binary_search()/ 3 times
- (+6/14/2010) pg 763: s/";see §21.6.4"/" because its cost is O(log2(N));see §21.6.4"/
- (+6/14/2010) pg 766: s/multi_map/multimap/

Chapter 22

- (+8/27/09) pg 778: s/22.2.8/22.2.6/
- (+6/3/2010) pg 778: s/If we ..., we'd also/We could also/
- (+6/3/2010) pg 778: s/vector<pair<string,Value_type>>/vector< pair<string,Value_type> >/ twice
- (+8/27/09) pg 782 s/Simula (22.2.6)/Simula (22.2.4)/
- (+6/15/09) pg 784 s/1948/1949/ *(to be precise, it ran on May 6, 1949)*
- (+4/28/09) pg 776: s/often achieved though/often achieved through/

- (+6/3/2010) pg 806: s/better checking of templates,/improved language facilities, such as/

Chapter 23

- (+8/27/09) pg 814 s/regular expressions (23.3-10)/regular expressions (23.5-10)/
- (+6/3/2010) pg 815: s/C-style **string**/C-style string/
- (+6/3/2010) pg 815: for insert: s/a character, a **string**, or a C-style string/a **string** or a C-style string/
- (+6/3/2010) pg 815: for append: s/**s.append(pos,x)** Insert **x** after **s[pos]**;/**s.append(x)** Insert **x** after the last character of **s**;/
- (+6/3/2010) pg 815: for append: s/a character, a **string**, or a C-style string/a **string** or a C-style string/
- (+6/3/2010) pg 815: s/ operations move characters/operations may move characters/

- (+7/5/2010) pg 815: replace the erase line with

```
	s.erase(pos)    Remove trailing characters from s starting with s[pos]. s's size becomes pos
	s.erase(pos,n)  Remove n characters from s starting at s[pos]. s's size becomes max(pos,size-n)
```

- (+7/5/2010) pg 815: for find: s/npos/string::npos/
- (+6/4/2010) pg 816: s/a stringstream/an ostringstream/
- (+7/5/2010) pg 818/s/read arg into stream/write arg into stream/
- (+6/4/2010) pg 819: Replace the first sentence started on the page with: "So if we try to read an **int** from a **string**, both **lexical_cast<int>("123")** and **lexical_cast<int>("123 ")**will succeed, but **lexical_cast<int>("123.5")** will not because of that last **.5**."
- (+8/27/09) pg 825 s/called **senders**/called **sender**/
- (+2/05/2010) pg 825: s/We use the empty string to indicate that an address wasn't found.//
- (+6/4/2010) pg 825: s/a **map**/a map/ three times
- (+11/5/2010) pg 824: s/sender.equal_range("John Doe");/sender.equal_range("John Doe <jdoe@machine.example>");/
- (+4/28/2010) pg 825: s/sender.equal_range("John Doe");/sender.equal_range("John Doe <jdoe@machine.example>");/
- (+6/15/09) pg 828: s/isletter/isalpha/
- (+6/15/09) pg 828: s/is a **letter**/is a letter/ (no code font)
- (+5/6/09) pg 829: s/12 June 2007/5 June 2007/
- (+5/5/09) pg 830: s/Maddoc's/Maddock's/
- (+8/27/09) pg 841 s/an English vowel or a ^/a space, a ^, or an English vowel/
- (+8/21/09) pg 846: s/so we get ^([\w]+)/so we get ^[\w]+/
- (+5/5/09) pg 847: s/regex_match/regex_match()/ 3 times in the text
- (+5/5/09) pg 847: s/regex_search/regex_search()/ 3 times in the text
- (+5/5/09) pg 849: s/lexical_cast/lexical_cast()/
- (+5/5/09) pg 849: s/the from_string function/from_string()/
- (+8/21/09) pg 849: s/Maddoc/Maddock/
- (+8/27/09) pg 850, s/23.7.7/23.7/
- (+6/15/09) pg 851: s/.../----/

Chapter 24

- (+8/27/09) pg 856 s/to a floating-point number/to a floating-point variable/
- (+8/27/09) pg 858 s/complex (24.8)/complex (24.9)/
- (+10/1/09) pg 858 s/In **<limits>**, **<limits.h>**/In **<limits>**, **<climits>**, **<limits.h>**/
- (+6/14/2010) pg 862: s/on the course site/on the book support site/
- (+8/27/09) pg 864 s/print the elements of a row by row/print the elements row by row/
- (+) pg 865: s/10/8/ three times
- (+4/8/09) pg 865: s/from the a[i]/from a[i]/ twice
- (+) pg 870: s/7/1/ twice on the swap_rows line
- (+) pg 870: s/9/2/ twice on the swap_rows line
- (+6/3/2010) pg 870: s/(i,j,k)/(i,j)/
- (+6/14/2010) pg 872: s/the elements from/the rows from/ twice
- (+8/27/09) pg 875: s/))) pivot_row = j/)))pivot_row = k/
- (+7/9/2010) pg 876: s/(we should have used elim_with_partial_pivot)/(elim_with_partial_pivot could do better in many cases)/
- (+11/16/09) pg 877 s/24.5.3/24.5.4/
- (+6/23/09) pg 879: s/rand_int/randint/ four times
- (+5/8/09) pg 883: s/Matrix<int> b(10)/Matrix<int> b(100)/

- (+6/3/2010) pg 885: s/Write a double() function/Write a triple() function/
- (+6/3/2010) pg 885: s/double/triple/ four times

Chapter 25

- (+4/8/09) pg 890 s/machine engineers/engineers/
- (+4/29/09) pg 908: s/f(vd);/fv(vd);/
- (+5/5/09) pg 910: s/a.sz!=a/a.sz!=sz/
- (+4/22/09) pg 910: add "return true;" as the last statement of assign()
- (+10/4/2010) pg 913: s/the dot in p[i].draw()/the dot in a[i].draw()/
- (+5/11/09) pg 914: s/it takes to produces/it takes to produce/
- (+) pg 915: s/<cost Q>/<const Q>/
- (+) pg 915: for the argument of f(): s/s2/s0/
- (+4/9/09) pg 919: s/Note that a 0 is "shifted in" from beyond bit 7/Note that a 0 is "shifted in "from beyond bit 0

(the least significant bit)/
- (+4/9/09) pg 919: s/Note that a 0 is "shifted in" from beyond bit 0/Note that two 0s are "shifted in" from beyond bit 7 (the most significant bit)/
- (+10/4/2010) pg 922: s/(unsigned) **int**/(signed) **int**/
- (+10/4/2010) pg 925: s/chars on our machine are unsigned (c behaves as uc and differs from sc)/chars on our machine are signed (c behaves as sc and differs from uc)/
- (+8/27/09) pg 927. Correct the code example lines to:

```
        unsigned char right = val&0xff; // rightmost (least significant) byte
        unsigned char left = val>>8;            // leftmost (most significant) byte
```

- (+8/27/09) pg 928 s/x becomes 24/y becomes 24/
- (+4/19/09) pg 930: s/ also know as/also known as/
- (+) pg 938: s/25.6.2/25.6.3/
- (+5/13/09) pg 946: s/(have found useful)?/(have found useful)./
- (+5/13/09) pg 947: s/25.23-4/25.4.3-4/

Chapter 26

- (+7/9/2010) pg 958: s/int test_all()/int test_all(istream& is)/
- (+7/9/2010) pg 958: s/(cin>>t)/(is>>t)/
- (+) pg 959: replace commas with spaces in the test input example.
- (+5/15/09) pg 956: s/different element at end/different element at beginning/ for the line with lots of 1s
- (+7/9/2010) pg 959: s/test_all()/test_all(ifstream("my_tests.txt"))/
- (+6/23/09) pg 960: s/rand_int/randint/ four times
- (+6/23/09) pg 960: 1/24.7:/24.7 and std_lib_facilities.h:/
- (+7/5/2010) pg 960: s/the average distance between elements is spread /the average distance between elements is uniformly distributed in [0 :spread)/
- (+6/23/09) pg 961: s/rand_int/randint/ two times
- (+7/5/2010) pg 962: s/look not just at the formal parameter/ look not just at the formal parameters/ (missing s)
- (+7/7/2010) pg 965: s/(vector<int>& v)/(const vector<int>& v)/
- (+5/16/09) pg 972: s/from the "main program":/from the "main program"):/
- (+7/7/2010) pg 976: s/if (ln.color() != i*100)/if (ln.color() != Color(i*100))/
- (+7/7/2010) pg 976: s/if (ln.style() != i*5)/if (ln.style() != Line_style(i*5))/
- (+7/7/2010) pg 977: s/if (2<last�first)/if (2 <= last-first)/
- (+4/19/09) pg 980: s/earlier tests runs/earlier test runs/
- (+) pg 980: s/++n/++i/
- (+5/18/09) pg 984: s/different element at end/different element at beginning/ for the line with lots of 1s
- (+7/7/2010) pg 984: Drill #2 : replace commas with spaces in the test input example.
- (+10/4/09) pg 985: in 1: s/26.2.1/26.3.2.1/
- (+8/27/09) pg 985: in 3: s/the exercise in sec26.2.1/exercise 1/

- (+7/7/2010) pg 986 s/(O(n^2)/O(n*n)/

Chapter 27

- (+) pg 989: s/ARM C/ARM C++/
- (+) pg 989: s/C++99/C99/
- (+) pg 991: K&R is published by Prentice Hall, not Addison-Wesley.
- (+7/13/2011) pg991: s/1997 version./1998 version./
- (+3/9/09) pg 996: s/The declaration of g() specifies no argument./The declaration of h() specifies no argument./
- (+11/16/09) pg 996: s/This does not mean that g()/This does not mean that h()/
- (+) pg 1006: s/(0)/(7)/
- (+) pg 1006: s/S1 a;/S2 a;/
- (+) pg 1006: s/S1* r = (S1*)&a;/S1* r = (S1*)&b;/
- (+4/9/09) pg 1013: s/std:string/std::string/
- (+7/19/09) pg 1014 : s/change 'd' in a to/change 'd' in aa to/
- (+5/20/09) pg 1019: s/write from/write to/
- (+12/28/2010) pg 1019: s/open fn for writing/open fn2 for writing/ (in comment)
- (+8/31/09) p1027: s/initialize *p/initialize *lst/
- (+1/26/2010) pg 1027: s/malloc(sizeof(struct List*));/malloc(sizeof(struct List));/
- (+6/26/09) pp1030-1031: move the definition **int count = 0;** to the top of the function **main()**. *In C89, declarations cannot come after statements in a block*.
- (+6/18/09) pg 1031: s/"Hello, World"/"Hello World"/

Appendix A

- (+) pg 1049: s/bound to a reference/bound to a local or namespace reference/
- (+5/22/09) pg 1054: /A12/A.12/
- (+5/22/09) pg 1058: /Try to convert v into a **D***/Try to convert **p** into a **D***/
- (+4/23/09) pg 1064: Add to the first prargraph under the table: "The comparisons <, <=, >, >= can also by used for pointers of the same type into the same object or array."
- (+4/23/09) pg 1064: s/and casting (type conversion)./, casting (type conversion)/, and comparison (==, !=, <, <=, >, and >=)./
- (+) pg 1065: s/sizeof(a)==sizeof(a[0])*max==sizeof(int)*max/sizeof(a); that is, sizeof(int)*max/
- (+8/31/09) p1075: s/sizof(int)/sizeof(int)/
- (+7/13/2011) pg1076: s/Date(string);/Date(const char*);/
- (3/9/09) pg 1079. Before "A.12.4.1 Virtual functions" insert a new paragraph: Members of a class can be initialized using the member initializer syntax **:member(initial_value)** (see A.12.3). Only members of the class itself, and not members of its base classes can be initialized this way. A base class can be initialized using the same syntax. For example:

```
struct Base {
        int mb;
        Base(int i) : mb(i) { }
};

struct Derived : Base {
        int md;
        Derived(int y) : md(y) { }                    // error: forgot to initilize Base
        Derived(int x, int y) : md(y), mb(x) { }        // error can't initialize base member from derived constructor
        Derived(int x, int y) :Base(y), md(x) { }       // ok
};
```

  An initializer used to initialize a base is class a base initializer or base-class initializer.
- (3/9/09) pg 1079. Before "A.12.4.1 Virtual functions" insert a new paragraph: Members of a class can be initialized using the member initializer syntax **:member(initial_value)** (see §A.12.3). Only members of the class itself, and not members of its base classes can be initialized this way. A base class can be initialized using the same syntax. For example:

```
struct Base {
        int mb;
        Base(int i) : mb(i) { }
};

struct Derived : Base {
        int md;
        Derived(int y) : md(y) { }                    // error: forgot to initialize Base
        Derived(int x, int y) : md(y), mb(x) { }        // error can't initialize base member from derived constructor
        Derived(int x, int y) :Base(y), md(x) { }       // ok
};
```

  An initializer used to initialize a base is class a base initializer or base-class initializer.

- (3/9/09) pg 1079. Before "A.12.4.1 Virtual functions" insert a new paragraph: Members of a class can be initialized using the member initializer syntax **:member(initial_value)** (see §A.12.3). Only members of the class itself, and not members of its base classes can be initialized this way. A base class can be initialized using the same syntax. For example:

```
struct Base {
        int mb;
        Base(int i) : mb(i) { }
};

struct Derived : Base {
        int md;
        Derived(int y) : md(y) { }                    // error: forgot to initialize Base
        Derived(int x, int y) : md(y), mb(x) { }        // error can't initialize base member from derived constructor
        Derived(int x, int y) :Base(y), md(x) { }       // ok
};
```

  An initializer used to initialize a base is class a base initializer or base-class initializer.

- (+) pg 1082: add bitfield sizes as on page 929.

Appendix B

- (+4/20/09) pg 1102: s/*p = --n;/*p++ = --n;/
- (+11/17/2010) pg 1103: s/Advance: like **p+=n**;/Like **p+=n**, but/
- (+11/17/2010) pg 1103: s/x=difference(p,q)/x=distance(p,q)/
- (+11/17/2010) pg 1103: s/Difference: like **q-p**; **difference()**/Like **q-p**, but **distance()**/
- (+2/23/2010) pg 1105: After the "Associative containers" table add: The ordered associative containers (map,

set, etc.) has an optional additional template argument specifying the type used for the comparator, e.g. **set<K,C>** uses a **C** to compare **K** values.

- (+5/28/09) pg 1124: s/The **make_pair** function/The **make_pair()** function/
- (+6/13/2010) pg 1131: s/**c isalpha()|isdigit()|ispunct()**/**isalpha(c)** or **isdigit(c)** or **ispunct(c)**/
- (+4/20/09) pg 1132: s/s1 or s2/s or s2/
- (+6/3/2010) pg 1132: for insert: s/a character, a **string**, or a C-style string/a **string** or a C-style string/
- (+6/3/2010) pg 1132: for append: s/**s.append(pos,x)** Insert **x** after **s[pos]**;/**s.append(x)** Insert **x** after the last character of **s**;/
- (+6/3/2010) pg 1132: for append: s/a character, a **string**, or a C-style string/a **string** or a C-style string/
- (+7/5/2010) pg 1132: replace the erase line with

```
s.erase(pos)    Remove trailing characters from s starting with s[pos]. s's size becomes pos
s.erase(pos,n)  Remove n characters from s starting at s[pos]. s's size becomes max(pos,size-n)
```

- (+7/5/2010) pg 1132: for find: s/npos/string::npos/

- (+4/20/09) pg 1139: s/for i>1;/for i>0;/
- (+4/19/09) pg 1141: s/value of s/value of p/ (twice)
- (+4/19/09) pg 1141: s/applies to s/applies to p/
- (+4/19/09) pg 1142: s/value of s is '%d'\n",x,s);/value of p is '%d'\n",x,p);/
- (+5/30/09) pg 1145: s/ungetch()s/ungetc()s/

Appendix C

- (+3/13/09) pg 1153: s/programming/Programming/ (in the URL)
- (+8/14/09) pg 1153: s/\Visual Studio 2005 Projects/\Visual Studio 2005\Projects/
- (+6/1/09) pg 1154: s/\Visual Studio 2005 Projects/\Visual Studio 2005\Projects/

Appendix E

- (+7/7/2010) pg 1162: s/for the window located at pw/for the window located at addr/
- (+7/7/2010) pg 1164: s/**Widget** has virtual function/**Widget** has virtual functions/
- (+7/7/2010) pg 1166: Improved detach():

```
void Window::detach(Shape& s)
   // guess that the last attached will be first released
{
   for (vector<Shape*>::size_type i = shapes.size(); 0<i; --i)
      if (shapes[i-1]==&s)
          shapes.erase(shapes.begin()+(i-1));
}
```

- (+4/6/09) pg 1167: s/int h, int w/int w, int h/
- (+7/7/2010) pg 1167: s/passed to it/passed to the **Vector_ref**/
- (+5/6/09) pg 1168: s/experiment number 17/experiment number 7/
- (+4/6/09) pg 1168: s/int h, int w/int w, int h/
- (+4/6/09) pg 1168: s/:Window(h,w,t)/:Window(w,h,t)/ *(and no, this doesn't change the effect of the program, but making the naming of variables consistent with the rest of the book makes the text less likely to confuse).*
- (+6/24/09) pg 1170: s/"mole"/"move"/

Glossary

- (+) pg 1173: add: **encapsulation** - protecting something meant to be private (e.g. implementation details) from unauthorized access.

Bibliography

- (+5/5/09) pg 1178: s/Maddoc/Maddock/

Index

- (+) pg 1186: s/initialization of, 417,543/initialization of, 470,543,1076,1079/
- (+8/27/09) pg 1190: s/**clock()**, 981-983/**clock()**, 981-983, 1147-1149/
- (+5/5/09) pg 1210: s/Maddoc/Maddock/
- (+3/17/09) pg 1222: s/reserve(), 651-652, 717, 1111/reserve(), 651-652, 667, 717, 1111/
- (+4/22/09) pg 1233: s/Wheeler, David, 108, 785, 930/Wheeler, David, 108, 785, 916, 930/

Thanks

Thanks to people who reported problems and often suggested remedies: Royer Alain, Tapani Alakiuttu, Petra Alm, Cassian Braconnier, Leo Carreon, Gunner Carstens, Steve Carter, Greg Cheong, Carlos E. Csiszer, J.P. Delport, Dustin Dettmer, Alessandro Gentilini, Yaakov Eisenberg, Richard Elderton, Eugene Feoktistov, Stephen Feyrer, Alessandro Gentilini, J.R. Halterlein, Christopher E. Harris, Carlos Hernandez, C.W.Holeman II, Jagadish R., Daniel Joffe, Nabil Kabbani, Christopher Keane, Prabhat Kiran, Chin-Yuan Kuo, Dirk Louis, Michael Leuschel, Teresa Leyk, Vinit Mahedia, Mark Millard, Sam Mitchell, Jaime Moreno, Akira Nakamura, Carlos Pantelides, Bob Parker, Bo Persson, Marcelo Pinto, Lukasz Piwko, Alec Ross, Robert J. Ryan, Iver Roessum David Spaniol, Alan Talbot, Dennis Tang, Kees Varekamp, Francisco Vidaller Art Werschulz, Kah-Yat Isaac Wong, Yongning Wu.