|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 数学与信息科学学院 | | | | | | |  |
|  |  | | | | | | |  |
|  |  | | | | | | |  |
|  | **实验报告** | | | | | | |  |
|  |  | | | | | | |  |
|  |  | | | | | | |  |
|  | **课程名称：** | | **程序设计技术** | | | | |  |
|  | **姓 名：** | | **任希恒** | | | | |  |
|  | **学 号：** | | **541910010217** | | | | |  |
|  | **专业班级：** | | **信科1902** | | | | |  |
|  | **指导教师：** | | **裴云霞** | | | | |  |
|  |  | |  | | | | |  |
|  |  | |  | | | | |  |
|  |  | **2019-2020** | | **学年第** | **2** | **学期** |  |  |

**实验（二）类和对象**

**实验日期： 2020年 10月4日 实验类型： 设计型 实验成绩：**

# 实验二 继承

# 一、目的与任务

（1）掌握利用单继承和多继承的方式定义派生类的方法。

（2）深刻理解在各种继承方式下构造函数和析构函数的执行顺序。

（3）理解和掌握公有继承、私有继承和保护继承对基类成员的访问机制。

（4）理解虚基类的目的和作用。

## 二、实验内容

**（1）调试如下程序**

#include<iostream.h>

class A

{

public :

A(const char \*s) { cout << s << endl; }

~A() {}

};

class B : virtual public A

{

public :

B(const char \*s1, const char \*s2) : A( s1 ) { cout << s2 << endl; }

};

class C : virtual public A

{

public :

C(const char \*s1, const char \*s2):A(s1) { cout << s2 << endl; }

};

class D : public B, public C

{

public :

D( const char \*s1,const char \*s2,const char \*s3,const char \*s4 ):

B( s1, s2 ), C( s1, s3 ), A( s1 )

{ cout << s4 << endl; }

};

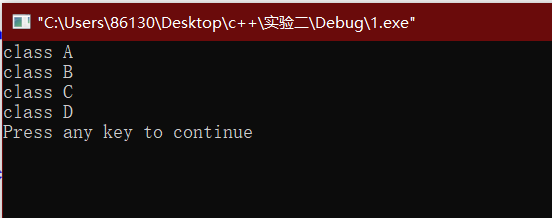
void main()

{

D \*ptr = new D( "class A", "class B", "class C", "class D" );

delete ptr;

}



（2）定义一个Rectangle类，它包含两个数据成员length和width，以及用于求长方形面积的成员函数。再定义Rectangle的派生类Rectangular，它包含一个新数据成员height和用来求长方体体积的成员函数。在main函数中，使用两个类，求某个长方形的面积和某个长方体的体积。

#include<iostream.h>

class re

{

protected:

int l;int w;

public:

int si;

void getlw(){cout<<"输入长、宽：";cin>>l>>w;}

int gets(){return si;}

void makes()

{

si=l\*w;

}

};

class regl:public re // le wi

{

protected:

int h;

public:

int vl;

void geth()

{

cout<<"高：";

cin>>h;

};

int getv(){return vl;};

void makev()

{

makes();

vl=gets()\*h;

}

};

void main()

{

re d;

regl c;

d.getlw ();

d.makes ();

cout<<"面积为："<<d.gets()<<endl;

c.geth ();

c.makev ();

cout<<"体积为:"<<c.getv ()<<endl;

}

（3）假设某销售公司有一般员工、销售员工和销售经理。月工资的计算办法是：

一般员工月薪=基本工资；

销售员工月薪=基本工资+销售额\*提成率；

销售经理月薪=基本工资+职务工资+销售额\*提成率。

编写程序，定义一个表示一般员工的基类Employee，它包含3个表示员工基本信息的数据成员：编号number、姓名name和基本工资basicSalary；

由Employee类派生销售员工Salesman类，Salesman类包含2个新数据成员：销售额sales和静态数据成员提成比例commrate；

再由Salesman类派生表示销售经理的Salesmanager类。Salesmanager类包含新数据成员：岗位工资jobSalary。

为这些类定义初始化数据的构造函数，以及输入数据input、计算工资pay和输出工资条print的成员函数。

设公司员工的基本工资是2000元，销售经理的岗位工资是3000元，提成率=5/1000。在main函数中，输入若干个不同类型的员工信息测试你的类结构。

#include<iostream.h>

class em

{

protected:

int number;

int basicsalary;

char name[10];

public:

em(){basicsalary=2000;}

void pay(){cout<<"姓名、编号:"<<endl;cin>>name>>number;}

void print(){cout<<"姓名："<<name<<"编号："<<number<<"工资："<<basicsalary<<endl;}

};

class s:public em

{

protected:

int sales;

static double com;

public:

s(){};

int salary;

void pay()

{

cout<<"姓名、编号、销售额："<<endl;

cin>>name>>number>>sales;

salary=basicsalary+(sales\*com);

}

void print(){cout<<"姓名："<<name<<"编号："<<number<<"工资："<<salary<<endl;}

};

double s::com=0.005;

class sm:public s

{

protected:

int jobsalary;

public:

sm(){jobsalary=3000;};

void pay()

{

cout<<"姓名、编号、销售额："<<endl;

cin>>name>>number>>sales;

salary=basicsalary+jobsalary+(sales\*com);

}

void print(){cout<<"姓名："<<name<<"编号："<<number<<"工资："<<salary<<endl;}

};

void main()

{

em obj1;

s obj2;

sm obj3;

cout<<"一般员工："<<endl;

obj1.pay();

obj1.print();

cout<<"销售员工："<<endl;

obj2.pay();

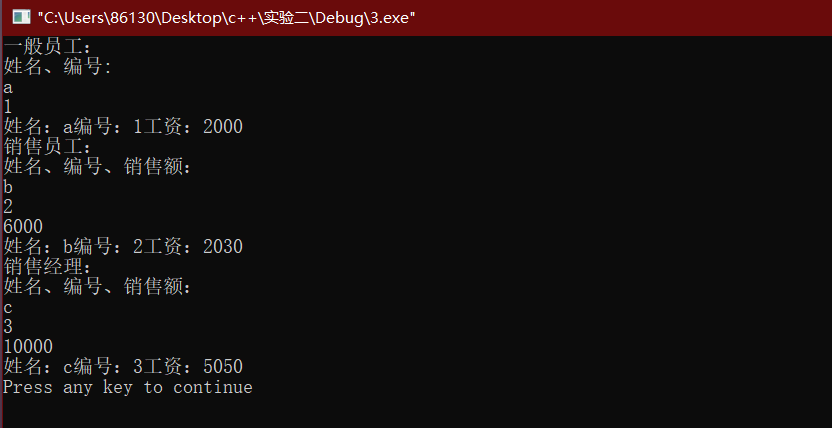
obj2.print();

cout<<"销售经理："<<endl;

obj3.pay();

obj3.print();

}



P306

#include<iostream.h>

class base

{

public :

void get(int i,int j,int k,int l)

{a=i;b=j;x=k;y=l;}

void p()

{

cout<<"a="<<a<<'\t'<<"b="<<b<<'\t'<<"x="<<x<<'\t'<<"y="<<y<<endl;

}

int a,b;

protected:

int x,y;

};

class A:public base

{

public:

void get(int i,int j,int k,int l)

{

base obj3;

obj3.get(50,60,70,80);

obj3.p();

a=i;b=j;x=k;y=l;

u=a+b+obj3.a;

v=y-x+obj3.b;

}

void p()

{

cout<<"a="<<a<<'\t'<<"b="<<b<<'\t'<<"x="<<x<<'\t'<<"y="<<y<<endl;

cout<<"u="<<u<<'\t'<<"v="<<v<<endl;

}

private:

int u,v;

};

int main()

{

base obj1;

A obj2;

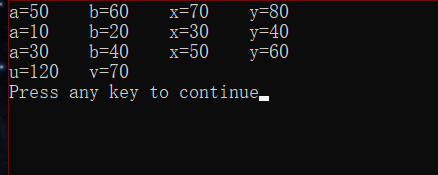
obj1.get(10,20,30,40);

obj2.get(30,40,50,60);

obj1.p();

obj2.p();

}



P307

#include<iostream.h>

class A

{

public :

A(int i,int j){ a=i; b=j ;}

void add(int x,int y){ a+=x;b+=y;}

void show(){cout<<"("<<a<<")\t("<<b<<")\n";}

private:

int a,b;

};

class B:public A

{

public:

B(int i,int j,int m,int n):A(i,j),x(m),y(n){}

void show(){ cout <<"("<<x<<")\t("<<y<<")\n";}

void fun(){ add(3,5); }

void ff(){ A::show();}

private:

int x,y;

};

int main()

{

A a(1,2);

a.show();

B b(3,4,5,6);

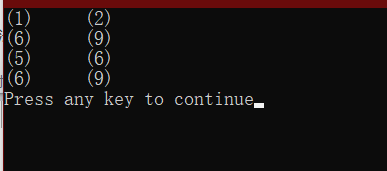
b.fun();

b.A::show();

b.show();

b.ff();

}



P309

#include<iostream.h>

#include<string>

class Person

{

protected:

const char\*name;

const char\*idNumber;

public:

Person(const char\*n,const char\*i)

{

name=n;

idNumber=i;

}

void Print()const

{

cout<<"Name:"<<name<<"\n\tidNumber:"<<idNumber<<endl;

}

};

class Teacher:public Person

{

private:

const char\*Title;

int Wage;

public:

Teacher(const char\*m,const char\*j,const char\*a, int b):Person(m,j)

{

name=m;

idNumber=j;

Title=a;

Wage=b;

}

void Print()const

{

cout<<"Name:"<<name<<"\n\tidNumber:"<<idNumber<<"\n\tTitle:"<<Title<<" Wage:"<<Wage<<endl;

}

};

void main()

{

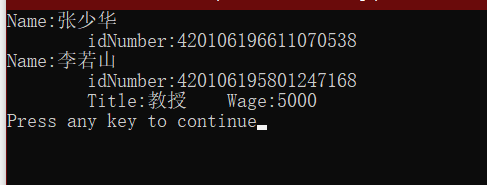
Person p("张少华","420106196611070538");

Teacher t("李若山","420106195801247168","教授",5000);

p.Print();

t.Print();

}



## 三、通过实验回答以下问题

（1）虚继承的作用是什么？

虚继承可以从不同途径继承来的同一基类，会在子类中存在多份拷贝。

（2）在继承体系结构中，当创建派生类对象时，基类的构造函数的参数通过什么方式获取？

在派生类中创建一个构造函数，然后用该构造函数调用基类的构造函数并且向构造函数传递初始值。