**Chapter 12 动态内存和类**

1. StringBad和String类的objects会存储
2. 一个指向字符串的pointer
3. 一个表示字符串长度的值（一般int）
4. 一个存number of objects
5. StringBad初始化的方式：listing12.2 strngBad.cpp
6. 析构函数的作用是释放constructor中new出来的空间
7. 由于指向字符串的指针在block结束时就会自动释放，所以不需要管，但是其中被new出来的空间并不会自动释放，需要用delete
8. 由于只有一个析构函数只能按顺序放进去，逆序释放。当出现释放顺序不匹配的时候就会出现问题。
9. 复制构造函数copy constructor

Eg. StringBad sailor = StringBad (sports); //constructor using sports

StringBad (const StringBad &); //隐式产生新的object，使用copy constructor初始化

1. Copy constructor的产生：如上用一个object初始化另一个object（产生物件的副本，特别是值传递，return对象都会产生copy constructor，但reference传递就不会），构造函数会自动使用copy constructor
2. 类别里constructor的原型： Class\_name (const Class\_name &)

String类别里copy constructor的原型：StringBad (const StringBad &)

1. 复制构造函数做什么

所有的参数都值复制

1. 产生的问题：
2. num\_strings不会++，但是析构函数会增加，导致构造函数和析构函数的不匹配，报错
3. 由于是值复制，就会导致两个pointer指向同一个字符串，然后其中一个在用完后被delete了，导致第二个pointer找不到地方，输出乱码出现各种错误并且存在potential danger
4. 修正：使用明确复制函数深度复制（deep copy）

意为不是复制字符串的位置建立新的指针，而是复制字符串然后把这个的新字符串的地址赋给成员指针，使得每个object都有一个完整的包含所有class物件的object

1. 赋值运算符=

类似于copy constructor，在前面的例子中出现“=”的时候可能会使用隐含的指定运算子，同样这只是成员对成员的复制，static的资料成员不受任何影响，解决方案也是定义深度复制的运算符

重载赋值运算符在定义的时候一定要delete之前的，再new，再定义之后的

但赋值构造函数则不用delete而可以直接new，再定义

与复制构造函数的区别：在定义的时候直接初始化用复制构造函数，定义后再初始化用赋值运算符

1. String类的一些功能，需要重载运算符
2. Return字符串的长度

Int length () const { return len; }

1. 字符串的比较

Friend bool operator< (const String &st, const String &st2);

Friend bool operator> (const String &st1, const String &st2);

Friend bool operator== (const String &st1, const String &st2);

1. 存取单个字符元

Char & operator [] (int i);

Const char & operator [] (int i) const;

其中[]内的东西将会是被引用的对象

而且由于operator[] ()是类别成员函数，所以修改数组内容时不会出错

1. Return 静态类别资料成员num\_strings的值

Static int Howmany(); //属于静态类成员函数

1. 新的预设函数应该是

String::String ()

{

Len = 0;

Str = new char [1];

Str [0] = ‘\0’ ;

}

其中不用new char的原因是析构函数的形式是delete [] str

1. 静态类成员函数
2. 在静态函数前加static修饰
3. 不能通过对象调用静态成员函数
4. 不能使用this指针
5. 只能使用类名和作用域解析运算符来调用
6. 只能使用静态数据成员，比如num\_strings
7. 一些注意事项（通常格式）

书P659-662

1. 函数返回对象
2. Return 一个const对象的reference

通常是为了提高效率，或者说要被return的就是传递过来的参数中的一个，比如return两个数中大的那一个

1. Return一个非const对象的reference

典型例子是重载运算符

1. Return一个对象
2. 一般出现在，被return的对象属于区域变量的数值，如果函数结束后，里面所有的变化都会释放，回到原来发出呼叫的函数，没有任何改变。Reference不会参考到任何对象，所以不能return一个reference，而是要return一整个对象
3. 会无可避免地使用到copy constructor
4. 常见于算数运算符重载
5. 返回const对象

为了防止两个对象赋值的时候出现问题，可以在return的时候return一个const形态的对象

1. 总结
2. 如果一个成员函数要回转一个区域变量的对象，return一个对象
3. 如果要回转的对象没有共有的复制构造函数，return一个对象的引用，比如ostream类
4. 如果某些成员函数可以回转一个对象或者一个引用，效率考虑，return一个引用，比如多载指定运算符
5. 指向对象的指针
6. 用法1：不重新new一个新的对象，只是指向原有的对象
7. 用法2：用string \*xxx = new string / new string (xxx) / new string (“xxx”)来建立一个指向新的内存的指针
8. -> 运算符用来存取指针类别的成员函数

Eg. If (sayings[i].length () < shortest -> length)

对象 指向对象的指针

1. 加\*，将指针变成对象
2. 定位new运算符 placement new
3. 如果两个new的内存重合，后面的会直接覆盖前面的
4. Delete不可以和placement new一起使用，但可以和new一起使用

故，必须明确呼叫某placement new的destructor，但首先要识别出要被释放空间的对象

Eg. Pc3->~ Justtesting (); //destroy object by pointed to by pc3