# 构造函数的种类

默认构造函数 （无参构造函数）

初始化构造函数（有参构造函数）

拷贝构造函数 （复制构造函数）[默认是浅拷贝]

**浅拷贝只是将值复制，本质还是一个，地址是相同的**
**深拷贝：拷贝的时候先开辟出和源对象大小一样的空间，然后将源对象里的内容拷贝到目标对象中去，这样两个指针就指向了不同的内存位置**

```
STRING( const STRING& s )
{
    //_str = s._str;
    //拷贝构造是初始化新对象，可以不使用delete
    _str = new char[strlen(s._str) + 1];
    strcpy_s( _str, strlen(s._str) + 1, s._str );
}
STRING& operator=(const STRING& s)
{
    if (this != &s)
    {
        //this->_str = s._str;
        //'=' 是在已有的对象上进行操作，从一个对象复制到另一个对象
        delete[] _str;
        this->_str = new char[strlen(s._str) + 1];
        strcpy_s(this->_str, strlen(s._str) + 1, s._str);
    }
    return *this;
}
```

**调用构造函数的三种情况**

1. **程序中需要新建立一个对象，并用另一个同类的对象对它初始化**
2. **当函数的参数为类的对象时**
3. **函数的返回值是类的对象**

```
#include "iostream"
using namespace std;

//定义一个Point类
```

```
class Point{
public:
    Point(int xx=0,int yy=0):x(xx),y(yy){}      //构造函数
    ~Point(){ };                                //析构函数
    Point(const Point &p);                  //复制构造函数
    int getX()const{return x;}
    int getY()const {return y;}
private:
    int x,y;//私有成员

};

Point::Point(const Point &p)
{
    x = p.x;
    y = p.y;
    cout << "Calling the copy constructor" <<endl;

}
//形参作为Point类对象的函数
void fun1(Point p)
{
    cout<< p.getX()<<endl;

}

//返回类的对象
Point fun2()
{
    Point a(1,2);
    return a;
}

int main()
{
    Point a(4);     //第一个对象A,该过程利用了重载，后面的y默认为0

    Point b(a);       //此时调用copy构造函数；情况1，用a初始化b，第一次调用copy构造函数
    cout << b.getX()<<endl;

    fun1(b);  //此时调用copy构造函数；类的对象在函数中为实参，第二次调用copy构造函数

    b = fun2();//此时调用copy构造函数；函数返回值为类的对象，第三次调用copy构造函数
    cout << b.getX()<<endl;
    return 0;
}
```

```
Point a(10);
Point b(20);
a = b; // 调用'='赋值运算符
```

转换构造函数 (将其他类型转换成本类)

## 一个类会默认生成那些函数

1. **无参的构造函数**
2. **拷贝构造函数**
3. **赋值运算符**
4. **析构函数（非virtual）**