

# 面向对象

---

封装，继承，多态

## 封装

访问权限

- public : 类内可以访问，类外也可以访问
- protected : 类内可以访问，类外不可以访问，子类可以访问父类的保护内容
- private : 类内可以访问，类外不可以访问，子类不可以访问父类的私有内容 **struct默认访问权限是共有，class默认访问权限是私有**

## 对象的初始化和清理

- 构造函数：进行初始化，对象创建时调用(编译器自动调用)
  - 语法：类名()
  - 无返回值也不写void
  - 可有参数，可重载
- 可分为有参构造和无参构造，可以重载
  - 拷贝构造，语法：(const person &p) 调用有参构造函数时加括号，括号里放相应的数据
- 匿名对象(无类名)
  - 语法：person()
  - 当前行执行结束后系统会立即回收
  - 可以赋值给对象
- 析构函数：进行清理，对象销毁前调用(编译器自动调用)
  - 语法：~类名()
  - 无返回值也不写void
  - 不可有参数，不可发生重载
  - 对象会自动释放

## 构造函数调用规则

- c++编译器至少给一个类添加3个函数
  - i. 默认构造函数

## ii. 默认析构函数

## iii. 默认拷贝构造函数

- 用户定义有参构造函数，c++不再提供默认无参构造，但会提供默认拷贝构造
- 用户定义拷贝构造函数，c++不再提供其它构造函数

## 深拷贝和浅拷贝

- 浅拷贝：简单的赋值拷贝操作
  - 问题：堆区的内存重复释放
- 深拷贝：在堆区重新申请空间，进行拷贝操作
  - 用new，new的返回值是指针

## 初始化列表

语法：构造函数():属性1(值1)，属性2(值2)，...{} 属性可以看成成员变量

## 静态成员

- 在前面加上static关键字
- 静态成员变量
  - 所有对象共享一份数据
  - 在编译阶段分配内存
  - 类内声明，类外初始化
  - 访问方式
    - a. 通过对象访问
    - b. 通过类名访问 语法：类名::成员变量名
- 静态成员函数
  - 所有对象共享同一个函数
  - 静态成员函数只能访问静态成员变量(因为实例对象还没有被创建，无法访问非静态成员变量)

## c++对象模型

- 编译器会对每个空对象分配一个字节空间，是为了区分空对象占内存的位置
- 静态成员变量((非)静态成员函数)不属于类对象上，也就不占用字节空间.也就是只有非静态成员变量属于类的对象

## this指针

this指针指向被调用的成员函数所属的对象 用途：

- 形参和成员变量同名，可以用this指针区分
- 在类的非静态成员函数返回对象本身(return \*this) 要注意值传递和引用传递的区别 引用会使编译器只对一个值进行操作，值传递是创建一个副本并将要操作的数据拷贝到这个副本，对这个副本进行操作
- 在成员函数后面加const，会使得this指针指向的值也无法修改
- 若想修改，则需要加变量前加mutable关键字(常对象只能调用常函数)

## 友元(friend)

目的就是让一个函数或者类，访问另一个类中的私有成员

1. 全局函数做友元
  - 将函数连同参数复制进类中，加上friend关键字
2. 类做友元
  - 将类名放到要读取私有成员的类中，加上friend关键字
3. 成员函数做友元
  - 将成员函数放到要读取私有成员的类中，前面加上类名

## 运算符重载

语法：返回值类型(可以是类名) operator 重载的运算符(参数){函数体};

可以使用全局函数和成员函数来重载运算符

### 左移运算符重载

- 一般不使用成员函数重载左移运算符，无法实现cout在左边
- 而使用全局函数来重载 语法：

```
ostream &operator<<(ostream &cout, person &p)
{
    cout << p.m_A << p.m_B;
    return cout; //返回cout可以使得输出p时输出别的内容
}
```

### 重载++运算符

- 重载前置++运算符: 返回值类型 &operator++() {};
- 重载后置++运算符: 返回值类型 &operator++(int){};

## 继承

语法: `class 子类: public 父类`