

# 浅拷贝

名称	值	类型
&stu1	0x003bfa84 {m_nGender=1 m_nName=0x008e03c8 "zhangsan" }	CStuden
m_nGender	1	int
m_nName	0x008e03c8 "zhangsan"	char *
&stu2	0x003bfa74 {m_nGender=1 m_nName=0x008d0568 "1" }	CStuden
m_nGender	1	int
m_nName	0x008d0568 "1"	char *

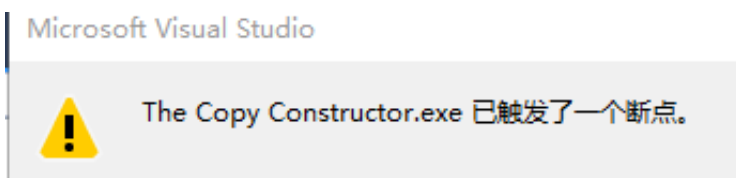
stu2拷贝给stu1

名称	值	类型
&stu1	0x004ffc24 {m_nGender=1 m_nName=0x005e0578 "zhangsan" }	CStuden
m_nGender	1	int
m_nName	0x005e0578 "zhangsan"	char *
&stu2	0x004ffc14 {m_nGender=1 m_nName=0x005e0578 "zhangsan" }	CStuden
m_nGender	1	int
m_nName	0x005e0578 "zhangsan"	char *

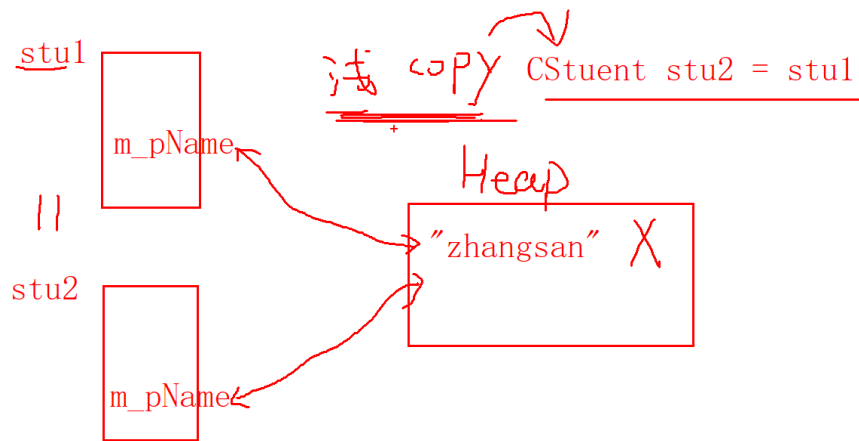
程序执行完毕，开始析构，把stu2申请的对象析构了，但是同时stu1申请的内存中的参数也被影响了，变成了无效的参数。这就造成了浅拷贝。

名称	值	类型
&stu1	0x004ffc24 {m_nGender=1 m_nName=0x005e0578 <字符串中的字符无效。> }	CStud
m_nGender	1	int
m_nName	0x005e0578 <字符串中的字符无效。>	char *
&stu2	0x004ffc14 {m_nGender=1 m_nName=0x00000000 <NULL> }	CStud
m_nGender	1	int
m_nName	0x00000000 <NULL>	char *

触发异常断点



浅拷贝图示



```

1  /*
2  浅拷贝
3  */
4
5  #include "stdafx.h"
6  #include <iostream>
7
8  class CStudent {
9  public:
10     //构造函数
11     CStudent(char* pName, int nGender = 1)
12         :m_nGender(nGender) //初始化参数列表
13     {
14         //申请堆空间，保存姓名
15         m_nName = new char[strlen(pName) + 1];
16         //拷贝姓名
17         strcpy_s(m_nName, strlen(pName) + 1, pName);
18         std::cout << "CStudent(int nGender)" << std::endl;
19     }
20     //析构函数
21     ~CStudent()
22     {
23         std::cout << "~CStudent()" << std::endl;
24         //释放申请的堆空间
25         if (m_nName != NULL) {
26             delete[] m_nName;
27         }
28         m_nName = NULL;

```

```
29     }
30
31 private:
32     int m_nGender; //性别
33     char* m_nName; //姓名
34 };
35
36 int main()
37 {
38     char szName[] = "zhangsan";
39     //申请一个CStudent类对象，名为stu1
40     //调用CStudent类的构造函数
41     CStudent stu1(szName, 1);
42     //申请一个CStudent类对象，名为stu2
43     //调用CStudent类的构造函数
44     CStudent stu2("1");
45     //将stu1对象拷贝给stu2对象
46     stu2 = stu1;
47
48     return 0;
49 }
```