计算机学院 高级程序语言设计 课程实验报告

实验题目：类与对象应用		学号：202300130150
日期：2024.3.19	班级： 4	姓名： 王成意

实验目的：
1. 掌握类与对象的创建与应用
2. 熟悉类中构造函数、复制构造函数、析构函数等的运行机制。

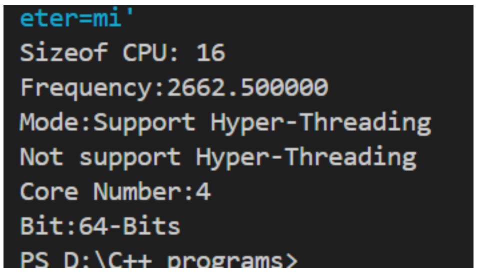
实验步骤与内容：
1. 实验任务：按照现实情况设计以下类
（1）类的声明与使用，CPU 类。可参考作业 4-19

```
#include<cstdio>

#include<iostream>

using namespace std;

enum wordlen

{

    Bit32,

    Bit64

};

enum core

{

    single,

    dual,

    quad

};

enum HyperThreading

{
```

```cpp
    nosupport,

    support

};

class CPU{

    private:

        float freq;

        HyperThreading mode;

        core cores;

        wordlen bits;

    public:

        CPU(float x, HyperThreading y, core z, wordlen
w);

        CPU(CPU &p);

        void show();

};

CPU::CPU(float x, HyperThreading y, core z, wordlen w)

{

    this->freq = x;

    this->mode = y;

    this->cores = z;

    this->bits = w;

}
```

```
CPU::CPU(CPU
&p):freq(p.freq),mode(p.mode),cores(p.cores),bits(p.bi
ts){}
void CPU::show()
{
    printf("Frequency:%f\n", freq);
    printf("Mode:");
    switch(mode)
    {
        case(support):
            printf("Support Hyper-Threading\n");
        case(nosupport):
            printf("Not support Hyper-Threading\n");
        default:
            break;
    }
    printf("Core Number:");
    switch(cores)
    {
        case(single):
            printf("1\n");
        case(dual):
            printf("2\n");
        case(quad):
            printf("4\n");
        default:
            break;
    }
    printf("Bit:");
    switch(bits)
    {
        case(Bit32):
            printf("32-Bits\n");
        case(Bit64):
            printf("64-Bits\n");
        default:
            break;
    }
}
int main()
{
    CPU N3160(2662.5, support, quad, Bit64);
    printf("Sizeof CPU: %llu\n", sizeof(CPU));
    N3160.show();
    return 0;

}
```

运行结果：



```
eter=mi'
Sizeof CPU: 16
Frequency:2662.500000
Mode:Support Hyper-Threading
Not support Hyper-Threading
Core Number:4
Bit:64-Bits
PS D:\C++ programs>
```

2）类的声明和对象的声明与使用，Computer 类。

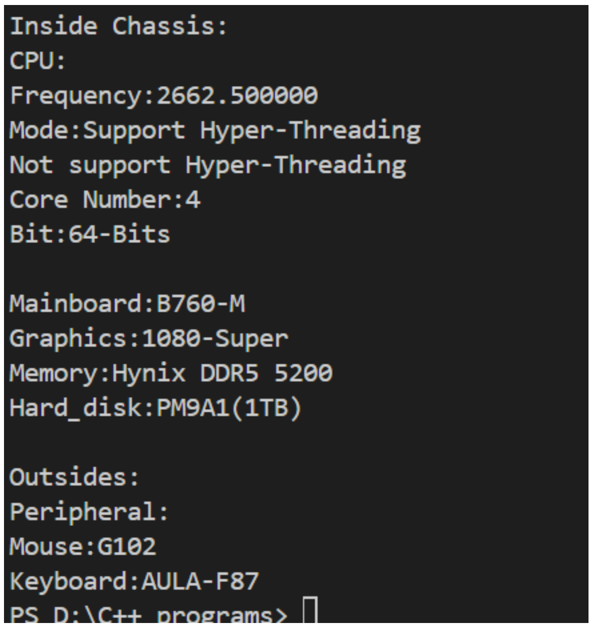
```
#include<cstdio>

#include<iostream>

#include<cstring>

#include<string>

using namespace std;

enum wordlen

{

    Bit32,

    Bit64

};

enum core

{

    single,

    dual,

    quad
```

```
};
enum HyperThreading
{
    nosupport,
    support
};
class CPU{
    private:
        float freq;
        HyperThreading mode;
        core cores;
        wordlen bits;
    public:
        CPU(float x, HyperThreading y, core z, wordlen
w);
        CPU(CPU &p);
        void show();
};
CPU::CPU(float x, HyperThreading y, core z, wordlen w)
{
    this->freq = x;
    this->mode = y;
    this->cores = z;

    this->bits = w;

}

CPU::CPU(CPU &p){

    this->freq = p.freq;

    this->mode = p.mode;

    this->cores = p.cores;

    this->bits = p.bits;

}

void CPU::show()

{

    printf("CPU:\n");

    printf("Frequency:%f\n", freq);

    printf("Mode:");

    switch(mode)

    {

        case(support):

            printf("Support Hyper-Threading\n");

        case(nosupport):

            printf("Not support Hyper-Threading\n");

        default:

            break;
    }
    printf("Core Number:");
    switch(cores)
    {
        case(single):
            printf("1\n");
        case(dual):
            printf("2\n");
        case(quad):
            printf("4\n");
        default:
            break;
    }
    printf("Bit:");
    switch(bits)
    {
        case(Bit32):
            printf("32-Bits\n");
        case(Bit64):
            printf("64-Bits\n");
        default:
            break;
    }

    puts("");

}

class peripheral

{

    public:

        peripheral(string a, string b);

        peripheral(peripheral &p);

        void show();

    private:

        string mouse;

        string keyboard;

};

peripheral::peripheral(string a,string b)

{

    this->mouse = a;

    this->keyboard = b;

}

peripheral::peripheral(peripheral &p){

    this->keyboard = p.keyboard;

    this->mouse = p.mouse;

}
```

```cpp
void peripheral::show()

{

    printf("Outsides:\n");

    printf("Peripheral:\n");

    printf("Mouse:");

    cout << this->mouse << endl;

    printf("Keyboard:");

    cout << this->keyboard << endl;

}

class Chassis{

    private:

        CPU cpu;

        string mainboard;

        string graphics;

        string memory;

        string hard_disk;

    public:

        Chassis(CPU &a, string b, string c, string d,

string e);

        Chassis(Chassis &p);

        void show();

};
```

```
Chassis::Chassis(Chassis

&p):cpu(p.cpu),mainboard(p.mainboard),graphics(p.graph

ics),memory(p.memory),hard_disk(p.hard_disk){}

Chassis::Chassis(CPU &a, string b, string c, string d,

string e):cpu(a)

{

    this->mainboard = b;

    this->graphics = c;

    this->memory = d;

    this->hard_disk = e;

}

void Chassis::show()

{

    printf("Inside Chassis:\n");

    this->cpu.show();

    printf("Mainboard:");

    cout << this->mainboard << endl;

    printf("Graphics:");

    cout << this->graphics << endl;

    printf("Memory:");

    cout << this->memory << endl;

    printf("Hard_disk:");
    cout << this->hard_disk << endl;

    puts("");

}

class Computer{

    private:

        Chassis ins;

        peripheral outs;

    public:

        Computer(Chassis &a, peripheral &b);

        void show();

};

Computer::Computer(Chassis &a,peripheral

&b):outs(b),ins(a){}

void Computer::show()

{

    ins.show();

    outs.show();

}

int main()

{

    CPU N3160(2662.5, support, quad, Bit64);

    peripheral outside("G102", "AULA-F87");
    Chassis inside(N3160, "B760-M", "1080-Super",
"Hynix DDR5 5200", "PM9A1(1TB)");
    Computer kkk(inside, outside);
    kkk.show();
    return 0;
}
```

运行结果：



```
Inside Chassis:
CPU:
Frequency:2662.500000
Mode:Support Hyper-Threading
Not support Hyper-Threading
Core Number:4
Bit:64-Bits

Mainboard:B760-M
Graphics:1080-Super
Memory:Hynix DDR5 5200
Hard_disk:PM9A1(1TB)

Outsides:
Peripheral:
Mouse:G102
Keyboard:AULA-F87
PS D:\C++ programs>
```

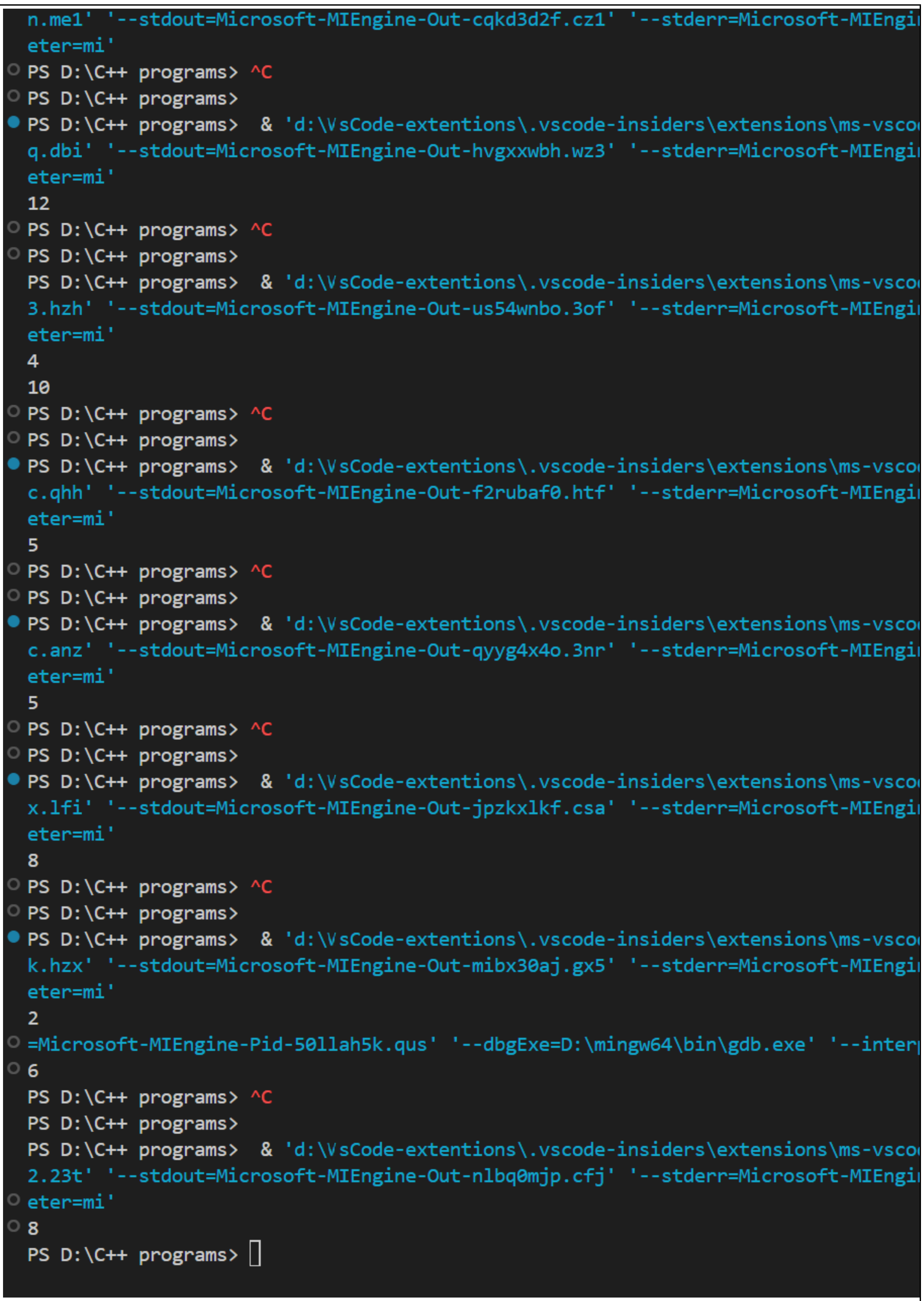
2. 基于课件的拓展练习：

（1）参考第 4 章 PPT 中的例 4-1，针对第 3 章 PPT 中的例 3-6 设计一个骰子类 Dice（分析属性和方法是什么？），用面向对象 OO 的方法实现其功能。

```
#include<cstdio>
#include<cstdlib>
#include<iostream>
#include<time.h>
using namespace std;
class dices{
    public:
        dices(int x1,int x2)
        {
            d1 = x1, d2 = x2;
        }
        void roll();
        int getSum();
        void putSum();

    private:
        int d1, d2;
};
int dices::getSum()
{
    return d1 + d2;
}
void dices::roll()
{
    d1 = 1 + rand() % 6;
    d2 = 1 + rand() % 6;
}
void dices::putSum()
```

```
{
    printf("%d\n", d1 + d2);
}
int main()
{
    srand((unsigned) time(NULL));
    dices ddd(0,0);
    ddd.roll();
    ddd.putSum();
    return 0;
}
```

运行结果：



```
n.me1' '--stdout=Microsoft-MIEngine-Out-cqkd3d2f.cz1' '--stderr=Microsoft-MIEngi
eter=mi'
PS D:\C++ programs> ^C
PS D:\C++ programs>
PS D:\C++ programs>  & 'd:\VsCode-extentions\.vscode-insiders\extensions\ms-vsco
q.dbi' '--stdout=Microsoft-MIEngine-Out-hvgxxwbh.wz3' '--stderr=Microsoft-MIEngi
eter=mi'
12
PS D:\C++ programs> ^C
PS D:\C++ programs>
PS D:\C++ programs>  & 'd:\VsCode-extentions\.vscode-insiders\extensions\ms-vsco
3.hzh' '--stdout=Microsoft-MIEngine-Out-us54wnbo.3of' '--stderr=Microsoft-MIEngi
eter=mi'
4
10
PS D:\C++ programs> ^C
PS D:\C++ programs>
PS D:\C++ programs>  & 'd:\VsCode-extentions\.vscode-insiders\extensions\ms-vsco
c.qhh' '--stdout=Microsoft-MIEngine-Out-f2rubaf0.htf' '--stderr=Microsoft-MIEngi
eter=mi'
5
PS D:\C++ programs> ^C
PS D:\C++ programs>
PS D:\C++ programs>  & 'd:\VsCode-extentions\.vscode-insiders\extensions\ms-vsco
c.anz' '--stdout=Microsoft-MIEngine-Out-qyyg4x4o.3nr' '--stderr=Microsoft-MIEngi
eter=mi'
5
PS D:\C++ programs> ^C
PS D:\C++ programs>
PS D:\C++ programs>  & 'd:\VsCode-extentions\.vscode-insiders\extensions\ms-vsco
x.lfi' '--stdout=Microsoft-MIEngine-Out-jpzkxlkf.csa' '--stderr=Microsoft-MIEngi
eter=mi'
8
PS D:\C++ programs> ^C
PS D:\C++ programs>
PS D:\C++ programs>  & 'd:\VsCode-extentions\.vscode-insiders\extensions\ms-vsco
k.hzx' '--stdout=Microsoft-MIEngine-Out-mibx30aj.gx5' '--stderr=Microsoft-MIEngi
eter=mi'
2
=Microsoft-MIEngine-Pid-50llah5k.qus' '--dbgExe=D:\mingw64\bin\gdb.exe' '--inter
6
PS D:\C++ programs> ^C
PS D:\C++ programs>
PS D:\C++ programs>  & 'd:\VsCode-extentions\.vscode-insiders\extensions\ms-vsco
2.23t' '--stdout=Microsoft-MIEngine-Out-nlbq0mjp.cfj' '--stderr=Microsoft-MIEngi
eter=mi'
8
PS D:\C++ programs> 
```

基于第 4 章 PPT 中的例 4-2，分析发现问题，进行探索、尝试。

```
#include<cstdio>

#include<iostream>

using namespace std;
```

```
class Point {   //Point 类的定义
public:
    Point(int xx=0, int yy=0) { x = xx; y =
yy; }    //构造函数，内联
    Point(const Point& p); //复制构造函数
  void setX(int xx) {x=xx;}
  void setY(int yy) {y=yy;}
    int getX() const { return x; } //常函数（第 5 章）
    int getY() const { return y; } //常函数（第 5 章）
private:
    int x, y; //私有数据
};
//成员函数的实现
Point::Point (const Point& p) {
  x = p.x;
  y = p.y;
  cout << "Calling the copy constructor " << endl;
}
//形参为 Point 类对象的函数
void fun1(Point p) {
    cout << p.getX() << endl;
}
```

```cpp
//返回值为 Point 类对象的函数
Point fun2() {
    Point a(1, 2);
    return a;
}
//主程序
int main() {
    Point a(4, 5); //构造第一个对象 a
    Point b = a; //情况一，用 a 初始化 b。第一次调用复制构
造函数，等价于 b(a)
                  //注：此处不是赋值运算，与单独 b=a 不同，不
调用=运算符重载。
    cout << b.getX() << endl;
    fun1(b);     //情况二，对象 B 作为 fun1 的实参。第二次调
用复制构造函数
    b = fun2(); //情况三，函数的返回值是类对象，函数返回时
调用复制构造函数
                 //注：vc 调用复制构造函数，但其他编译器优化成
调用=运算符重载
    cout << b.getX() << endl;
    return 0;
}
```

对于：

```
Point b = a; //情况一，用 a 初始化 b。第一次调用复制构造函数，等价于 b(a)
```

B 需要开辟新的内存，调用 point::point(const point &p)
对于：

```
fun1(b);
```

b 作为实参传递给

```
void fun1(Point p) {

    cout << p.getX() << endl;

}
```

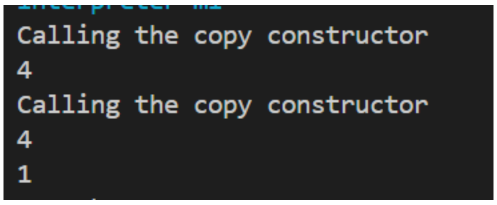
中的 p（形参），也就是形参和实参结合的时候，再次调用了复制构造函数；
对于

```
b = fun2()
```

，因为

```
Point fun2() {

    Point a(1, 2);

    return a;

}
```

返回值是类的对象，所以函数返回时应该调用了复制构造函数，但是 vscode 优化，把复制构造函数变成重载运算符=。
运行结果：



```
Calling the copy constructor
4
Calling the copy constructor
4
1
```

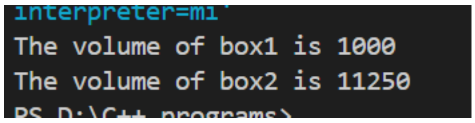
3. 分析附件实验 4 素材中的程序运行结果。
（1）构造函数重载及参数初始化表，c9-3.cpp

```
#include <iostream>

using namespace std;
```

```cpp
class Box
{
public:
  ~Box();
  Box(int h=10, int w=10, int len=10) : height(h),
width(w), length(len) {} //参数初始化表
  int volume();

private:
  int height;
  int width;
  int length;
};
Box::~Box()
{

}
int Box::volume()
{
  return (height * width * length);
}
```

```
int main()
{
  Box box1;
  cout << "The volume of box1 is " << box1.volume() <<
endl;
  Box box2(15, 30, 25);
  cout << "The volume of box2 is " << box2.volume() <<
endl;
  return 0;
}
```

结果：



```
interpreter=mi'
The volume of box1 is 1000
The volume of box2 is 11250
PS D:\C++ programs>
```

（2）使用默认参数的构造函数，c9-4.cpp。

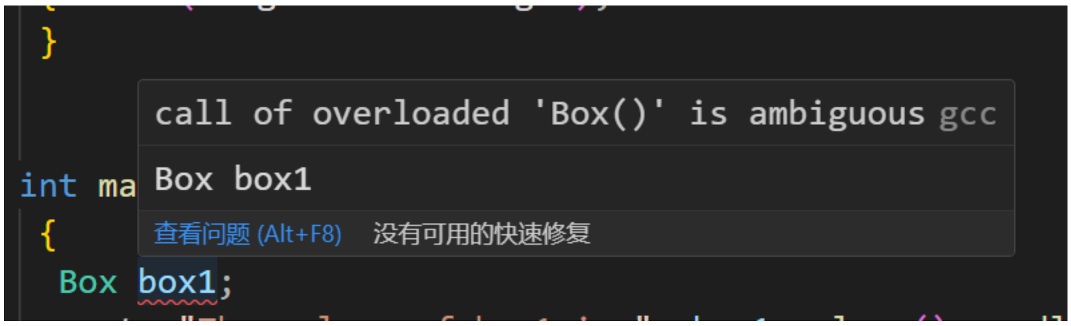
　问：与 c9-3 比，它还能加入一个默认构造函数 Box() 吗？

不可以。

加上默认构造函数：

```
class Box
 {public:
   Box() = default;
   Box(int=10,int=10,int=10);   //参数默认值


   int volume();
  private:
   int height;

   int width;

   int length;

 };

```

出现结果：



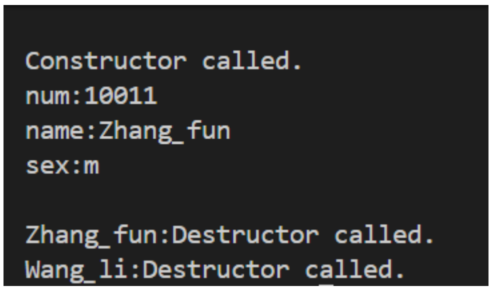
```
 }
        call of overloaded 'Box()' is ambiguous gcc
        Box box1
int ma  查看问题 (Alt+F8)   没有可用的快速修复
 {
   Box box1;
```

析构函数。 c9-5.cpp。 注意析构函数执行的顺序。

```
//析构函数示例

#include <iostream>

#include <string>

using namespace std;

class Student

 {public:

   Student(int n,string nam,char s)

    {num=n;

     name=nam;

     sex=s;

     cout<<"Constructor called."<<endl;

    }
   ~Student()
   {cout<<name<<":Destructor called."<<endl;}

   void display()
   {cout<<"num:"<<num<<endl;
    cout<<"name:"<<name<<endl;
  cout<<"sex:"<<sex<<endl<<endl;
   }

  private:
    int num;
    string name;
    char sex;
};

int main()
  {Student stud1(10010,"Wang_li",'f');
   stud1.display();
   Student stud2(10011,"Zhang_fun",'m');
   stud2.display();
   return 0;
```

```
}
```

运行结果：

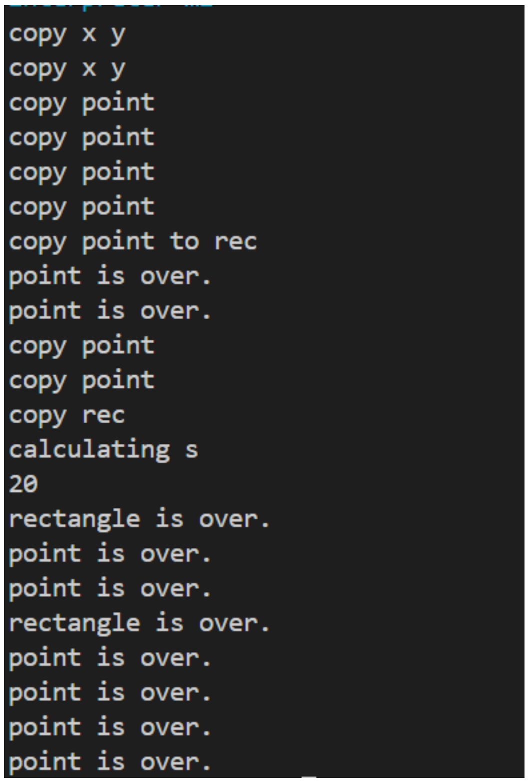


```
Constructor called.
num:10011
name:Zhang_fun
sex:m

Zhang_fun:Destructor called.
Wang_li:Destructor called.
```

类似地，在写作业时发现：

```cpp
#include<cstdio>
#include<cmath>
using namespace std;
class point{
    public:
        point(int x, int y);
        point(point &p);
        ~point();
        int getx()
        {
            return x;
        }
        int gety()
        {
            return y;
        }
    private:
        int x, y;
};
point::point(int xx,int yy)
{
    printf("copy x y\n");
    x = xx;
    y = yy;
}
point::point(point &p)
{
    printf("copy point\n");
    x = p.x;
    y = p.y;
}
point:: ~point()
{
    printf("point is over.\n");
}
class rectangle{
    public:
        rectangle(point x, point y);

        rectangle(rectangle &p);

        ~rectangle();

        point get1()

        {

            return a;

        }

        point get2()

        {

            return b;

        }

        int s();


    private:

        point a, b;

};

rectangle::rectangle(rectangle &p):a(p.a),b(p.b)

{

    printf("copy rec\n");

}

rectangle::rectangle(point xx,point yy):a(xx),b(yy)

{
    printf("copy point to rec\n");
}
int rectangle::s()
{
    printf("calculating s\n");
    return abs((a.getx() - b.getx()) * (a.gety() -
b.gety()));
}
rectangle::~rectangle()
{
    printf("rectangle is over.\n");
}
int main()
{
    point n(1, 1), m(5, 6);
    rectangle rec(n, m);
    rectangle rec2(rec);
    printf("%d\n", rec2.s());
    return 0;
}
```

的运行结果：



```
copy x y
copy x y
copy point
copy point
copy point
copy point
copy point to rec
point is over.
point is over.
copy point
copy point
copy rec
calculating s
20
rectangle is over.
point is over.
point is over.
rectangle is over.
point is over.
point is over.
point is over.
point is over.
```

当变量结束作用时调用析构函数。

结论分析与体会：
此次试验十分有意义，加深了我对类和对象的理解。