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

构造函数(constructor)

基本概念(教材P179)

- 成员函数的一种
  - 名字与类名相同，可以有参数，不能有返回值(void也不行)
  - 作用是对对象进行初始化，如给成员变量赋初值
  - 如果定义类时没写构造函数，则编译器生成一个默认的无参数的构造函数
    - 默认构造函数无参数，不做任何操作

基本概念

- 如果定义了构造函数，则编译器不生成默认的无参数的构造函数
- 对象生成时构造函数自动被调用。对象一旦生成，就再也不能在其上执行构造函数
- 一个类可以有多个构造函数

基本概念

- 为什么需要构造函数：

1) 构造函数执行必要的初始化工作，有了构造函数，就不必专门再写初始化函数，也不用担心忘记调用初始化函数。

2) 有时对象没被初始化就使用，会导致程序出错。

```
class Complex {
        private :
                double real, imag;
        public:
                void Set( double r, double i);
}; //编译器自动生成默认构造函数

Complex c1;  //默认构造函数被调用
Complex * pc = new Complex;  //默认构造函数被调用
```

```
class Complex {
        private :
                double real, imag;
        public:
                Complex( double r, double i = 0);
};
Complex::Complex( double r, double i) {
        real = r; imag = i;
}
Complex c1;  // error, 缺少构造函数的参数
Complex * pc = new Complex; // error, 没有参数
Complex c1(2); // OK
Complex c1(2,4), c2(3,5);
Complex * pc = new Complex(3,4);
```

- 可以有多个构造函数，参数个数或类型不同

```
class Complex {
        private :
                double real, imag;
        public:
                void Set( double r, double i );
                Complex(double r, double i );
                Complex (double r );
                Complex (Complex  c1,  Complex  c2);
};
Complex::Complex(double r, double i)
{
        real = r; imag = i;
}
```

```
Complex::Complex(double r)
{
        real = r; imag = 0;
}
Complex::Complex (Complex  c1,  Complex  c2);
{
        real = c1.real+c2.real;
        imag = c1.imag+c2.imag;
}
Complex c1(3) , c2 (1,0), c3(c1,c2);
// c1 = {3, 0}, c2 = {1, 0}, c3 = {4, 0};
```

- 构造函数最好是public的，private构造函数不能直接用来初始化对象

```
class CSample{
        private:
                CSample() {
                 }
};
int main(){
        CSample Obj; //err. 唯一构造函数是private
        return 0;
}
```

构造函数在数组中的使用

```
class CSample {
        int x;
public:
        CSample() {
                cout << "Constructor 1 Called" << endl;
        }
        CSample(int n) {
                x = n;
                cout << "Constructor 2 Called" << endl;
        }
};
```

```
int main(){
    CSample array1[2];
    cout << "step1"<<endl;
    CSample array2[2] = {4,5};
    cout << "step2"<<endl;
    CSample array3[2] = {3};
    cout << "step3"<<endl;
    CSample * array4 =
            new CSample[2];
    delete []array4;
    return 0;
}
```

输出：

```
Constructor 1 Called
Constructor 1 Called
step1
Constructor 2 Called
Constructor 2 Called
step2
Constructor 2 Called
Constructor 1 Called
step3
Constructor 1 Called
Constructor 1 Called
```

构造函数在数组中的使用

```
class  Test {
      public:
            Test( int n) { }                  //(1)
            Test( int n, int m) { }       //(2)
            Test() { }                          //(3)
};
Test  array1[3] = { 1, Test(1,2) };
// 三个元素分别用(1),(2),(3)初始化
Test  array2[3] = { Test(2,3), Test(1,2) , 1};
// 三个元素分别用(2),(2),(1)初始化
Test  * pArray[3] = { new Test(4), new Test(1,2) };
//两个元素分别用(1),(2) 初始化
```