**实验3：类与对象的进一步讨论（1）**

**姓名\_\_\_金宣成\_班级\_计科03\_\_学号\_202203151310\_\_**

* **请阅读此说明：实验3满分100分；做完实验后请按要求将代码和截图贴入该文档。然后将此文档、源代码文件（.hpp, .cpp）打包上传到学习通。**
* **上传包结构为：**

**压缩包命名为：实验3 学号+姓名**

**包含3个子文件夹（分别命名为：ex3-1，ex3-2，ex3-3） + 1个实验文档；**

**每个子文件夹中包含相关对应实验题的源代码**

**//为防止与系统名字冲突，可以将3-1 distance和3-2 clock首字母大写**

**1、位置类position：（30分）**

**设计并实现一个平面坐标系内的位置类Position。包含的基本数据成员有：横坐标，纵坐标；包含的基本成员函数有：设置位置；读取位置；判断第几象限；计算到源点的距离；计算到其他点的距离；计算经过源点到这个位置的直线的斜率；计算经过这个位置到其他点的直线的斜率；按坐标轴平移位置。其他成员函数功能可以自行补充。**

* **实验要求：**

**按照描述完成Position类的基本的设计和实现。将数据成员设计为私有(private)成员；将成员函数设计为公有(public)成员。并通过以下测试程序。**

#include "position.hpp"

#include <iostream>

using namespace std;

int main() {

Position a; //默认坐标位置为原点

Position b(4,5),c(b),d(0,0),e(1); //b的位置为x轴值4，y轴值5

cout<<"a(0,0):"; a.show();

a.set(5,15);

cout<<"a(5,15):"; a.show();

cout<<"b(4,5):"; b.show();

b.set(-4.5,6.7);

cout<<"b(-4.5,6.7):"; b.show();

cout<<"c(4,5):"; c.show();

c.set(-10, -100);

cout<<"c(-10,-100):"; c.show();

cout<<"d(0,0):"; d.show();

d.set(20.5,5.5);

cout<<"d(20.5,5.5):"; d.show();

cout<<"e(1,0):"; e.show();

e.set();//默认为原点

cout<<"e(0,0):"; e.show();

cout<<Distance(a,b)<<endl;

cout<<Distance(c)<<endl;//默认求与原点的距离

cout<<a.slope()<<endl; //与原点构成直线的斜率

cout<<a.slope(d)<<endl; //与d构成直线的斜率

a.move(3);//沿x轴平移

cout<<"a(8,15):"; a.show();

b.move(-4,5);

cout<<"b(-8.5,11.7):"; b.show();

c.move(0,6);//沿y轴平移

cout<<"c(-10,-94):"; c.show();

return 0;

}

* **实验提交：**

**将完整的源代码和测试截图 粘贴在下面。**

* **源代码粘贴处：将Position类的声明和定义实现附在此处即可**
* //----------------------------position.hpp-------------------------

class position

{

private:

    double x = 0;

    double y = 0;

public:

    position(double a,double b);

    position(double a);

    position();

    void show();

    void set(double a ,double b);

    void set();

    double Distance() const;

    double Distance(const position& a) const;

    double slope();

    double slope(const position & a) const;

    void move(double a);

    void move(double a,double b);

    ~position();

};

* //----------------------------position.cpp-------------------------

#include <iostream>

#include "position.hpp"

#include <cmath>

using namespace std;

position::position(double a = 0, double b = 0)

{

    x = a;

    y = b;

}

position::position(double a = 0){

    x = a;

}

position::position(){

}

void position::set(double a,double b){

    x = a;

    y = b;

}

void position::set(){

    x = 0;y = 0;

}

void position::show(){

    cout << "x = " << x << "   " << "y = " << y << endl;

}

double position::Distance(const position& a) const{

    return sqrt((a.x-x) \* (a.x-x) + (a.y-y)\*(a.y-y));

}

double position::Distance() const{

    return sqrt(x\*x + y\*y);

}

double position::slope(){

    return y/x;

}

double position::slope(const position & a) const{

    return (a.y - y)/(a.x - x);

}

void position::move(double a){

    x += a;

}

void position::move(double a ,double b){

    x += a;

    y += b;

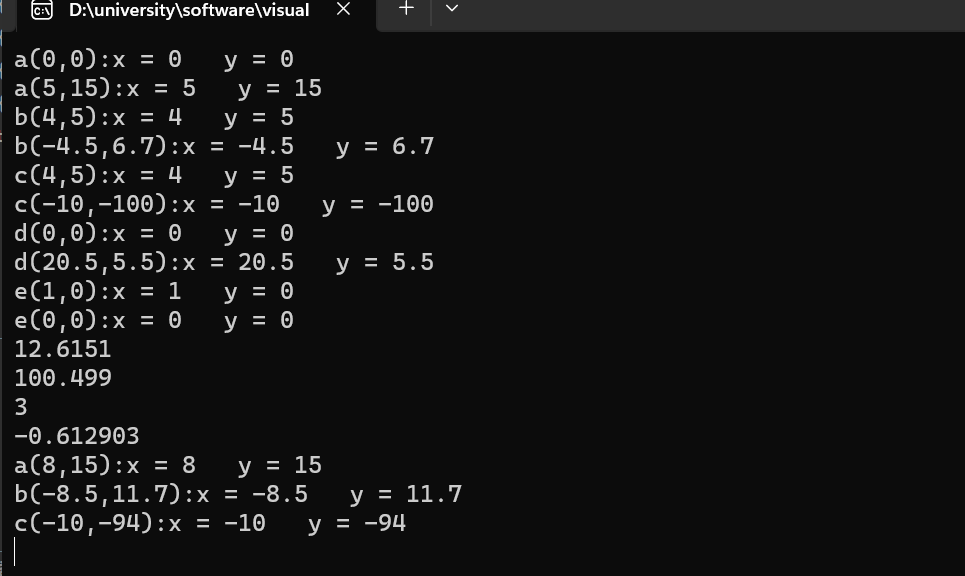
}

position::~position()

{

}

* **程序测试截图：**

****

**2、对象的指针和数组、拷贝构造（40分）**

//----------------------------box.hpp-------------------------

#pragma once

#include <iostream>

using namespace std;

class box {

public:

box(int hv = 10, int wv = 10, int lv = 10) :h(hv), w(wv), len(lv) {

cout << "I\'m a box:" << h << "-" << w << "-" << len << endl;

}

box(const box& other) {

h = other.h;

w = other.w;

len = other.len;

cout << "I\'m a copy box:" << h << "-" << w << ":" << len << endl;

}

void reset() {

h = 10;

w = 10;

len = 10;

}

~box() {

cout << "A box says: Bye-bye." << h << "-" << w << "-" << len << endl;

}

private:

int h, w, len;

};

//----------------------------clock.hpp-------------------------

#pragma once

#include <iostream>

using namespace std;

class Clock {

public:

Clock(int hv = 1, int mv = 1, int sv = 1) :h(hv), m(mv), s(sv) {

cout << "I\'m a clock:" << h << ":" << m << ":" << s << endl;

}

clock(const Clock& other) {

h = other.h;

m = other.m;

s = other.s;

cout << "I\'m a copy clock:" << h << ":" << m << ":" << s << endl;

}

void reset() {

h = 0;

m = 0;

s = 0;

}

~Clock() {

cout << "A clock says: Bye-bye." << h << ":" << m << ":" << s << endl;

}

private:

int h, m, s;

};

//-----------------------demo.hpp--------------------------------

#pragma once

#include <iostream>

#include "box.hpp"

#include "clock.hpp"

using namespace std;

class Demo {

public:

Demo(int xv = 0, int hv = 0, int mv = 0, int sv = 0) :x(xv, mv, sv), y(hv, mv, sv) {

cout << "Demo-Constructor1." << endl;

}

Demo(const box& abox, const Clock& aclock) :y(aclock), x(abox) {

cout << "Demo-Constructor2." << endl;

}

Demo(const Demo& other) {

cout << "Demo-Copy Constructor." << endl;

}

Clock getclock() { return y; }

box getbox() { return x; }

~Demo() {

cout << "Demo-Destructor." << endl;

}

public:

box x;

Clock y;

};

//-----------------------maintest.cpp-----------------------

#include<iostream>

#include "clock.hpp"

#include "demo.hpp"

using namespace std;

void fun1(Clock xclock) {

xclock.reset();

}

Demo fun2(Demo xDemo) {

xDemo.getclock().reset();

return xDemo;

}

int main()

{

Clock aclock(8, 34, 45), bclock(aclock);

box abox, bbox(20, 30);

Demo ademo(1, 2, 3, 4), bdemo(ademo), cdemo(abox, aclock), ddemo(3, 4);

Clock \*pclock;

box \*pbox;

Demo \*pDemo;

pclock = new Clock(6, 7, 8);

pbox = new box(cdemo.getbox());

pDemo = new Demo[2]{ Demo(6,7,8,9),Demo(bbox,bclock) };

fun1(\*pclock);

fun2(pDemo[1]);

delete pclock;

delete[]pDemo;

delete pbox;

return 0;

}

* **实验要求：**

**理解和掌握类类型的对象数组和对象指针的使用。理解拷贝构造函数应用的不同场景。**

**5分❶按照给出例程的文件命名各个部分并完成程序的装配。**

**ex3\_2 project : box.hpp+clock.hpp+demo.hpp+maintest.cpp**

**5分❷先阅读程序, 先写出你理解的程序运行结果。**

**再运行程序，验证你的结论。**

**结合程序实际执行理解输出结果，注意总结自己理解不到位的知识点并及时巩固。能够分辨拷贝构造函数应用的三种场景：声明、函数参数值传递、函数返回值传递。 可以使用注释部分代码的方法，帮助自己理解程序的工作结果。**

**10分❸对于Demo类型的对象，其数据成员x,y 的构造顺序是怎样的？（先获得x的空间还是先获得y的空间？）数据成员的这个顺序是由什么决定的？注意观察初始化列表的顺序，和它有关系吗？请给出你的结论，并想办法证实它。**

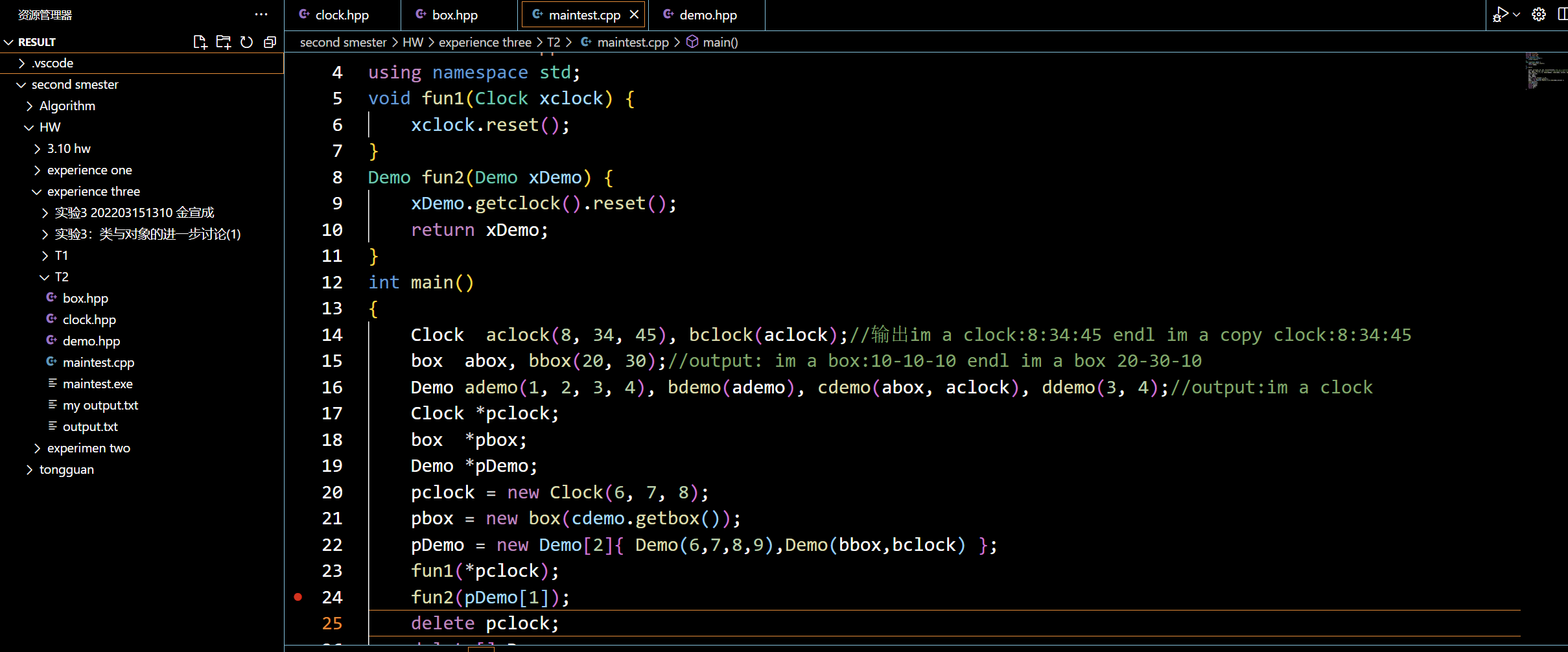
**5分❹给出程序中所有与 cdemo 这个对象相关的输出。**

**5分❺ 理解静态对象和动态对象构造析构顺序的差异。**

**10分❻ 给出和指针pDemo相关的输出。**

* **实验提交：**

**❶给出实验的项目装配截图 贴在此处**

****

**❷你模拟的程序输出结果：**

**//--------------------------------预测输出的分析----------------------------------------//**

**im a clock:8:34:45//aclock**

**im a copy clock:8:34:45//bclock**

**//r15**

**im a box:10-10-10//abox**

**im a box 20-30-10//bbox**

**//r16**

**im a box:1-3-4**

**im a clock:2:3:4**

**Demo-Constructer1.//ademo**

**Demo-Copy Constructor.//bdemo**

**im a copy clock:8:34:45**

**im a copy box:10-10-10**

**Demo-Constructer2.//cdemo**

**im a box:3-0-0**

**im a clock:4:0:0**

**Demo-Constructor1.//dDemo**

**//r20,这样也行？？？？目前猜测是在pclock的地方申请地址，储存的是一个Clock(6,7,8)**

**im a clock:6:7:8//pclock**

**//r21,应该是一个拷贝，拷贝的source是cdemo 的box**

**im a copy box:10-10-10//pbox**

**//r22,产生了两个demo**

**im a box:6-8-9**

**im a clock:7:8:9**

**Demo-Constructer1.//pdemo[0]**

**im a copy clock:8:34:45**

**im a copy box:20-30-10**

**Demo-Constructer2.//pdemo[1]**

**//fun1 output**

**im a copy clock:6:7:8**

**A clock says: Bye-bye.0:0:0//没有改pclock的值**

**//fun2 output,用了拷贝构造,但demo的拷贝构造就写了一个输出，没有值，应该就是默认值？**

**//但是这个函数类型也是Demo，有两个拷贝构造？那这里析构的顺序是怎么样的啊？**

**//猜测是先析构box，因为头文件在前面:)**

**Demo-Copy Constructor.**

**Demo-Copy Constructor.**

**A box says: Bye-bye.10-10-10**

**A clock says: Bye-bye.0:0:0**

**//r25**

**A clock says: Bye-bye.6:7:8**

**//r26，我记得是先析构pDemo[1].顺序就是构造反过来**

**A box says: Bye-bye.20-30-10**

**A clock says: Bye-bye.8:34:45//pDemo[1]的析构**

**A clock says: Bye-bye.7:8:9**

**A box says: Bye-bye.6-8-9**

**//r27**

**A box says: Bye-bye.10-10-10**

**//r28接下来开始自动析构部分**

**//顺序是dDemo、cDemo、bDemo、aDemo、bbox、abox、bclock、aclock**

**//对着上面的构造反过来抄就好了应该**

**A clock says: Bye-bye.4:0:0**

**A box says: Bye-bye.3-0-0//dDemo**

**A box says: Bye-bye.10-10-10**

**A clock says: Bye-bye.8:34:45//cDemo**

**A clock says: Bye-bye.1:1:1**

**A box says: Bye-bye.10-10-10//bDemo,这里不是很确定，上面假设了先box构造，那这就应该是先clock？**

**A clock says: Bye-bye.2:3:4**

**A box says: Bye-bye.1-3-4//aDemo**

**A box says: Bye-bye.20-30-10//bbox**

**A box says: Bye-bye.10-10-10//abox**

**A clock says: Bye-bye.8:34:45//bclock**

**A clock says: Bye-bye.8:34:45//aclock**

**//-------------------------------最终预测输出------------------------------------//**

**im a clock:8:34:45**

**im a copy clock:8:34:45**

**im a box:10-10-10**

**im a box 20-30-10**

**im a box:1-3-4**

**im a clock:2:3:4**

**Demo-Constructer1.**

**Demo-Copy Constructor.**

**im a copy clock:8:34:45**

**im a copy box:10-10-10**

**Demo-Constructer2.**

**im a box:3-0-0**

**im a clock:4:0:0**

**Demo-Constructor1.qd**

**im a clock:6:7:8**

**im a copy box:10-10-10**

**im a box:6-8-9**

**im a clock:7:8:9**

**Demo-Constructer1.**

**im a copy clock:8:34:45**

**im a copy box:20-30-10**

**Demo-Constructer2.**

**im a copy clock:6:7:8**

**A clock says: Bye-bye.0:0:0**

**Demo-Copy Constructor.**

**Demo-Copy Constructor.**

**A box says: Bye-bye.10-10-10**

**A clock says: Bye-bye.0:0:0**

**A clock says: Bye-bye.6:7:8**

**A box says: Bye-bye.20-30-10**

**A clock says: Bye-bye.8:34:45**

**A clock says: Bye-bye.7:8:9**

**A box says: Bye-bye.6-8-9**

**A box says: Bye-bye.10-10-10**

**A clock says: Bye-bye.4:0:0**

**A box says: Bye-bye.3-0-0**

**A box says: Bye-bye.10-10-10**

**A clock says: Bye-bye.8:34:45**

**A clock says: Bye-bye.1:1:1**

**A box says: Bye-bye.10-10-10**

**A clock says: Bye-bye.2:3:4**

**A box says: Bye-bye.1-3-4**

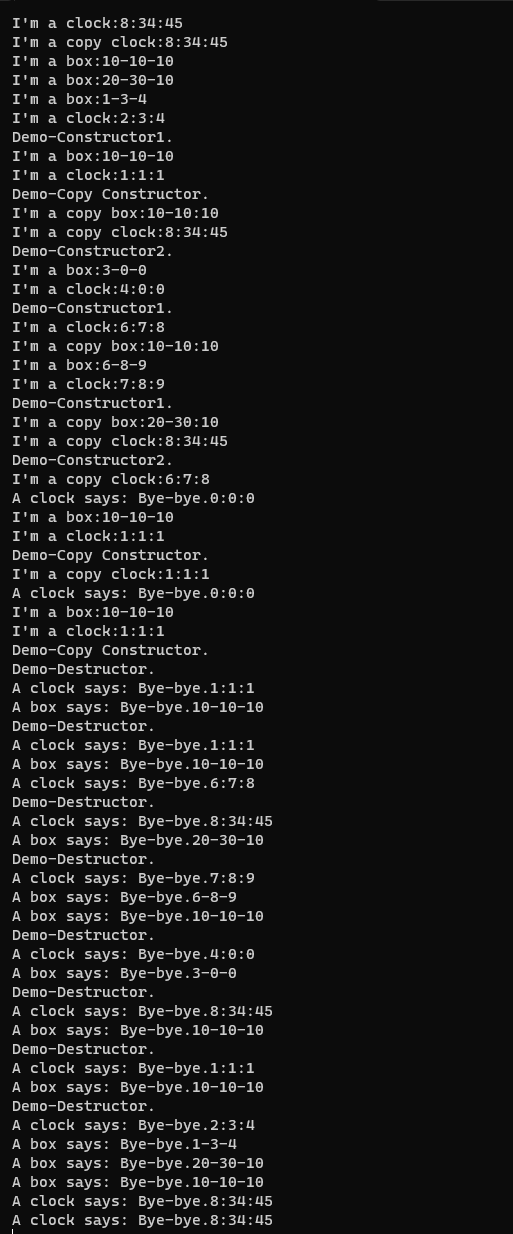
**A box says: Bye-bye.20-30-10**

**A box says: Bye-bye.10-10-10**

**A clock says: Bye-bye.8:34:45**

**A clock says: Bye-bye.8:34:45**

**实际程序输出结果（可截图）：**



**知识总结：**

**//---------------------------------对正确答案的分析-----------------------------------//**

**I'm a clock:8:34:45**

**I'm a copy clock:8:34:45**

**//------------clock构造------------------------------------//**

**I'm a box:10-10-10**

**I'm a box:20-30-10**

**//-----------box构造---------------------------------------//**

**I'm a box:1-3-4**

**I'm a clock:2:3:4**

**Demo-Constructor1.**

**I'm a box:10-10-10**

**I'm a clock:1:1:1**

**Demo-Copy Constructor.//----------这里错了，Demo拷贝构造里面虽然就一个输出，但是只要生成了，就会有im a。。。。**

**I'm a copy box:10-10:10**

**I'm a copy clock:8:34:45**

**Demo-Constructor2.//后面是冒号，没看清。顺序反了，不应该看后面的初始化列表的顺序，要看前面申明时括号里的？这里不太懂**

**I'm a box:3-0-0**

**I'm a clock:4:0:0**

**Demo-Constructor1.**

**//----------------Demo构造------------------------------//**

**I'm a clock:6:7:8//r20**

**I'm a copy box:10-10:10//r21**

**I'm a box:6-8-9**

**I'm a clock:7:8:9**

**Demo-Constructor1.//Demo[0]**

**I'm a copy box:20-30:10**

**I'm a copy clock:8:34:45**

**Demo-Constructor2.//Demo[1]这里的顺序跟上面一样**

**I'm a copy clock:6:7:8**

**A clock says: Bye-bye.0:0:0//fun1 的输出，但没有真的改pclock的值**

**//---------------接下来时fun2的输出，很复杂----------------//**

**I'm a box:10-10-10**

**I'm a clock:1:1:1**

**Demo-Copy Constructor.//构造了xDemo**

**I'm a copy clock:1:1:1//Clock getclock使用了拷贝构造函数，拷贝的是xDemo中的Clock y**

**A clock says: Bye-bye.0:0:0//析构了上一条拷贝出来的getclock**

**I'm a box:10-10-10**

**I'm a clock:1:1:1**

**Demo-Copy Constructor.//要return了，Demo fun2开始构造，把xDemo拷贝出来**

**Demo-Destructor.**

**A clock says: Bye-bye.1:1:1**

**A box says: Bye-bye.10-10-10//return结束了，把xDemo析构掉**

**Demo-Destructor.**

**A clock says: Bye-bye.1:1:1**

**A box says: Bye-bye.10-10-10//fun2结束了，把fun2析构掉**

**//----------fun2的输出完毕----------------------------------//**

**A clock says: Bye-bye.6:7:8**

**//----------pclock的析构-----------------------------------//**

**Demo-Destructor.**

**A clock says: Bye-bye.8:34:45**

**A box says: Bye-bye.20-30-10//先执行自己的析构函数，在执行系统的析构，顺序上，反过来就行了**

**Demo-Destructor.**

**A clock says: Bye-bye.7:8:9**

**A box says: Bye-bye.6-8-9**

**//----------pDemo的析构------------------------------------//**

**A box says: Bye-bye.10-10-10**

**//----------------------开始return------------------------//**

**Demo-Destructor.**

**A clock says: Bye-bye.4:0:0**

**A box says: Bye-bye.3-0-0**

**Demo-Destructor.**

**A clock says: Bye-bye.8:34:45**

**A box says: Bye-bye.10-10-10**

**Demo-Destructor.**

**A clock says: Bye-bye.1:1:1**

**A box says: Bye-bye.10-10-10**

**Demo-Destructor.**

**A clock says: Bye-bye.2:3:4**

**A box says: Bye-bye.1-3-4**

**//------------析构Demo-----------------------------------//**

**A box says: Bye-bye.20-30-10**

**A box says: Bye-bye.10-10-10**

**//-------------析构box-----------------------------------//**

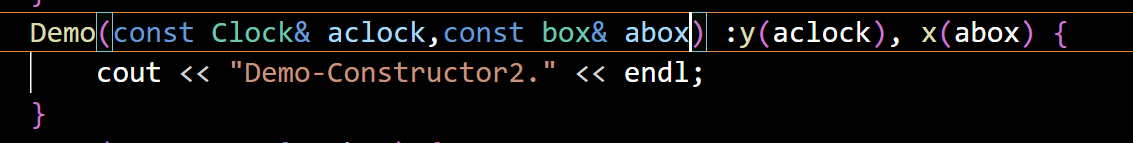
**A clock says: Bye-bye.8:34:45**

**A clock says: Bye-bye.8:34:45**

**//-------------析构Clock---------------------------------//**

**❸答：（1）都是先x后y**

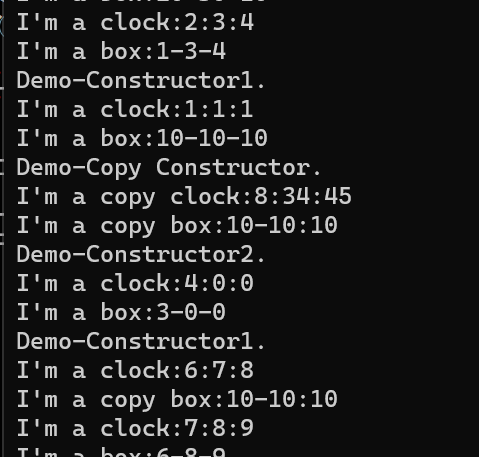
**（2）对于这个问题，我显示把Demo中的第二个构造函数改成了这样：**

****

**但还是先x后y，排除了与申明时顺序有关的可能性。**

**然后修改了头文件的顺序，也没有用。**

**最后，修改了Demo.hpp中声明x,y的顺序，顺序就换过来了：**

****

**变成先clock 后 box了**

**（3）没有关系的，可以看cdemo，它的构造函数的初始化列表是先y的，但是最终还是先x**

**❹答：**

**I'm a copy box:10-10:10**

**I'm a copy clock:8:34:45**

**Demo-Constructor2.**

**//这个是在定义的时候构造函数的输出。**

**Demo-Destructor.**

**A clock says: Bye-bye.8:34:45**

**A box says: Bye-bye.10-10-10**

**//这个是在return的时候析构函数的输出。**

**❺答：**

1. **静态定义，顺序是指定好的，就是先构造的后析构**
2. **动态定义，顺序可以自己用delete的顺序来定，但是对于一个delete中的顺序还是遵循先构造的后析构的规定。**

**❻答：**

**I'm a box:6-8-9**

**I'm a clock:7:8:9**

**Demo-Constructor1.**

**I'm a copy box:20-30:10**

**I'm a copy clock:8:34:45**

**Demo-Constructor2.**

**//这个是r22中的构造函数的输出**

**Demo-Destructor.**

**A clock says: Bye-bye.8:34:45**

**A box says: Bye-bye.20-30-10**

**Demo-Destructor.**

**A clock says: Bye-bye.7:8:9**

**A box says: Bye-bye.6-8-9**

**//这个是析构函数的输出**

**3、类的可缺省成员：Student（30分）**

**要求理解类的可缺省部分，以及缺省部分的适用情况。对于有派生数据的类要学会自己重写可缺省部分的操作。**

* **实验要求： 实现两种不同的student类(见ex3\_3\_student文件夹中的Student\_A,Student\_B), 并使用提供的主程进行测试。**

主程见：StudentApp.cpp, 请装配Student\_A的项目并运行观察结果；

然后完成student2.cpp,同样装配项目并观察运行结果，两者的结果是否一致？

输出结果应该为：

stu1:

stu2:Jennie Mao

stu3:Jennie Mao

stu4:John Smith

stu1:John Smith

stu1:Peter

stu3:Tom

* **实验提交：**

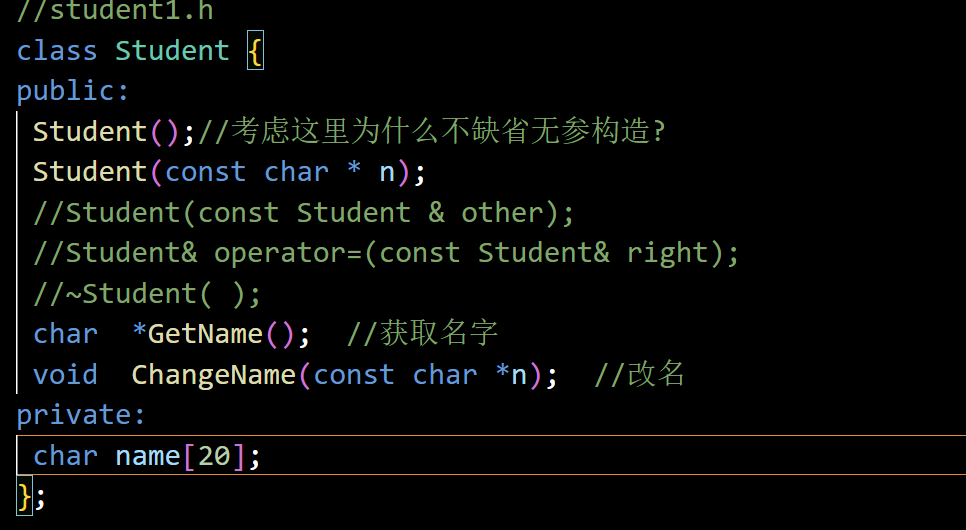
**提交完善后的Student\_A 和 Student\_B**

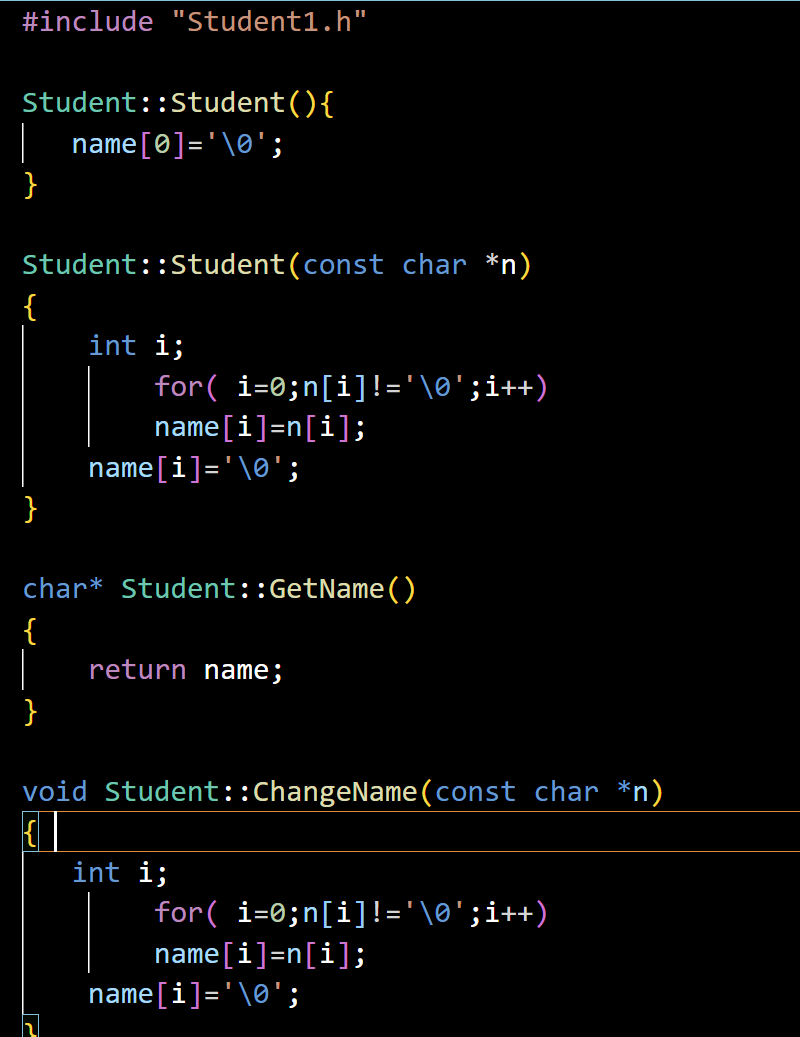
**注：我在Student A and B中把形参char \*n改成了const char\*n，否则我的编译器无法匹配输入的string类型**

**对于Student A为什么不缺省无参构造，缺省后我发现Student stu1**

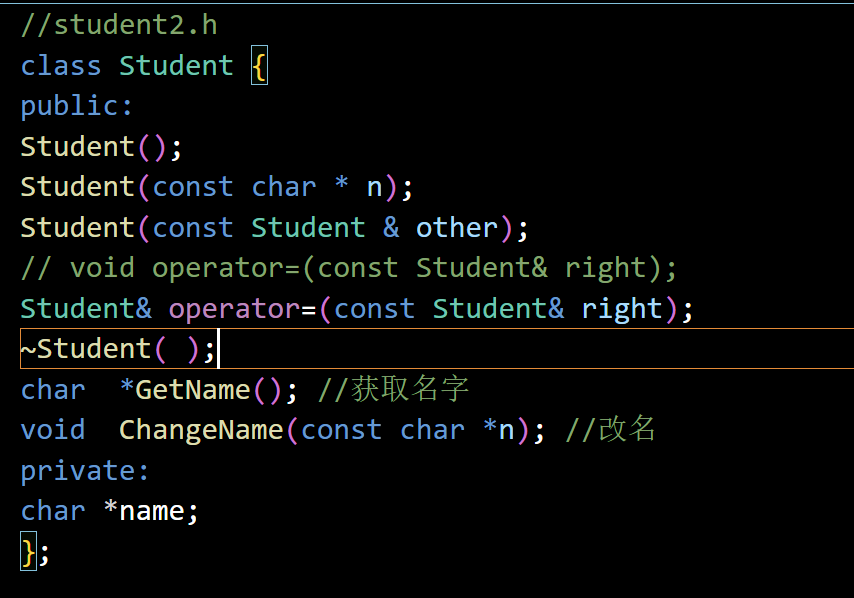
**会报错，这个可能是因为缺省的无参构造无法初始化name。**

**Student A:**

Hpp: ****

Cpp: 

**Student B:**

Hpp: 

Cpp: 