C语言中memset函数

功能：将s所指向的某一块内存中的每一个字节的内容全部设置为ch指定的ASCII值，块的大小由第三个参数指定，这个函数通常为新申请的内存做初始化工作

用法：void *memset(void *s, char ch, unsigned n);

```
#include <string.h>
#include <stdio.h>
#include <memory.h>

int main(){
    char buffer[] = "hello world/n";
    printf("buffer befor set: %s/n", buffer);
    memset(buffer, '*', strlen(buffer));
    return 0;
}
```

memset 可以方便的清空一个结构类型的变量或数组

如：

```
struct sample_struct{

char csName[16];

int iSeq;

int iType;

}
```

对于变量

struct sample_struct stTest;

用memset清空的化：

memset(&stTest, 0, sizeof(struct sample_struct))

memset清空一个类的陷阱：

对有虚拟函数的类对象，绝不能使用**memset**来进行初始化操作

例如：

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

class parent
{
public:
    virtual void output();
};
void parent::output()
{
    printf("parent!");
}

class son : public parent
{
public:
    virtual void output();
};
void son::output()
{
    printf("son!");
}

int main()
{
    son s;
    memset(&s , 0 , sizeof(s));
    parent& p = s;
    p.output();
    return 0;
}
```

程序运行出错，原因是：

初始化obj的时候，将obj包含的指向虚函数表VTBL的指针也被清除了。包含虚函数的类对象都有一个指向虚函数表的指针，次指针被用于解决运行时和动态类型强制转换是虚函数的调用问题。该指针是被隐藏的，对程序员来说，这个指针也是不可存取的。当进行memset操作时，这个指针的值也要被初始化，这样一来，只要一调用虚函数，程序便会崩溃。

当类中有虚函数的时候，编译器会为类插入一个我们看不见的数据并建立一个表。这个表就是**虚函数表（vtbl）**，那个我们看不见的数据就是指向虚函数表的指针——**虚表指针（vptr）**。虚函数表就是为了保存类中的虚函数的地址。我们可以把虚函数表理解成一个数组，数组中的每个元素存放的就是类中虚函数的地址。当调用虚函数的时候，程序不是像普通函数那样直接跳到函数的代码处，而是先取出vptr即得到虚函数表的地址，根据这个来到虚函数表里，从这个表里取出该函数的地址，最后调用该函数。所以只要不同类的vptr不同，他对应的vtbl就不同，不同的vtbl装着对应类的虚函数地址，这样虚函数就可以完成它的任务了。