第六次实验

继承－派生

1:源代码：

#include <iostream>

using namespace std;

class planet {

protected:

double distance;

int revolve;

planet(){

distance=0;

revolve=0;

}

planet(double d, int r){

distance=r;

revolve=d;

}

};

class earth:public planet{

double circumference;

public:

earth(double d,int r):planet(d,r){

circumference=2\*d\*3.1416;

}

void show();

};

void earth::show(){

cout<<"Circumference:"<< circumference<<endl;

cout<<"Distance:"<<distance<<endl;

cout<<"Revolve:"<<revolve<<endl;

}

int main( void) {

double dis;

cin >> dis;

earth ob(dis,365);

ob.show( );

return 0;

}

运行结果：

**93000000**

**Circumference:5.84338e+08**

**Distance:365**

**Revolve:93000000**

**Program ended with exit code: 0**

**2:**

**源代码：**

#include <iostream>

#include <cstring>

using namespace std;

class Person{

protected:

char \*name;

public:

Person(const char \* s){

name=new char[strlen(s)+1];

strcpy(name, s);

printf("Base constructoer\n");

}

~Person(){

if(!name)

delete [] name;

printf("Base destructor\n");

}

void show(){

printf("\tName:%s\n",name);

}

};

class Student:public Person{

private:

int score;

int age;

char \*major;

public:

Student(char \*name,int s,int a,char \*m):Person(name){

major=new char[strlen(m)+1];

strcpy(major, m);

score=s;

age=a;

printf("Derive constructor\n");

}

~Student(){

if(!major)

delete [] major;

printf("Derive destructor\n");

}

void show(){

Person::show();

cout<<"\tMajor:"<<major<<endl;

cout<<"\tAge:"<<age<<endl;

cout<<"\tScore:"<<score<<endl;

}

};

int main(int argc, const char \* argv[]) {

// insert code here...

Student obj1("Tom",76,19,"Computer scense");

obj1.show();

return 0;

}

运行结果：

**Base constructoer**

**Derive constructor**

**Name:Tom**

**Major:Computer scense**

**Age:19**

**Score:76**

**Derive destructor**

**Base destructor**

**Program ended with exit code: 0**

**3:**

**源代码：**

#include <iostream>

using namespace std;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Base-empolyee\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

class empolyee{

protected:

char \*name; //Name

int individualEmpNo; //Personal empolyee No.

int grade; //Grades

float accumPay; //Accumulata account to this object

static int empolyeeNo; //The number of this company

public:

//empolyee(char \*s,int i,int g,int ac);//Constructor of class empolyee to initial the data memeber

empolyee();

~empolyee(){

if(!name)

delete [] name;

empolyeeNo--;

}

void pay(); //Pay for this one

void display(); //show the informations of this guy

void promote(int increment);

void dispalayStatus();

};

/\*empolyee::empolyee(char \*s,int i,int g,int ac){

name=new char[strlen(s)+1];

strcat(name, s);

individualEmpNo=i;

grade=g;

accumPay=ac;

empolyeeNo++;

}\*/

empolyee::empolyee()

{

char namestr[50];

//输人雇员姓名时首先临时存放在namestr中

cout<<"请输入下一个雇员的姓名:";

cin>>namestr;

name=new char[strlen(namestr)+1];

//动态申请用于存放姓名的内存空间

strcpy(name, namestr);

//将临时存放的姓名复制到 name

individualEmpNo= empolyeeNo++;

//新输人的员工，其编号为目前最大编号加1

grade=1; //级别初值为1

accumPay=0.0; //月薪总额初值为0

}

void empolyee ::promote(int increment)

{

grade += increment ;

//升级，提升的级数由 increment指定

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*empolyee-Derived-technician\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

class technician:public empolyee{

private:

float hourlyRate;

int workHours;

public:

technician();

~technician(){}

void pay();

void displayStatus();

};

technician :: technician()

{ hourlyRate=100; //每小时酬金100元

}

void technician::pay()

{ cout<<"请输入"<<name<<"本月的工作时数:";

cin>>workHours;

accumPay=hourlyRate\* workHours;

//计算月薪，按小时计酬

cout<<"兼职技术人员"<<name<<"编号"<<individualEmpNo

<<"本月工资"<<accumPay<<endl;

}

void technician::displayStatus()

{ cout<<"兼职技术人员"<<name<<"编号" << individualEmpNo << "级别为" <<grade<<

"级，已付本月工资"<<accumPay<<endl;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*empolyee-Derived-Base-manager\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

class manager:virtual public empolyee{

protected:

float monthlyPay;

public:

manager();

~manager(){}

void pay();

void displayStatus();

};

manager::manager()

{

monthlyPay=8000; //固定月薪8000元

}

void manager :: pay()

{

accumPay=monthlyPay; //月薪总额即固定月薪数

cout<<"经理"<<name<<"编号"<<individualEmpNo

<<"本月工资"<<accumPay<<endl;

}

void manager :: displayStatus()

{

cout<<"经理"<<name<<"编号"<<individualEmpNo

<<"级别为"<<grade<<"级，已付本月工资"<<accumPay<<endl;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*empolyee-Derived-Base-saleaman\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

class salesman:virtual public empolyee{

protected:

float CommRate;

float sales;

public:

salesman();

~salesman(){}

void pay();

void displayStatus();

};

salesman::salesman()

{

CommRate=0.04; //销售提成比例4％

}

void salesman::pay()

{

cout<<"请输入"<<name<<"本月的销售额:";

cin>>sales;

accumPay=sales\*CommRate; //月薪＝销售提成

cout<<"推销员"<<name<<"编号"<<individualEmpNo

<<"本月工资"<<accumPay<<endl;

}

void salesman::displayStatus()

{ cout<<"推销员"<<name<<"编号"<<individualEmpNo

<<"级别为"<<grade<<"级，已付本月工资"<<accumPay<<endl;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*manager&&saleman-Derived-salesmanager\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

class salesmanager:public manager,public salesman{

public:

salesmanager();

~salesmanager(){}

void pay();

void displayStatus();

};

salesmanager ::salesmanager()

{ monthlyPay=5000;

CommRate= 0.005;

}

void salesmanager ::pay()

{

cout<<"请输入"<<empolyee ::name<<"所管辖部门本月的销售总额:";

cin>>sales;

accumPay=monthlyPay+CommRate\*sales; //月薪＝固定月薪十销售提成

cout<<"销售经理"<<name<<"编号"<<individualEmpNo

<<"本月工资"<<accumPay<<endl;

}

void salesmanager::displayStatus()

{ cout<<"销售经理"<< name<<"编号"<<individualEmpNo

<<"级别为"<<grade<<"级，已付本月工资"<<accumPay<<endl;

}

int empolyee::empolyeeNo=0;

int main(int argc, const char \* argv[]) {

// insert code here...

manager m1;

technician t1;

salesmanager sm1;

salesman s1;

m1.promote(3); //经理m1提升3级

m1.pay(); //计算m1月薪

m1.displayStatus(); //显示m1信息

t1.promote(2); //t1提升2级

t1.pay(); //计算t1月薪

t1.displayStatus(); //显示tl信息

sm1.promote(2); //sml提升2级

sm1.pay(); //计算sml月薪

sm1.displayStatus(); //显示sml信息

s1.pay(); //计算s1月薪

s1.displayStatus(); //显示s1信息

return 0;

}

**运行结果：**

**请输入下一个雇员的姓名:lisi**

**请输入下一个雇员的姓名:wangeu**

**请输入下一个雇员的姓名:zhangsan**

**请输入下一个雇员的姓名:liyu**

**经理LISI编号0本月工资8000**

**经理LISI编号0级别为4级，已付本月工资8000**

**请输入WANGEU本月的工作时数:25**

**兼职技术人员WANGEU编号1本月工资2500**

**兼职技术人员WANGEU编号1级别为3级，已付本月工资2500**

**请输入ZHANGSAN所管辖部门本月的销售总额:4325**

**销售经理ZHANGSAN编号2本月工资5021.62**

**销售经理ZHANGSAN编号2级别为3级，已付本月工资5021.62**

**请输入LIYU本月的销售额:24**

**推销员LIYU编号3本月工资0.96**

**推销员LIYU编号3级别为1级，已付本月工资0.96**

**Program ended with exit code: 0**

**作业7：**

**源代码：**

#include <iostream>

using namespace std;

class A{

protected:

int m,n;

public:

void set(int a,int b){

m=a;

n=b;

}

void show(){

cout<<m<<" "<<n<<"\n";

}

};

class B:public A{

int s;

public:

void sets(){

s=n\*m;

}

void shows(){

cout<<s<<endl;

}

};

int main(int argc, const char \* argv[]) {

// insert code here...

B obj;

obj.set(2,3);

obj.show();

obj.sets();

obj.shows();

return 0;

}

**运行结果：**

**2 3**

**6**

**Program ended with exit code: 0**

**作业8:**

**源代码：**

#include <iostream>

using namespace std;

class planet {

protected:

double distance;

int revolve;

planet(){

distance=0;

revolve=0;

}

planet(double d, int r){

distance=r;

revolve=d;

}

};

class earth:public planet{

double circumference; public:

earth(double d,int r):planet(d,r){

circumference=2\*d\*3.1416;

}

void show();

};

void earth::show(){

cout<<"Circumference:"<< circumference<<endl;

cout<<"Distance:"<<distance<<endl;

cout<<"Revolve:"<<revolve<<endl;

}

int main( void) {

double dis;

cin >> dis;

earth ob(dis,365);

ob.show( );

return 0;

}

运行结果：

**93000000**

**Circumference:5.84338e+08**

**Distance:365**

**Revolve:93000000**

**Program ended with exit code: 0**