2020/05/12_C++_第6课_拷贝构造、CMyString

笔记本:	C++
创建时间:	2020/5/12 星期二 15:30
作者:	ileemi
标签:	CMyString/CString, 拷贝构造

拷贝构造和CMystring

拷贝构造

1、为啥需要拷贝构造因为一个类里面, 可能有些申请指针变量, 比如申请的内存指针. 如果没有拷贝构造, 就会只是简单的赋值, 这样就会发生一个问题 多个类对象操作了同一个内存. 造成你申请的这块内存值不可控, 为了避免类似的问题, 所以需要一个拷贝构造函数. 在拷贝构造函数里面重新申请内存, 这样不同的对象之间就实现了内存隔离, 数据独立.
没有拷贝构造叫做浅拷贝, 有构造函数的叫做深拷贝按照抽象世界来理解, 深拷贝像是同一类事物, 一个个独立的个体. 浅拷贝就是完全克隆

2、编译器怎么知道要触发拷贝构造编译器的逻辑很简单, 一次赋值操作, 编译器会先看看有没有与之匹配的拷贝构造函数, 如果有就调用, 如果没有就简单的赋值(完全克隆)
Test(buf); // 这行是传参
void Test(CBuff bufArg)
Test函数的参数对于函数作用域来说是一个独立的参数变量, 这个时候回触发对象赋值操作, 于是触发上述的对象拷贝过程

3、如果在类中没有定义拷贝构造函数，编译器会自行定义一个。如果类带有指针变量，并有动态内存分配，则它必须有一个拷贝构造函数

4、拷贝构造调用时机:

- 类对象传参
- 返回值为类对象
- 直接使用类对象实例化一个新对象

解决问题:

结构体->传参（值拷贝）
类->传参（值拷贝）

```cpp
#include <iostream>
using namespace std;

class CBuff
{
public:
    CBuff(const char* pData = nullptr, int nCount = 0)
    {
        cout << "CBuff::CBuff" << endl;
        if (pData == nullptr)
        {
            m_pBuff = nullptr;
            m_nCount = 0;
        }
        else
        {
            //申请新的内存，存入新的数据
            m_nCount = nCount;
            m_pBuff = (char*)malloc(m_nCount);
            memcpy(m_pBuff, pData, m_nCount);
        }
    }
    ~CBuff()
    {
        cout << "CBuff::~CBuff" << endl;
        //释放之前的内存
        if (m_pBuff != nullptr)
        {
            free(m_pBuff);
            m_pBuff = nullptr;
            m_nCount = 0;
        }
    }
    void SetData(const char* pData, int nCount)
    {
        //释放之前的内存
        if (m_pBuff != nullptr)
        {
            free(m_pBuff);
            m_pBuff = nullptr;
            m_nCount = 0;
        }
        //申请新的内存，存入新的数据
        m_nCount = nCount;
        m_pBuff = (char*)malloc(m_nCount);
        memcpy(m_pBuff, pData, m_nCount);
```

```
    }
    //显示数据
    const char* GetData()
    {
        return m_pBuff;
    }
private:
        int m_nVal;
        char* m_pBuff;
        int m_nCount;
};

//CBuff& bufArg
//不适用引用解决这个Bug
//此时的形参将创建一个新对象，创建新对象应该让其调用构造
void Test(CBuff bufArg)
{
        cout << bufArg.GetData() << endl;
}
int main()
{
        CBuff buf("Hello", 6);
        Test(buf);  //这里调用Test函数，将对象buf传递，这里会调用析构函
数

        return 0;   //程序运行到这里再次调用析构函数，程序崩溃
}
```

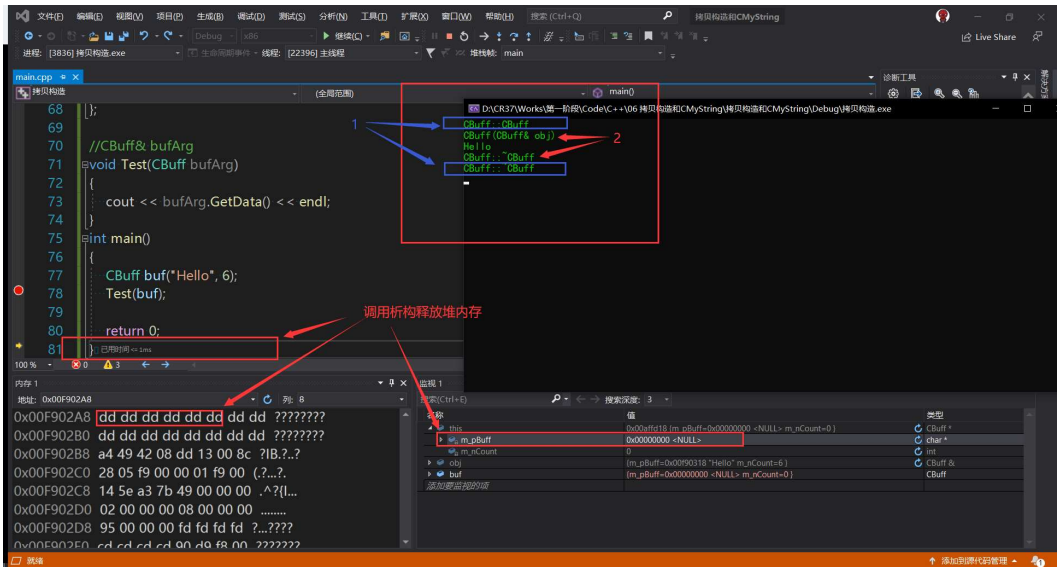
浅拷贝：存在两个指针指向同一块内存，重复释放内存的问题
解决Bug思路：
将浅拷贝转换成深拷贝，接管拷贝的过程

使用拷贝构造

```
//拷贝过程中obj的内容不进行修改，所以可以在类名前加const
CBuff(const Buff& obj)   //拷贝构造
{
    cout << "CBuff(CBuff& obj)" << endl;

    //深拷贝
    m_nCount = obj.m_nCount;
    //申请对应的堆空间
    m_pBuff = (char*)malloc(m_nCount);
    memcpy(m_pBuff, obj.m_pBuff, m_nCount);
}
```

同一个作用域内，类内的共有方法可以访问私有数据

两次调用构造函数（一次构造，一次拷贝构造），两次析构函数如下图所示：



总结

拷贝构造：

1、当我们没有实现拷贝构造的时候，编译器会默认的生成一个拷贝构造，其功能就是内存拷贝

```
void Test(CBuff bufArg);
Test(buf);
当没有拷贝构造的时候，传参相当于memcpy(&bufArg, &buf, sizeof(buf))
当有拷贝构造的时候，调用拷贝构造

这个Bug就是C++为了兼容C语言结构体的内存赋值存在的浅拷贝问题


//直接调用拷贝构造
CBuff buf0(buf);
//下面的了解即可
CBuff buf1 = buf;
CBuff buf2 = { buf };
CBuff buf3{ buf };
```

拷贝构造依然是众多重载函数中的一种，依然遵循重载函数的规则

构造函数和拷贝构造的区别：

区别再于参数的类型，当参数为类对象的时候，调用拷贝构造

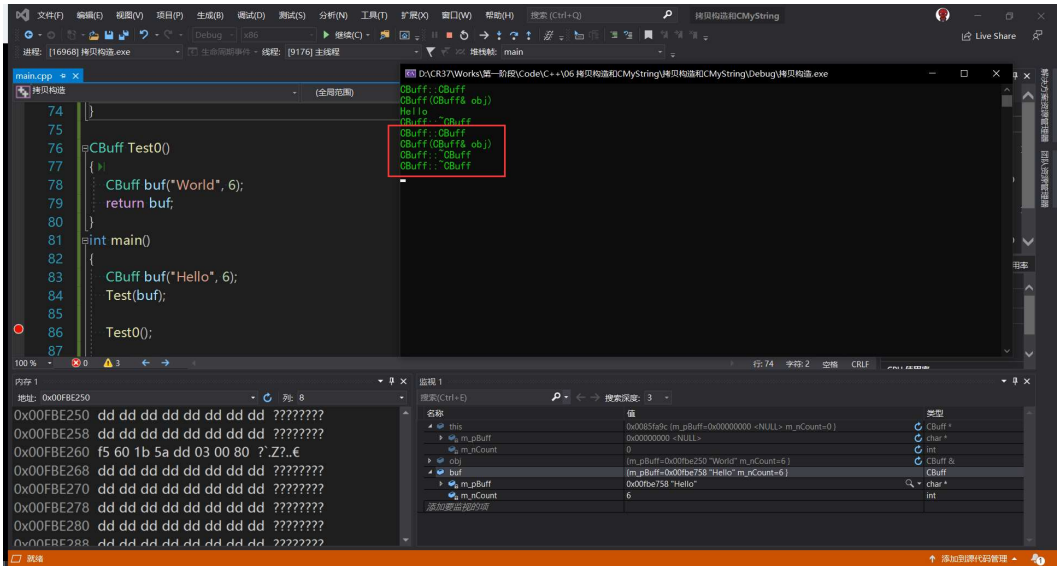
类对象作为返回值的时候也会出现拷贝构造

2、拷贝构造常见出现的时机

- 类对象传参 Test(buf);
- 定义一个类对象 CBuff buf0(buf);

- 类对象作为返回值时也会调用拷贝构造

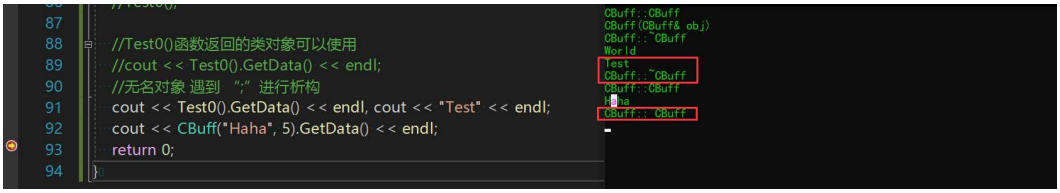
```
CBuff cTest()
{
        CBuff buf("World", 6);  //调用构造

    //先调用拷贝构造，然后调用析构，释放 CBuff buf("World", 6)
        return buf;
}
```



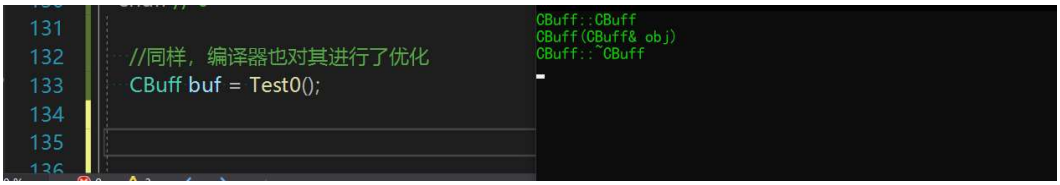
3、无名对象（临时对象）

- 类对象作为返回值
- 直接创建一个无名对象

```
CBuff cTest()
{
        CBuff buf("World", 6);  //调用构造

    //先调用拷贝构造，然后调用析构，释放 CBuff buf("World", 6)
        return buf;
}
int main()
{
    cTest();    //无名对象
    cout << cTest().GetData() << endl;

    //无名对象什么时候析构?
    //无名对象 遇到 “;” 进行析构

    //直接创建一个无名对象
    cout << CBuff("Haha", 5).GetData() << endl;
}
```




```cpp
//CBuff& bufArg

void Test(CBuff bufArg)
{
        cout << bufArg.GetData() << endl;
}
int main()
{
    // 这里，按照常理，应该有两次构造两次析构，
    //但是编译器对其进行了优化，直接拿无名对象的参数去构造形参
        Test(CBuff("Haha", 5));      //优化后这里有一次构造一次析构

    // 同样，编译器也对其进行了优化
    CBuff buf = Test0();
```



实现CMystring

对字符串进行相关的操作

定长字符串存储 ->变成存储

字符串相关操作:

- 初始化大小
- 获取字符串的长度
- 拼接
- 查找
- 替换
- 拷贝
- 提取子串
- //分割(涉及到数据结构)
- 去除空白字符(strip)
- //正则

功能细分:

- 查找
  正向查找 -> 从前向后找

反向查找 -> 从后向前找

- 提取子串
  左侧提取
  右侧提取
  中间提取
- 格式化字符串
  姓名-小白, 年龄-12, 性别-男
  str.Format("姓名-%s, 年龄-%s, 性别-%s", "小白", 12, 男);