String class、Stack&Heap、new&delete and their implementation

3月有学过，十月再看印象不深了

1 String class

- [x] String class的手写 -> test_for_string.cpp

仍然出现一堆问题：

- Q1: , 为啥返回一个指针能够输出整个数组?
  A: 读指针( )操作能够读取从m_data开始直到读到'\0'结束。

```
int main(){
    char *p = "why so sad?";
    cout << p << endl;
    return 0;
}

output: wht so sad?
[warnning: char*与stirng类型存在类型转换]
```

- Q2:当参数有默认值时，是在声明还是定义写默认值，还是两处都写?
  A: 在任意一处写都可以（推荐在声明中写），但不能两处同时写，防止出现前后不一致的情况：

```
//声明
String(const char* cstr = 0);

//定义
inline
String::String(const char* cstr)
{
    if(cstr){
        m_data = new char[strlen(cstr) + 1];
        strcpy(m_data, cstr);
```

```
    }
    else{
        m_data = new char[1];
        *m_data = '\0';
    }
}

//若都写，则报错
error: default argument given for parameter 1 of ‘String::Str
previous specification in ‘String::String(const char*)’ here
```

- 传引用不要忘记，第一遍自己写的时候完全忘记了。
- String是class，其中的成员m_data为char型指针；不要与 搞混了，这里p是String类型的指针，它指向的String类型数组里存的是三个的指针成员m_data。成员函数中的this指针为String类型的指针，指向该成员函数所属的这个String类实体。其中this->m_data可以省略写为m_data，this默认存在可以不写，写出来不算错，但在参数列表中写出来算错。 返回该String类型实体，外面用String&来接。
- ☐ Q4:为什么 后s1不能执行 操作，且 输出为1?
- inline function如果定义在class里则不需要加inline关键词；而若在class外定义则需要显式添加inline关键词。
- 友元函数只是需要在直接操作类的private member(pm)时才需要，如果这些对pm的操作已经有public function包装了则不需要（区别参见complex与string的os output function）。
- 拷贝构造函数的传参如果不引用则会报错：

String class的实现内部均是宝藏和细节，需要多次反复编写记忆。

关键概念	关键概念的解释
The Big Three	带指针的类的拷贝构造、拷贝赋值与析构函数
self assignment	检测自我赋值
Three steps	拷贝赋值的三步骤
pointer output	指针所指内容的输出方式

2 Stack&Heap

> Stack空间是指定域中存放local object的一片内存，其内部空间并非动态分配，而是系统自动分配自动回收，函数调用时存放Stack中返回地址，局部变量等信息；例如

Heap空间是系统动态分配(dynamic allocated)的一块全局(global)内存空间，当调用new来动态分配时，分配的空间便属于Heap空间。例如 ，而动态分配的内存一定要通过delete对应去删除，即new array <=> new delete。

3 new&delete

```
/* new在编译器中的具体实现:
 *      先分配内存 (pointer mem) , 再ctor (data mem)
 */
complex* p = new complex(1,2);
```

在编译器中为:

```
1. complex* p;
2. void* mem = operator new(sizeof(complex)); //分配内存
3. p = static_cast<complex*>(mem); //强制类型转换
4. p->complex::complex(1,2); //构造函数
```

:operator new为function，其内部为C的malloc();
static_cast为类型转换;
complex::complex即为ctor。

```
/* delete在编译器中的具体实现:
 *      先dtor (data mem) , 再释放内存 (pointer mem)
 */
String* ps = new String("hello");
delete ps;
```

在编译器中为:

```
1. ps->String::~String(); //String::~String(ps);  //析构函数
2. operator delete(ps); //释放内存
```

:operator delete为function，其内部为free();
String::~String()即为dtor。

4 实际内存分配问题中 in VC：

Q:既然cookie中记录了长度，为什么还要两个cookie?

Q:new array 对应 delete array
A:delete array调用**n**次dtor，delete调用1dtor，发生内存泄露。