【Effective CPP】Day6

Book	Effective C++
Author	
Summary	
Date	@2022/05/13

【Ch2】Constructors, Destructors, and Assignment Operators

Item 10: Have assignment operators return a reference to *this

One of the interesting thing about assignments is that we can chain them together:

```
int x, y, z;
x = y = z = 15;
```

The way this is implemented is that assignment returns a reference to its left-hand argument, and that's the convention we should follow.

```
class Widget {
public:
  ...
  Widget& operator=(const Widget& rhs) {
    ...
    return *this;
  }

  Widget& operator+=(const Widget& rhs) {
    ...
    return *this;
  }
};
```

This is just a convention. It is followed by all the built-in types and those in the standard library.

Things to Remember

Have assignment operators return a reference to `*this`.

Item 11: Handle assignment to self in operator=

An assignment to self occurs when an object is assigned to itself:

```
class Widget {...};
Widget w;
w = w;
```

If we try to manage resources ourself, we might fall into the trap of accidentally releasing a resource before when we are done with it.

```
class BitMap {...};

class Widget {
private :
  BitMap *pb;
};
```

Here is an implementation of operator= that looks reasonable but unsafe if applied to itself:

```
Widget& Widget::operator=(const Widget& rhs) {
  delete pb;
  pb = new BitMap(*rhs.pb);
  return *this;
}
```

The problem is that `rhs` and this could point to the same object. When they are, the `delete` not only destroys the bitmap for the current object, it destroys the bitmap for `rhs` as well.

We can do an identity check to prevent this error:

```
Widget& Widget::operator(const Widget& rhs) {
  if(&rhs == this)
    return *this;

  delete pb;
  pb = new BitMap(*rhs.pb);
  return *this;
}
```

This works, but we need to consider the overhead of adding extra evaluation.

A better and more general solution would be:

```
Widget& Widget::operator=(const Widget& rhs) {
  BitMap *pOrig = pb;
  pb = new BitMap(*rhs.pb);
  delete pOrig;
  return *this;
}
```

Things to Remember

Make sure `operator=` is well-behaved when an object is assigned to itself.