

成员对象和封闭类

郭 炜 刘家瑛



北京大学



成员对象和封闭类

- **成员对象**: 一个类的成员变量是另一个类的对象
- **包含成员对象的类叫 封闭类 (Enclosing)**

```
class CTyre {    //轮胎类
    private:
        int radius; //半径
        int width;  //宽度
    public:
        CTyre(int r, int w):radius(r), width(w) { }
};

class CEngine { //引擎类
};
```



class CCar { //汽车类 → “封闭类”

private:

int price; //价格

CTyre tyre;

CEngine engine;

public:

CCar(int p, int tr, int tw);

};

CCar::CCar(int p, int tr, int w):price(p), tyre(tr, w){

};

int main(){

CCar car(20000,17,225);

return 0;

}



- 如果 CCar 类不定义构造函数, 则

CCar car; // error → 编译出错

- 编译器不知道 car.tyre 该如何初始化
- car.engine 的初始化没有问题: 用默认构造函数

- 生成封闭类对象的语句 → 明确“对象中的成员对象”

→ 如何初始化



封闭类构造函数的初始化列表

- 定义封闭类的构造函数时, 添加初始化列表:

```
类名::构造函数(参数表):成员变量1(参数表), 成员变量2(参数表), ...  
{  
...  
}
```

- 成员对象初始化列表中的参数

- 任意复杂的表达式
- 函数 / 变量 / 表达式中的函数, 变量有定义



调用顺序

- 当封闭类对象生成时,
 - S1: 执行所有成员对象的构造函数
 - S2: 执行 封闭类 的构造函数
- 成员对象的构造函数调用顺序
 - 和成员对象在类中的说明顺序一致
 - 与在成员初始化列表中出现的顺序无关
- 当封闭类的对象消亡时,
 - S1: 先执行 封闭类 的析构函数
 - S2: 执行 成员对象 的析构函数
- 析构函数顺序和构造函数的调用顺序相反



封闭类例子程序

```
class CTyre {  
    public:  
        CTyre() { cout << "CTyre contructor" << endl; }  
        ~CTyre() { cout << "CTyre destructor" << endl; }  
};  
  
class CEngine {  
    public:  
        CEngine() { cout << "CEngine contructor" << endl; }  
        ~CEngine() { cout << "CEngine destructor" << endl; }  
};
```



```
class CCar {  
    private:  
        CEngine engine;  
        CTyre tyre;  
    public:  
        CCar( ) { cout << "CCar contructor" << endl; }  
        ~CCar() { cout << "CCar destructor" << endl; }  
};  
  
int main(){  
    CCar car;  
    return 0;  
}
```

程序的输出结果是：

CEngine contructor
CTyre contructor
CCar contructor
CCar destructor
CTyre destructor
CEngine destructor