

Ch.6 类与对象

2018 / 11

类的典型构成

- 私（公）有成员
- 公（私）有函数
- 构造函数/析构函数
- 拷贝（构造）函数
- 一般函数
- 重载函数（=）
- 重写函数
- 重载运算符

对象

- 对象与类的关系
- 对象的生成
- 对象的销毁

对象生成的时机

- 什么时候产生?
 - `Object a;`
 - `Object b(a);`
 - `Object c = a;`
 - `Object * p = new Object();`

构造函数

- 什么时候运行？
 - 对象产生的时机（诞生即执行）
 - 对象产生的方式与空间分配（堆/栈）

销毁的时机

- 分配在栈上的对象，当栈销毁
- 堆上对象，调用`delete`主动销毁

析构函数

- 什么时候运行？
 - 对象销毁的时机（什么时候 销毁）
 - 对象销毁的方式（堆上对象/栈上对象）

对象生成的时机

- 什么时候产生?
 - Object a;
 - Object b(a);
 - Object c = a;
 - Object * p = new Object();

特别注意

- 使用拷贝构造 → 调用拷贝构造函数
- 使用 = 进行赋值
 - 对象已经存在 → 调用重载的 = 运算符
 - 对象尚未存在 → 调用拷贝构造函数

深入的话题

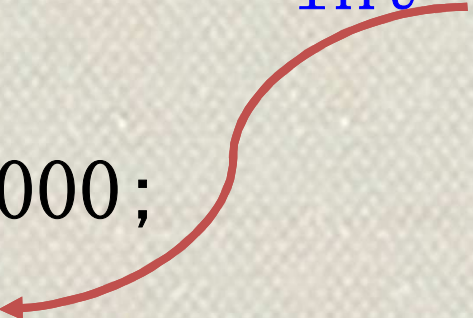
- 重载等号的高效写法

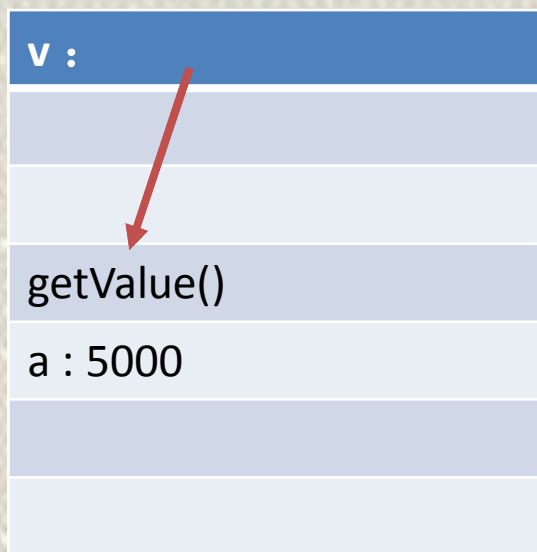
```
City & operator = (const City & o)
{
    name = o.name;
    zone = o.zone;
    return *this;
}
```


v 和 a 的关系

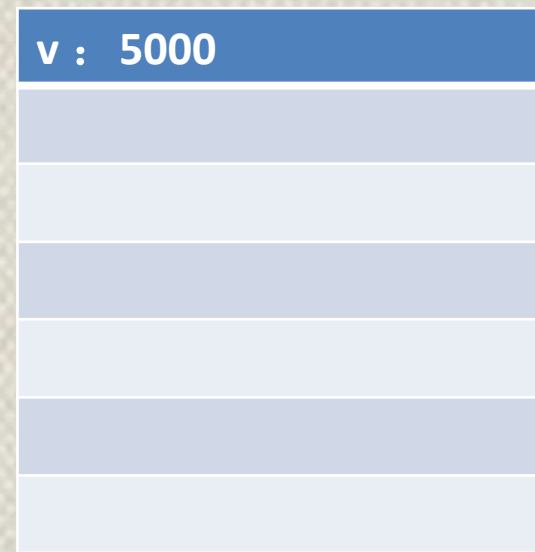
```
int getValue()  
{  
    int a = 5000;  
    return a;  
}
```

`int v = getValue();`





返回，栈销毁



a: City("Cambridge")
a.name: "Cambridge"
a.zone: "England"
b
b.name:
b.zone:
b.operator =(a) : &

a: City("Cambridge")
a.name: "Cambridge"
a.zone: "England"
b
b.name:"Cambridge"
b.zone:"England"

返回值和返回引用

- 多一次对象拷贝的过程
- 如果拷贝函数很复杂，那么会导致低效

改变对象的分配行为

- 对象只在堆上使用
- 对象只在栈上使用
- 栈上空间什么时候产生，什么时候销毁？
- 堆上空间什么时候产生？

如何限定

- 限定访问权限（`public/private`）
- 重载`new/delete`运算符，限定是否可以访问

栈上分配

- 重载new/delete，不进行实现（？）
- 将重载的new/delete限定为private

堆上分配

- 只能在堆上分配的话，限定不能调用构造函数即可
- 将所有的构造函数都私有化（那构造函数还有什么用）
- 构造函数供静态函数来进行调用即可

Q & A