C++ by Dissection

by Ira Pohl

Addison-Wesley ISBN 0-201-78733-4

Enlisting the best-selling dissection method, this title teaches programming techniques and presents the C++ language using a sound and structured method. Working code is dissected with explanations to the student on each newly encountered programming element and idiom found in the code. The presentation assumes no programming background; both CS1 students and experienced programmers unfamiliar with Java will find a carefully structured presentation of the C++ language and its key programming concepts.

Features

- Assumes no previous programming experience
- Includes complete, executable code in every chapter, which is explained using the classic dissection method—a structured walkthrough of the code.
- Contains four to seven small code dissections per chapter, and then each chapter culminates with a larger, extended code dissection.
- Includes "Software Engineering" sections in each chapter that reinforce sound software engineering skills.
- Stresses programming style and methodology throughout, with careful explanation as to its importance and rationale.
- Provides common programming errors, end-of-chapter review questions, exercises, and summaries, as well as an Instructor's Manual with answers (for qualified college instructors), to make this an ideal text from which to learn and teach the material.
- Complete with a CD-ROM containing a compiler and an electronic version of the book that allows readers to search, take notes, and highlight right on their computer.

You can buy this book online through Barnes and Noble.

The C++ code examples in the book are available in several forms. The code is arranged in directories corresponding to the chapters of the book. To get to the unpacked version for any system, or to get individual program files, you can use the FTP directory (www.cse.ucsc.edu/~pohl/C++BD/). Files are also available on the CD-ROM which comes with the text.

FTP Directory

More information and the source code for the Java tio package used by the Java comparison examples is also available.

Additional Program Examples

This is a sample code provided by Uwe F. Mayer to test the distinction in catch signatures.

```
// Testing catch signatures const_catch.cpp
// Uwe F. Mayer - version 1.

#include <iostream>
#include <string>
using namespace std;

void print1(char *s) {
  cout << "print char * : " << s << endl;
}

void print2(const char *s) {
  cout << "print const char * : " << s << endl;
}
```

```
int main()
{
 char st[]="char * start";

 print1("const char * start"); // works!
 print2(st); // works!

 char s[]= "char * SOS";
  try {
    throw s;
  }
  catch(char * se) {
     cerr << "catch char * : " << se << endl;
  }
  catch(const char * se) {
     cerr << "catch const char * : " << se << endl;
  }
  try {
     throw "constant char * SOS";
  }
  catch(char * se) {
     cerr << "catch char * : " << se << endl;
  }
  catch(const char * se) {
     cerr << "catch const char * : " << se << endl;
  }
  try {
     throw s;
  }
  catch(const char * se) {
     cerr << "catch const char * : " << se << endl;
  }
  catch(char * se) {
     cerr << "catch char * : " << se << endl;
  }
  try {
     throw "constant char * SOS";
  }
  catch(const char * se) {
     cerr << "catch const char * : " << se << endl;
  }
  catch(char * se) {
     cerr << "catch char * : " << se << endl;
  }
}
```

Errata

Caveats:

Some standard libraries may have been modified since this book was written and the function prototypes may differ from what is available on your compiler. Check your compiler vendor for the correct prototypes.

Notations: p425 means page 425
+8 means 8 lines from top
-7 means 7 lines from bottom

If you encounter errata in this book not listed here, please contact Ira Pohl via email at pohl@cse.ucsc.edu with your errata. You will be cited in the correction if you are the first to report it.

---

p107 (Gyun Woo) line -5, The extensions of the source file names do not match with the above code. They should be changed to cpp:

g++ circle.cpp circle_main.cpp

---

p118 (Gyun Woo) in the figure, The variable name P should be lowercase p.

---

p127 (Gyun Woo) line -2, The input data is different to those in the output screen. Change
23 to 15 and 6 as follows:

Suppose we run this probram and input 23 and 6 when prompted.

---

p135 (Gyun Woo) line 3 and 4, The multiplication symbol has not printed correctly.

It should be printed as following:

line 3, s_1 X s_2 X ... X s_k
line 4, 3 X 5 elements, and c has 7 X 9 X 2 elements ...

where s_n means s subscripted by n and X denotes the multiplication symbol.

---

p149 (Gyun Woo) line -12 and -11, The meaning of the sentence can be more clear if we add a word 'direct':

... declaration, which lets a client have 'direct' access to all names ...

Rationale: The client can still access to all names using the scope resolution operator even though the using clause is not declared.

---

p177 (Gyun Woo) line 10, The word 'only' should be moved in front of the word 'declares':

The `class deck` only declares the class member functions; ...

Rationale: The class deck also declares the member variable, hence the original sentence:

The `class deck` declares only the class member functions; ...

is misleading.

---

p198 (Gyun Woo) line 8 and 9, The phrase 'by pfcn pF = &X::print' should be moved into the next sentence.

change
It is initialized by `pfcn pF = &X::print` to point at the member `X::visible`. The pointer `pf` is initialized to point at ...
to
It is initialized to point at the `member X::visible`. The pointer `pf` is initialized by `pfcn pF = &X::print` to point at ...

---

p200 (Gyun Woo) line -2, The font face of the word 'union' should be changed into the normal text font. It does not mean the keyword union.

---

p202 (Gyun Woo) line -6, The font face of the second 'print' should be changed into the code font:

means that `print` is a ....

---

p206 (Ira Pohl) Exercise 16

do not make isflush and isstraight be member functions, so no need for const at end of function

change

```
bool isflush(card my_hand[], int nc) const;

bool isstraight(card my_hand[], int nc) const;
```

to

```
bool isflush(card my_hand[], int nc);

bool isstraight(card my_hand[], int nc);
```

---

p272 (Ira Pohl) line 4, bad break of code. Move to the left so that the following is on one line:

```
<< greater(static_cast<rational>(i), z);
```

---

p299 (Gyun Woo) line -2, It seems the 'first type' indicates the type to be converted (one type) and the 'second type' indicates the other (second type), which is not true. Hence, the phrase 'the first type' should be changed to 'the second type', and vice versa. If these ordinals are used for indicating the argument position in the function header, it may be clear to mention it directly:

... provided the type of the first argument is at least as wide as the type of the second argument.

---

p300 (Gyun Woo) line 5, The operator '<=' should be changed to '<' : Change

```
if (sizeof(x) <= sizeof(y))
```

to

```
if (sizeof(x) < sizeof(y))
```

---

p305 (Gyun Woo) line 8, It is more clear to add the subject 'the template' following the word 'and':

... char and the template can be used ...

---

p319 (Gyun Woo) problem 11, line 2, The word 'generic' should be deleted to read:

Rewrite the following bubble sort using template:

---

p373 (Ira Pohl) exercises 5 and 6. These two exercises erroneously assume that the programmer can access the internals of the list STL class by asking to code member functions. Replace exercises 5 and 6 with the following text:

5. For list<T> , write the function

```
iterator insert(iterator w_it, T v);
```

which inserts v before w_it and returns an iterator pointing at the inserted element. (See Section 7.1, A Simple STL Example, on page 323.)

6. For list<T> , write the function

```
void erase(iterator w_it);
```

which erases the element pointed at by w_it. (See Section 7.1, A Simple STL Example, on page 323.)

---

p 415 (Ira Pohl) exercise 1 - There is no grad_student code. Change "For student and grad_student code, ... Use student::read to implement grad_student::read " to "For person and student code, ... Use person::read to implement student::read "

p415 (Ira Pohl) exercise 2 - main() is in Section 8.3, and pointers are incorrect. Change Section 8.1.1 .. on page 382 ... to Section 8.3 .. on page 383 .. Also need to change code from

```
reinterpret_cast<grad_student *>(ps) -> print();
dynamic_cast<student *>(pgs) -> print();
pgs -> student::print();
ps -> grad_student::print();
```

to

```
person* ptr_person;
student* ptr_student;

ptr_person = &abe;
ptr_student = &phil;

reinterpret_cast<student *>(ptr_person) -> print();
dynamic_cast<person *>(ptr_student) -> print();
ptr_person -> student::print(); //illegal use of person
ptr_student -> person::print();
reinterpret_cast<student *>(ptr_person) -> print();
```