8. 堆、栈与内存管理

#include<iostream>

ostream& operator<<

cout通过接受指针来实现输出

os << str.get_c_str()

为什么要定义为全局函数？

不然会是 << os

所谓栈 stack 堆 heap

stack是存在于作用域内的一个内存空间

调用函数时

heap是由操作系统提供的global内存空间

⇓

new动态分配

stack objects的生命期

auto object

其析构函数会被自动调用

static object 其生命在调用结束仍存在

直到函数结束

global objects

即构造函数与析构函数的调用相时机

heap objects的生命期

memory leak

p所指的heap object仍然存在

但是p的生命周期结束了作用域之外看不到p

new：先分配memory，再调用ctor构造函数

Complex* pc = new Complex(1,2)

编译器：⇓

Complex *pc;

① void* mem = operator new(sizeof(Complex));

内部调用malloc(n)

② pc = static_cast<Complex*>(mem);

转型

③ pc->Complex::Complex(1,2)

隐藏this ⇓ 调用构造函数

Complex::Complex(pc,1,2)

this

delete：先调用dodtor，再释放内存

① String::~String(ps);

② operator delete(ps);

刚刚说的是malloc(), free()

如果new一个Complex

cookies

调试模式时的内存占用(VC下)

如果你分配的是数组

new Complex[3]

delete[] p