拷贝构造函数

- **使用拷贝构造函数定义对象**

```
class Object
{
private:
    int i;
    int j;
public:
    Object (int i, int j)
    {
        this->i = i;
        this->j = j;
    }
    Object (const Object &other)     /* 拷贝构造函数，不写也可以*/
    {
        this->i = other.i;
        this->j = other.j;
    }

}

/* 定义对象 */
Object a;
Object b = a;
object c(a);

/* 动态创建对象 */
Object *p = new Object(a);

void Test (Object obj)
{

}

Test(a);    /* 函数传值调用 */
```

- **一般情况不建议编写拷贝构造函数（防止成员变量的遗漏）**
- **如果继承了一个父类，拷贝构造函数需要进行以下操作**

```
class parent
{
public:
    int B;
}

class Object
```

```
{
private:
    int i;
    int j;
public:
    Object (int i, int j)
    {
        this->i = i;
        this->j = j;
    }
    Object (const Object &other) : parent(other)    /* 拷贝构造函数继承父类 */
    {
        this->i = other.i;
        this->j = other.j;
    }

}
```

- **在没有拷贝构造函数的时候，拷贝的是数据的地址，实际上没有拷贝数据**

```
class Object
{
private:
    int m_size;
    char *m_buf;
public:
    Object (const char *str)
    {
        m_size = strlen(str) + 1;
        m_buf  = new char[m_size];
        strcpy(m_buf, str);
    }

    ~Object()                   /* 析构对象的时候，如果没有拷贝构造函数，那么就会同时析构同一块内存两次，因此报错 *
    {
        delete [] m_buf;
    }

    const char *get_text()
    {
        return m_buf;
    }

    Object (const Object &other)    /* 使用拷贝构造函数，分配另外一块内存保存数据，否则，程序会出错 */
    {
        m_size = strlen(str) + 1;
        m_buf  = new char[m_size];
        strcpy(m_buf, str);
    }
}
```

- **子类未实现拷贝构造时，会调用父类的拷贝构造函数（无论实现或未实现）**
- **子类一旦实现拷贝构造，则必须显示的调用父类的拷贝构造函数**
-
-
-
-
-