课程名称：面对象程序设计与应用　指导教师：张潇

班级：信工 16-2 班　姓名：王璐　学号：1610480214

实验项目名称：

实验二 类与对象

实验目的及要求：

1.掌握类的概念、类的定义格式、类与结构的关系、类的成员属性和类的封装性；理解类的成员的访问控制的含义，公有、私有和保护成员的区别。
2.掌握成员函数的实现与调用方法，能够根据给定的要求定义类并实现类的成员函数。
3.掌握类对象的定义；深刻领会类与对象的区别，类实现数据隐藏与封装的原理等。
4.掌握构造函数和析构函数的含义与作用、定义方式和实现，能够根据要求正确定义和重载构造函数。学会编写与应用复制构造函数。
5.了解静态成员的使用，对象成员的概念，掌握对象成员的初始化方法，掌握对象数组的使用。

实验原理：

C++面对象程序设计的原理，class 类封装的应用，成员函数数据结构的相关知识，构造析构函数的使用方法。

实验内容（方法和步骤）：

验证性题目：

1.（1）源代码:

```
#include <iostream>
#include <string>
#include <cstring>
using namespace std;
class Person{
	private:
		char *name;
		int age;
	public:
		Person(const Person &p);
		Person(char *Name,int Age);
		~Person();
		void setAge(int x){age=x;}
		void print();
};
```

```
Person::Person(char *Name,int Age){
        name=new char[strlen(Name)+1];
        strcpy(name,Name);
        age=Age;
        cout<<"constructor ...."<<endl;
}
Person::Person(const Person &p){
        name=new char[strlen(p.name)+1];
        strcpy(name,p.name);
        age=p.age;
        cout<<"Copy constructor ...."<<endl;
}
Person::~Person(){
        cout<<"destructor ..."<<age<<endl;
        delete name;
}
void Person::print(){
        cout<<name<<"\t The Address of name: "<<&*name<<endl;
}
int main(){
        Person p1("张勇",21);
        Person p2=p1;
        p1.setAge(1);
        p2.setAge(2);
        p1.print();
        p2.print();
        return 0;
}
```

（2）源代码：

```
#include <iostream>
#include <string>
using namespace std;
class Book {
        private:
                string bkName;
                double price;
                static int number;
                static double totalPrice;
        public:
                Book() { bkName=""; price=0; number++;}
                Book(string , double);
                ~Book();
                void setName(string bname){bkName=bname;}
```

```
		void setPrice(double bprice){
			totalPrice-=price;
			price=bprice;
			totalPrice+=price;
		}
		double getPrice(){return price;}
		string getName() {return bkName;}
		static int getNumber() {return number;}
		static double getTotalPrice() {return totalPrice;}
		void display();
};
Book::Book(string name,double Price){
	bkName=name;
	price=Price;
	number++;
	totalPrice+=price;
}
Book::~Book(){
	number--;
	totalPrice-=price;
}
void Book::display(){
	cout<<"book name :"<<bkName<<" "<<"price:"<<price<<endl;
	cout<<"number:"<<number<<" "<<"totalPrice:"<<totalPrice<<endl;
	cout<<"call static function "<<getNumber()<<endl;
}
int Book::number=0;
double Book::totalPrice=0;
int main(){
	Book b1("C++程序设计",32.5),b2;
	b2.setName("数据库系统原理");
	b2.setPrice(23);
	cout<<b1.getName()<<"\t"<<b1.getPrice()<<endl;
	cout<<b2.getName()<<"\t"<<b2.getPrice()<<endl;
	cout<<"总共: "<<b1.getNumber()<<"\t 本书"
		<<"\t  总价：	"<<b1.getTotalPrice()<<"\t 元"<<endl;
	{
		Book b3("数据库系统原理",23);
		cout <<"总共："<<b1.getNumber()<<"\t 本书"
			<<"\t 总价：  "<<b1.getTotalPrice()<<"\t 元"<<endl;
	}
	cout<<"总共："<<Book::getNumber()<<"\t 本书"
		<<"\t 总价：  "<<Book::getTotalPrice()<<"\t 元"<<endl;
	b2.display();
```

```
        return 0;
}
```

（3）源代码：

```
class X{
        private:
                int a=0,&b;          //引用在定义时应该初始化
                const int c;                //常量定义 c 时应该初始化
                void setA(int i){a=i;}          //应该放到公共成员
                X(int i){a=i;}                     //应该放到公共成员里
        public:
                int X(){a=b=c=0;}                   //b 和 c 都不能赋值
                X(int i,int j,int k) {a=i;b=j; c=k;}    // b，c 不能赋值
                static void setB(int k) {b=k;}               //静态成员函数只能访问静态变量，b 不是静态变量
                setC(int k) const {c=c+k;}        //要有 void  ，加了 const c 不能修改
        };
void main(){
        X x1;               //无参数构造函数是错的
        X x2(3);
        X x3(1,2,3);        //2，3 不能赋值到 b 和 c
        x1.setA(3);           //私有成员无法访问
}
```

（4）源代码：

```
#include <iostream>
#include <string>
#include <cstring>
using namespace std;
class X{
                int a;
                char *b;
                float c;
        public:
                X(int x1,char *x2,float x3):a(x1),c(x3){
                        b=new char[sizeof(x2)+1];
                        strcpy(b,x2);
                }
                X():a(0),b("X::X()"),c(10){}
                X(int x1,char *x2="X::X(...)",int x3=10):a(x1),b(x2),c(x3){}
                X(const X&other){
                        a=other.a;
                        b="X::X(const X &other)";
                        c=other.c;
                }
                void print() {cout<<"a="<<a<<"\t"<<"b="<<b<<"\t"<<"c="<<c<<endl;}
};
int main(){
        X *A=new X (4,"X::X(int, char, float)",32);
        X B, C(10), D(B);
        A->print();    B.print();
        C.print();      D.print();
        return 0;
}
```

（4）的第一问，源代码：

```
#include <iostream>
#include <string>
#include <cstring>
using namespace std;
class X{
                int a;
                char *b;
                float c;
        public:
                X(int x1,char *x2,float x3):a(x1),c(x3){
                        b=new char[sizeof(x2)+1];
                        strcpy(b,x2);
                }
                X():a(0),b("X::X()"),c(10){}
                X(int x1,char *x2="X::X(...)",int x3=10):a(x1),b(x2),c(x3){}
                X(const X&other){
                        a=other.a;
                        b="X::X(const X &other)";
                        c=other.c;
                }
                void print() {cout<<"a="<<a<<"\t"<<"b="<<b<<"\t"<<"c="<<c<<endl;}
};
int main(){
        X *A=new X (4,"X::X(int, char, float)",32);
        X B, C(10), D(B);
        A->print();    B.print();
        C.print();      D.print();
        return 0;
}
```

第二问，源代码:

```
#include <iostream>
using namespace std;
```

```
class Implementation {
        public:
                Implementation(int v) {value=v;}
                void setValue(int v) {value=v;}
                int getValue() const {return value;}
        private:
                int value;
};
class Interface{
        public:
                Interface(int);
                void setValue(int);
                int getValue() const;
        private:
                Implementation *ptr;
};
Interface::Interface(int v):ptr(new Implementation(v)){}
void Interface::setValue(int v) { ptr->setValue(v);}
int Interface::getValue() const {return ptr->getValue();}
int main(){
        Interface i(5);
        cout<<i.getValue()<<endl;
        i.setValue(10);
        cout<<i.getValue()<<endl;
        return 0;
}
```

第三问，源代码：

```
#include <iostream>
using namespace std;
class A{
                int x;
        public:
                A():x(0) {cout<<"constructor A() called..."<<endl; }
                A(int i):x(i) { cout<<"X"<<x<<"\tconstructor..."<<endl;}
                ~A() {cout<<"X"<<x<<"\tdestructor..."<<endl;}
};
class B{
                int y;
                A X1,X2[3];
        public:
                B(int j):X1(j),y(j) {cout<<"B"<<j<<"\tconstructor..."<<endl;}
                ~B() {cout<<"B"<<y<<"\tdestructor..."<<endl;}
};
```

```
int main(){
	A X1(1),X2(2);
	B B1(3);
	return 0;
}
```

2.设计性题目

（1）源代码:

```
Strack.cpp
#include "stack.h"
#include <iostream>
using namespace std;
Stack::Stack(int stacksize){
	if(stacksize>0){
		maxSize=stacksize;
		data=new int[stacksize];
		for(int i=0;i<maxSize;i++)
			data[i]=0;
	}
	else{
		data=0;
		maxSize=0;
	}
	top=0;
}
Stack::~Stack(){
	delete[] data;
}
void Stack::push(int x){
	if (top<maxSize){
		data[top]=x;
		top++;
	}
	else{
		cout<<"堆栈已满，不能再入栈数据："<<x<<endl;
	}
}
int Stack::pop(){
	if(top<=0){
		cout<<"堆栈已空！"<<endl;
		exit(1);
	}
	top--;
	return data[top];
}
int Stack::howMany(){
	return top;
}

Strack.h:
#ifndef Stack_h
#define Stack_h
class Stack{
	private:
		int *data;
		int top;
		int maxSize;
	public:
		Stack(int stacksize=10);
		~Stack();
		void push(int x);
		int pop();
		int howMany();
};
#endif

Strackmain.cpp:
#include "stack.cpp"
#include <iostream>
using namespace std;
int main(){
	Stack s1;
	s1.push(1);
	s1.push(12);
	s1.push(32);
	int x1=s1.pop();
	int x2=s1.pop();
	int x3=s1.pop();
	cout<<x1<<"\t"<<x2<<"\t"<<x3<<endl;
	cout<<"目前栈内数据为:"<<s1.howMany()<<endl;
	return 0;
}
```

(2)源代码：

```
#include <iostream>
using namespace std;
class Salary{
	private:
```

}

```
		double Wage,Subsidy,Rent,WaterFee,ElecFee;
	public:
		Salary(double w,double s,double r,double wt,double e) {
			Wage=w;
			Subsidy=s;
			Rent=r;
			WaterFee=wt;
			ElecFee=e;
		};
		Salary (){
			Wage=0;
			Subsidy=0;
			Rent=0;
			WaterFee=0;
			ElecFee=0;
		};
		void setWage(double w){Wage=w;};
		void setSubsidy(double s){Subsidy=s;};
		void setRent(double r){Rent=r;};
		void setWaterFee(double wt){WaterFee=wt;};
		void setElecFee(double e){ElecFee=e;};
		double getWage(){return Wage;};
		double getSubsidy(){return Subsidy;};
		double getRent(){return Rent;};
		double getWaterFee(){return WaterFee;};
		double getElecFee(){return ElecFee;};
		double RealSalary(){ return Wage+Subsidy-Rent-WaterFee-ElecFee;};
		void print(){ cout<<"基本工资："<<getWage()<<endl<<"岗位津贴："<<getSubsidy()<<endl<<"房租：
"<<getRent()<<endl<<"水费："<<getWaterFee()<<endl<<"电费："<<getElecFee()<<endl<<"实发工资
"<<RealSalary()<<endl; };
};
int main(){
	Salary door(2000,1000,200,222,123.4);
	door.print();
	cout<<"设置基本工资："<<endl;
	double x;
	cin>>x;
	door.setWage(x);
	door.print();
	return 0;
}
```

（3）源代码：

```
#include <iostream>
#include <string>
using namespace std;
class Salary{
	private:
		double Wage,Subsidy,Rent,WaterFee,ElecFee;
	public:
		Salary(double w,double s,double r,double wt,double e) {
			Wage=w;
			Subsidy=s;
			Rent=r;
			WaterFee=wt;
			ElecFee=e;
		};
		Salary (){
			Wage=0;
			Subsidy=0;
			Rent=0;
			WaterFee=0;
			ElecFee=0;
		};
		void setWage(double w){Wage=w;};
		void setSubsidy(double s){Subsidy=s;};
		void setRent(double r){Rent=r;};
		void setWaterFee(double wt){WaterFee=wt;};
		void setElecFee(double e){ElecFee=e;};
		double getWage(){return Wage;};
		double getSubsidy(){return Subsidy;};
		double getRent(){return Rent;};
		double getWaterFee(){return WaterFee;};
		double getElecFee(){return ElecFee;};
		double RealSalary(){ return Wage+Subsidy-Rent-WaterFee-ElecFee;};
		void print(){ cout<<"基本工资："<<Wage<<endl<<"岗位津贴："<<Subsidy<<endl<<"房租：
"<<Rent<<endl<<"水费："<<WaterFee<<endl<<"电费："<<ElecFee<<endl<<"实发工资"<<RealSalary()<<endl; };
};
class Worker{
	private:
		string name,dept;
		int age;
		Salary salary;
		static int sum;
	public:
		Worker(string n,int a,string d,double w,double s,double r,double wt,double e){
			name=n;
```

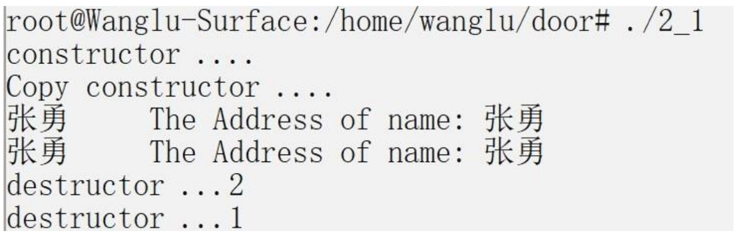
```
                age=a;
                dept=d;
                salary.setWage(w);
                salary.setSubsidy(s);
                salary.setRent(r);
                salary.setWaterFee(wt);
                salary.setElecFee(e);
                sum++;
                };
        Worker(){name="none";age=0;dept="none";sum=sum+1;};
        ~Worker(){sum--;};
        void setname(string n){name=n;};
        void setage(int a){age=a;};
        void setdept(string d){dept=d;};
        void setsalary(double w,double s,double r,double wt,double e){
                salary.setWage(w);
                salary.setSubsidy(s);
                salary.setRent(r);
                salary.setWaterFee(wt);
                salary.setElecFee(e);
        };
        string getname(){return name;};
        int getage(){return age;};
        string getdept(){return dept;};
        Salary getsalary(){return salary;};
        static int getsum(){return sum;};
        void print(){
                cout<<"姓名："<<name<<endl<<"年龄："<<age<<endl<<"工作部门："<<dept<<endl;
                 cout<<"基本工资："<<salary.getWage()<<endl<<"岗位津贴："<<salary.getSubsidy()<<endl<<"房租："<<salary.getRent()<<endl<<"水费："<<salary.getWaterFee()<<endl<<"电费："<<salary.getElecFee()<<endl<<"实发工资"<<getsalary().RealSalary()<<endl<<"总人数："<<getsum()<<endl; };

};
int Worker::sum=0;
int main(){
        Worker door("王璐",23,"cumtb",20000,1000,200,222,123.4);
        door.print();
        string name,dept;
        int age;
        double w,s,r,wt,e;
        cout<<"请输入 姓名 年龄 工作部门 基本工资 岗位津贴 房租 水费 电费"<<endl;
        cin>>name>>age>>dept>>w>>s>>r>>wt>>e;
        cout<<endl<<endl;
        Worker van(name,age,dept,w,s,r,wt,e);
```

```
	van.print();
	return 0;
}
```

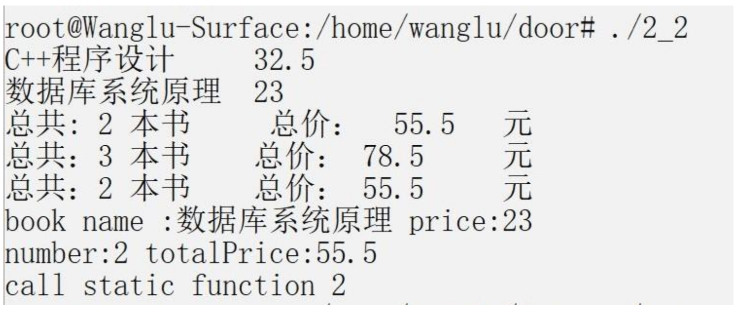
实验结果与分析：

在写程序的过程中遇到了一些问题，例如头文件的导入和使用，最终经过查询，自己编写 makefile 文件实现了多个源文件的生成。在使用局部静态变量时，初始化遇到了一些问题，最终经过查资料，解决了问题。

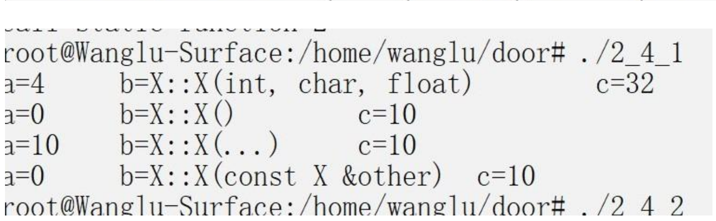
图片结果：



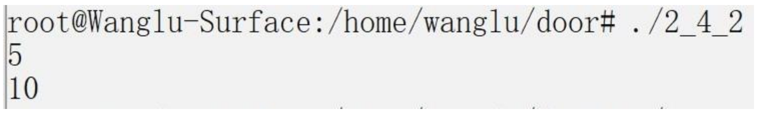
```
root@Wanglu-Surface:/home/wanglu/door# ./2_1
constructor ....
Copy constructor ....
张勇      The Address of name: 张勇
张勇      The Address of name: 张勇
destructor ...2
destructor ...1
```



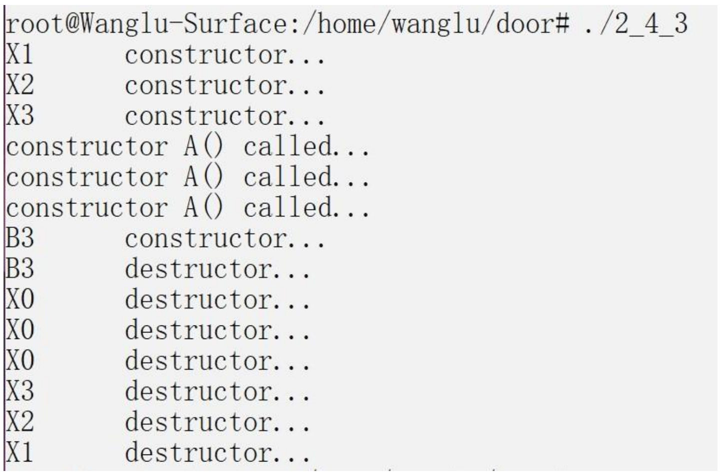
```
root@Wanglu-Surface:/home/wanglu/door# ./2_2
C++程序设计      32.5
数据库系统原理   23
总共: 2 本书      总价:    55.5    元
总共: 3 本书     总价:  78.5      元
总共: 2 本书     总价:  55.5      元
book name :数据库系统原理 price:23
number:2 totalPrice:55.5
call static function 2
```



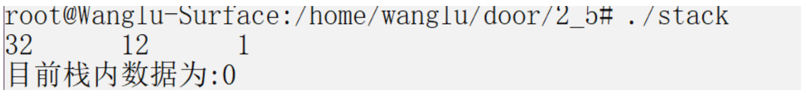
```
root@Wanglu-Surface:/home/wanglu/door# ./2_4_1
a=4      b=X::X(int, char, float)        c=32
a=0      b=X::X()         c=10
a=10     b=X::X(...)      c=10
a=0      b=X::X(const X &other)   c=10
root@Wanglu-Surface:/home/wanglu/door# ./2_4_2
```



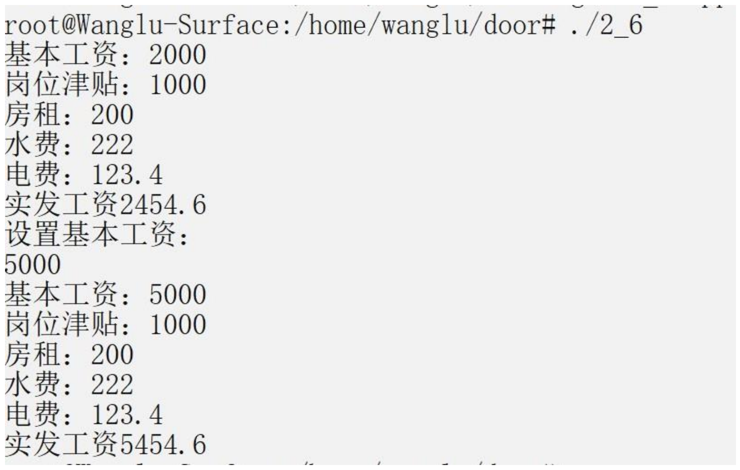
```
root@Wanglu-Surface:/home/wanglu/door# ./2_4_2
5
10
```



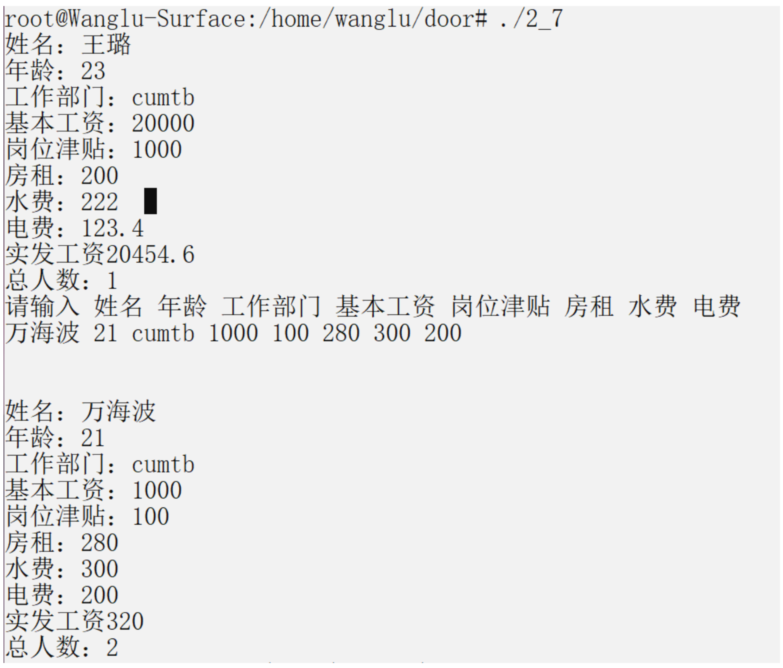
```
root@Wanglu-Surface:/home/wanglu/door# ./2_4_3
X1      constructor...
X2      constructor...
X3      constructor...
constructor A() called...
constructor A() called...
constructor A() called...
B3      constructor...
B3      destructor...
X0      destructor...
X0      destructor...
X0      destructor...
X3      destructor...
X2      destructor...
X1      destructor...
```



```
root@Wanglu-Surface:/home/wanglu/door/2_5# ./stack
32      12      1
目前栈内数据为:0
```



```
root@Wanglu-Surface:/home/wanglu/door# ./2_6
基本工资: 2000
岗位津贴: 1000
房租: 200
水费: 222
电费: 123.4
实发工资2454.6
设置基本工资:
5000
基本工资: 5000
岗位津贴: 1000
房租: 200
水费: 222
电费: 123.4
实发工资5454.6
```



```
root@Wanglu-Surface:/home/wanglu/door# ./2_7
姓名: 王璐
年龄: 23
工作部门: cumtb
基本工资: 20000
岗位津贴: 1000
房租: 200
水费: 222
电费: 123.4
实发工资20454.6
总人数: 1
请输入 姓名 年龄 工作部门 基本工资 岗位津贴 房租 水费 电费
万海波 21 cumtb 1000 100 280 300 200


姓名: 万海波
年龄: 21
工作部门: cumtb
基本工资: 1000
岗位津贴: 100
房租: 280
水费: 300
电费: 200
实发工资320
总人数: 2
```

成绩：　　　　　　　　批阅教师签名：　　　　　　　　年　　月　　日