程序设计实习

郭炜　微博 http://weibo.com/guoweiofpku
http://blog.sina.com.cn/u/3266490431
刘家瑛 微博 http://weibo.com/pkuliujiaying

常量对象、常量成员函数和常引用

常量对象 (教材P194)

- 如果不希望某个对象的值被改变，则定义该对象的时候可以在前面加const关键字。

常量对象 (教材P194)

➢如果不希望某个对象的值被改变，则定义该对象的时候可以在前面加const关键字。

```
class Demo{
        private :
                int value;
        public:
                void SetValue() {       }
};
const Demo Obj;  // 常量对象
```

常量成员函数

- 在类的成员函数说明后面可以加const关键字，则该成员函数成为常量成员函数。

常量成员函数

➢在类的成员函数说明后面可以加const关键字，则该成员函数成为常量成员函数。

➢常量成员函数执行期间不应修改其所作用的对象。因此，在常量成员函数中不能修改成员变量的值（静态成员变量除外），也不能调用同类的非常量成员函数(静态成员函数除外）。

常量成员函数

```
class Sample
{
public:
        int value;
        void GetValue()  const;
        void func() { };
        Sample() { }
};
void Sample::GetValue() const
{
        value = 0; // wrong
        func(); //wrong
}
```

常量成员函数

```
class Sample
{
public:
        int value;
        void GetValue()  const;
        void func() { };
        Sample() { }
};
void Sample::GetValue() const
{
        value = 0; // wrong
        func(); //wrong
}
```

```
int main()  {
    const Sample o;
    o.value = 100; //err.常量对象不可被修改
    o.func(); //err.常量对象上面不能执行非常量成员函数
    o.GetValue(); //ok,常量对象上可以执行常量成员函数
    return 0;
} //在Dev C++中，要为Sample类编写无参构造函数才可以，Visual Studio 2010中不需要
```

常量成员函数的重载

- 两个成员函数，名字和参数表都一样，但是一个是const，一个不是，算重载。

常量成员函数的重载

```
class CTest {
        private :
                int n;
        public:
                CTest() { n = 1 ; }
                int GetValue()  const  { return n ; }
                int GetValue() { return 2 * n ; }
};
int main()  {
        const CTest objTest1;
        CTest objTest2;
        cout << objTest1.GetValue() << "," <<   objTest2.GetValue() ;
         return 0;
}
```

常引用

- 引用前面可以加const关键字，成为常引用。

不能通过常引用，修改其引用的变量。

常引用

- 引用前面可以加const关键字，成为常引用。

不能通过常引用，修改其引用的变量。

```
const int & r = n;
r = 5; //error
n = 4; //ok
```

常引用

对象作为函数的参数时，生成该参数需要调用复制构造函数，效率比较低。用指针作参数，代码又不好看，如何解决？

常引用

可以用对象的引用作为参数，如：

```
class Sample {
    …
};
void PrintfObj(Sample &  o)
{
    ……
}
```

常引用

可以用对象的引用作为参数，如：

```
class Sample {
    …
};
void PrintfObj(Sample &  o)
{
    ……
}
```

对象引用作为函数的参数有一定风险性，若函数中不小心修改了形参o，则实参也跟着变，这可能不是我们想要的。如何避免？

常引用

可以用对象的常引用作为参数，如：

```
class Sample {
    …
};
void PrintfObj( const Sample &  o)
{
    ……
}
```

这样函数中就能确保不会出现无意中更改o值的语句了。