

XI`AN TECHNOLOGICAL UNIVERSITY

实验报告

实验课程名称 继承（三）

专 业： 计算机科学与技术

班 级： 16060104

姓 名： 杨 旭

学 号： 16060104117

实验学时： 2

指导教师： 刘白林

成 绩：

2017 年 6 月 18 日

西安工业大学实验报告

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 专业 | 计算机科学与技术 | 班级 | 计科四班 | 姓名 | 杨旭 | 学号 | 16060104117 |
| 实验课程 | C++ | 指导教师 | 刘白林 | 实验日期 |  | 同实验者 |  |
| 实验项目 | 继承（三） | | | | | | |
| 实验设备及器材 | 微型 | | | | | | |

一、实验目的

1．理解多继承的概念，熟悉多继承的定义及应用；

2．理解多继承方式下的二义性产生原因，熟悉解决二义性的方法；

3．进一步熟悉继承的综合应用。

二、实验学时

课内实验：2课时 课外练习：2课时

三 本实验涉及的新知识

㈠ 多继承的概念

如果一个派生类由二个或二个以上的基类派生得到，称为多重继承（多继承）

㈡ 多继承派生类的定义

格式：

**class 派生类名:继承方式1 , 基类名1, 继承方式2, 基类名2, …**

**{ 派生类新定义的成员 }**

㈢ 多继承派生类的构造函数

1．定义

格式：

**构造函数名(总参数表):基类名1(参数表1), 基类名2(参数表2),…**

**{ 构造函数的函数体 }**

2．说明

⑴ 多继承的构造函数的总参数表必须提供所有基类构造函数的值及新增数据成员的值。

⑵ 定义派生类对象时，多继承方式下构造函数的执行顺序为：先执行所有基类的构造函数（按排列顺序）后执行派生类的构造函数。

⑶ 释放派生类对象时，析构函数的执行顺序与构造函数的执行顺序相反。

㈣ 多继承派生类中为数据成员提供值的函数

在多继承方式下，为数据成员提供值的函数必须同时为所有基类的数据成员提供值。

㈤ 多继承的二义性

1．同名函数产生的二义性

在多重继承方式下，如果在多个基类中有同名成员函数，可能出现二义性问题。解决办法是在调用函数时使用类名及作用域运算符。

2．间接基类产生的二义性

在多重继承下，如果一个派生类由二个或以上基类派生得到，而这些基类又是由一个基类派生得到的，也会产生二义性。解决办法是使用虚基类，即在定义直接基类时，将间接基类说明为虚基类。

定义格式：

**class 直接基类名 : virtual 继承方式 间接基类名**

**{ 派生类新定义的成员 }**

四、实验内容

㈠ 验证及认知实验

按要求调试下列程序，并回答相关问题。

程序1（exp\_601.cpp）

#include<iostream.h>

class X

{ int x;

public :

X(int a=0){x=a;}//基类X的构造函数

void set\_x(int a){x=a;}

void show(void)

{cout<<" x="<<x;}

};

class Y

{ int y;

public :

Y(int b=0){y=b;}//基类Y的构造函数

void set\_y(int b){y=b;}

void show(void)

{cout<<" y="<<y;}

};

class Z:public X,public Y

//class Z:public X,private Y

{ int z;

public :

Z(int a=0,int b=0,int c=0): X(a),Y(b)

{z=c;} //派生类Z的构造函数

void set\_xyz(int a,int b,int c)

{ set\_x(a); set\_y(b);z=c;}

void show\_z(void)

{cout<<" z="<<z;}

void print(void)

{ show(); //显示x的值

show(); //显示y的值

show\_z(); //显示z的值

}

};

void main()

{ Z obj1(3,4,5),obj2;

obj2.set\_xyz(10,20,30);

obj1.print();cout<<endl;

obj2.print();cout<<endl;

}

问题：

1. 编译该程序，会出现编译错误，其原因是在定义Z类的print ( )成员函数中的show( )函数调用时出现了 二义性 。解决的方法是：
2. 将“show(); //显示x的值”中的“show( )”改为 X::show()
3. 将“show(); //显示y的值”中的“show( )”改为 Y::show()
4. 调试成功后的输出结果是：



1. 成员函数void set\_xyz(int a,int b,int c)中“set\_x(a)”的作用是 对基类X的构造函数进行初始化 ，“set\_y(b)”的作用是 对基类Y的构造函数进行初始化 。

程序2（exp\_602.cpp）

#include<iostream.h>

class base

{ private:

int a;

public:

base(int x)

{ a=x;

cout<<"Constructing base a="<<a<<endl;

}

};

class base1:public base

{ private:

int b;

public:

base1(int x1,int x2):base(x1)

{ b=x2;

cout<<"Constructing base1 b="<<b<<endl;

}

};

class base2:public base

{ private:

int c;

public:

base2(int x1,int x2):base(x1)

{ c=x2;

cout<<"Constructing base2 c="<<c<<endl;

}

};

class derived: public base1,public base2

{ private:

int d;

public:

derived(int x1,int x2,int x3,int x4):base1(x1,x2),base2(x1,x3)

{ d=x4;

cout<<"Constructing derived d="<<d<<endl;

}

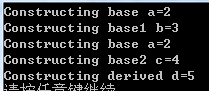
};

void main(void)

{ derived obj(2,3,4,5);}

问题：

1. 编译运行程序的输出结果是：



1. 从输出结果中可看出，间接基类“base”被调用了两次，产生原因是“derived”的两个直接基类“base1”、“base2”拥有共同基类“base”，产生了 二义性 。

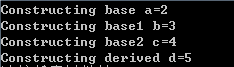
⑻ 将程序中“class base1:public base”改为“class base1:virtual public base”

“class base2:public base”改为“class base2:virtual public base”

“derived(int x1,int x2,int x3,int x4):base1(x1,x2),base2(x1,x3)”改为

“derived(int x1,int x2,int x3,int x4):base(x1),base1(x1,x2),base2(x1,x3)”

重新编译运行程序的输出结果是：



1. “virtual public base”表示间接基类“base”为 虚基类 。

㈡ 知识应用实验

按要求调试程序（完善类的定义及主程序）。

1．题目：通过继承方式处理学生成绩。

2．内容

⑴ 类的定义：

人“person”类（person.h中定义）：包括姓名、性别、出生日期（日期“date”本身是一个类，包括年、月、日三个成员——定义在date.h中）三个数据成员；

学生“student”类（student.h中定义)，公有继承person类，新增学号、系、进校时间（date对象）三个数据成员；

成绩score类（score.h中定义）：新增课程科数、成绩数组数据成员。

⑵ 成绩处理：源程序存放在“exp\_603.cpp”中。

3．要求

⑴ 完善类的定义；

⑵ 程序调试办法：每建好一个类，均在main()函数中进行测试成功后再定义下一个类，测试程序以“exp\_test.cpp”存盘；

⑶ 所有类定义测试后再调试、完善成绩处理程序。

程序：

class date

{

private:

int year, month, day; //年、月、日三个私有成员

public:

date(void)

{

year = 1980; month = 1; day = 1;

}

void set\_date(int y, int m, int d)

{

year = y;

month = m;

day = d;

} //完成成员函数的定义

int get\_year(void)

{

return year;

} //完成成员函数的定义

int get\_month(void)

{

return month;

} //完成成员函数的定义

int get\_day(void)

{

return day;

} //完成成员函数的定义

};

#include <iostream>

#include<string.h>

#include<iomanip>

#include"date.h"

using namespace std;

class person

{

protected:

char name[11];

char sex[2];

date birthday;

public:

person(void); //无参构造函数

void set\_person(char \*na, char \*s, int y, int m, int d);

char \*get\_name(void)

{

return name;

} //完成成员函数的定义

char \*get\_sex(void)

{

return sex;

} //完成成员函数的定义

int get\_year(void)

{

return birthday.get\_year();

} //完成成员函数的定义

int get\_month(void)

{

return birthday.get\_month();

}

int get\_day(void)

{

return birthday.get\_day();

} //完成成员函数的定义

void print(void);

};

person::person(void) //无参构造函数

{

strcpy\_s(name,strlen("无名氏")+1, "无名氏");

strcpy\_s(sex,strlen("男")+1, "男");

birthday.set\_date(1980, 1, 1);

}

void person::set\_person(char \*na, char \*s, int y, int m, int d)

{

strcpy\_s(name,strlen(na)+1, na);

strcpy\_s(sex,strlen(s)+1, s);

birthday.set\_date(y, m, d);

} //完成成员函数的定义

void person::print(void)

{

cout << "姓名:" << name << endl;

cout << "性别:" << sex << endl;

cout << "出生日期:" << birthday.get\_year() << "年";

cout << birthday.get\_month() << "月";

cout << birthday.get\_day() << "日" << endl;

}

#include"person.h"

class student :public person

{

public:

int id;

char department[20];

date enterdate;

public:

student(void);

void set\_student(int n, char \*s, int y, int m, int d);

int get\_id(void)

{

return id;

} //完成成员函数的定义

char \*get\_department(void)

{

return department;

} //完成成员函数的定义

int get\_enteryear(void)

{

return enterdate.get\_year();

} //完成成员函数的定义

int get\_entermonth(void)

{

return enterdate.get\_month();

} //完成成员函数的定义

int get\_enterday(void)

{

return enterdate.get\_month();

} //完成成员函数的定义

void print(void);

void print\_base();

};

student::student(void)

{

strcpy\_s(name,strlen("无名氏")+1, "无名氏");

strcpy\_s(sex,strlen("男")+1,"男" );

birthday.set\_date(1980, 1, 1);

id = 0;

strcpy\_s(department,strlen("计算机")+1, "计算机");

enterdate.set\_date(2000, 9, 1);

}

void student::set\_student(int n, char \*s, int y, int m, int d)

//n、s、y、m、d分别为id、department、enterdate提供值

{

id = n;

strcpy\_s(department,strlen(s)+1, s);

enterdate.set\_date(y, m, d);

} //完成成员函数的定义

void student::print(void)

{

cout << "学号:" << id << endl;

person::print();

cout << "系(专业):" << department << endl;

cout << "进校日期:" << enterdate.get\_year() << "年";

cout << enterdate.get\_month() << "月";

cout << enterdate.get\_day() << "日" << endl;

}

void student::print\_base()

{

cout << setw(8) << get\_id();

cout << setw(12) << get\_name();

cout << setw(4) << get\_sex();

cout << setw(6) << get\_year() << "-" << get\_month() << "-" << get\_day();

cout << setw(20) << get\_department();

cout << setw(6) << get\_enteryear() << "-" << get\_entermonth();

cout << "-" << get\_enterday() << endl;

}

#pragma once

#include"student.h"

const int M = 10;

class score :public student

{

private:

float sc[M], aver;

int m;

public:

score(void); //无参构造函数

void set\_score(float x[], int n); //提供成绩

float get\_score(int i) //得到第i科成绩

{

return sc[i];

} //完成成员函数的定义

float get\_aver(void) //得到平均成绩

{

return aver;

} //完成成员函数的定义

void print(void);

void print\_score(void);

};

score::score(void) //无参构造函数

{

strcpy\_s(name,strlen("无名氏")+1, "无名氏");

strcpy\_s(sex,strlen("男")+1, "男");

birthday.set\_date(1980, 1, 1);

id = 0;

strcpy\_s(department, strlen("计算机")+1,"计算机");

enterdate.set\_date(2000, 9, 1);

int i;

m = M;

for (i = 0; i<m; i++)

sc[i] = 0;

aver = 0;

}

void score::set\_score(float x[], int n) //提供成绩:完成成员函数的定义

{

int i; float sum = 0;

m = n;

for (i = 0; i<m; i++)

{

cin >> x[i];

sum = sum + x[i];

}

aver = sum / m;

}

void score::print(void) //重载输出print()

{

student::print();

int i;

for (i = 0; i<m; i++)

cout << " " << sc[i];

cout << " " << aver << endl;

}

void score::print\_score(void)

{

int j;

cout << setw(8) << get\_id();

cout << setw(12) << get\_name();

for (j = 0; j<m; j++)

cout << setw(6) << get\_score(j);

cout << " " << setw(6) << get\_aver() << endl;

}

//成绩处理程序：保存在exp\_603.cpp中

#include<iostream>

#include<iomanip>

#include"score.h"

void input\_base(score \*p, int n); //学生基本数据输入

void input\_score(score \*p, int n, int m); //学生成绩输入

void print\_base(score \*p, int n); //学生基本数据输出

void print\_score(score \*p, int n, int m); //学生成绩输出

score &average(score \*p, int n, int m); //普通函数：平均成绩计算

void sort(score \*p, int n, int m);//普通函数：按平均成绩排序

void main()

{

int n, m;

cout << "学生人数："; cin >> n;

cout << "考试科数："; cin >> m;

score \*p, aver;

p = new score[n]; //动态分配内存单元——动态数组

if (p == NULL)

{

cout << "内存分配失败" << endl;

return;

}

int ch;

do

{

cout << "\n\n 请选择:\n";

cout << " 1. 输入学生基本数据\n";

cout << " 2. 输入学生成绩\n";

cout << " 3. 计算课程平均成绩\n";

cout << " 4. 输出学生基本数据\n";

cout << " 5. 输出学生成绩\n";

cout << " 6. 按平均成绩排序\n";

cout << " 0. 退出\n";

cout << "\n 输入你的选择:";

cin >> ch;

switch (ch)

{

case 1: input\_base(p, n); break;

case 2: input\_score(p, n, m); break;

case 3: aver = average(p, n, m); break;

case 4: print\_base(p, n); break;

case 5: print\_score(p, n, m); aver.print\_score(); break;

case 6: sort(p, n, m); break;

case 0: break;

}

} while (ch);

delete[]p; //释放内存

}

void input\_base(score \*p, int n) //学生基本数据输入

{

int i, id, y1, m1, d1, y2, m2, d2;

char name[11], sex[3], dpt[20];

cout << "\n 请输入学生基本数据:";

for (i = 0; i<n; i++)

{

cout << "第" << i + 1 << "个学生:\n";

cout << "学号:"; cin >> id;

cout << "姓名:"; cin >> name;

cout << "性别:"; cin >> sex;

cout << "出生年:"; cin >> y1;

cout << "出生月:"; cin >> m1;

cout << "出生日:"; cin >> d1;

cout << "所在系:"; cin >> dpt;

cout << "入学年:"; cin >> y2;

cout << "入学月:"; cin >> m2;

cout << "入学日:"; cin >> d2;

p[i].set\_person(name, sex, y1, m1, d1);//完成函数的调用

p[i].set\_student(id, dpt, y2, m2, d2);//完成函数的调用

}

}

void input\_score(score \*p, int n, int m)

{

int i, j; float x[M];

for (i = 0; i<n; i++)

{

cout << p[i].get\_id() << p[i].get\_name() << "的成绩：" << endl;

for (j = 0; j<m; j++)

{

cout << "第" << j + 1 << "科成绩：";

cin >> x[j];

}

p[i].set\_score(x, m);//完成函数的调用

}

}

void print\_base(score \*p, int n)

{

int i;

for (i = 0; i<n; i++)

p[i].print\_base();

cout << endl;

}

void print\_score(score \*p, int n, int m)

{

int i;

for (i = 0; i<n; i++)

p[i].print\_score();

}

score &average(score \*p, int n, int m) //用返回引用的方法

{

int i, j; float s[M] = { 0 };

static score aver; //返回的对象必须是静态的

for (j = 0; j<m; j++)

{

for (i = 0; i<n; i++)

s[j] = s[j] + p[i].get\_score(j);

s[j] = s[j] / n;

}

aver.set\_person("平均成绩", " ", 0, 0, 0);

aver.set\_score(s, n);//完成函数的调用

return aver;

}

void sort(score \*p, int n, int m)//选择法排序：完成空白处的内容

{

score t; float a;

int i, j, k;

for (i = 0; i<n - 1; i++)

{

a = p[i].get\_aver(); k = i;

for (j = i + 1; j<n; j++)

if (p[j].get\_aver()>p[i].get\_aver())

{

a = p[j].get\_aver(); k = j;

}

if (i != k)

{

t = p[i]; p[i] = p[k]; p[k] = t;

}

}

}㈢ 程序设计实验

编程对日期时间型数据进行处理。要求：

1、定义日期型“Date”类，保存在“date.h”中（成员函数包括日期赋初值，设置日期，读取年、月、日，显示年、月、日及判断是否闰年等）。

2、定义时间型“Time”类，保存在“Time.h”中（与实验一中的类相同）。

3、定义日期时间型“DateTime”类保存在“datetime.h”中：

⑴ 公有继承Date类及Time类；

⑵ 不增加数据成员；

⑶ 增加日期时间处理成员函数，包括初始化日期时间、设置日期时间、显示日期时间等。

4、编写main()函数保存在“exp\_603.cpp”中，内容自定。

#pragma once

//1、定义日期型“Date”类，保存在“date.h”中（成员函数包括日期赋初值，设置日期，读取年、月、日，显示年、月、日及判断是否闰年等）。

#include <iostream>

using namespace std;

class date

{

protected:

int year;

int month;

int day;

public:

void set\_date(int y, int m, int d);

void print\_date();

};

void date::set\_date(int y, int m, int d)

{

year = y;

month = m;

day = d;

}

void date::print\_date()

{

if (year % 4 != 0 || year % 100 != 0 && year % 400 == 0)

{

cout << year << '-' << month << '-' << day << "（闰年）" << endl;

}

else

{

cout << year << '-' << month << '-' << day;

}

}

#include "date.h"

class time:public date

{

private:

int hour;

int minute;

int second;

public:

time();

void set\_time(int h, int m, int s);

int get\_hour();

int get\_minute();

int get\_second();

void print\_time();

};

time::time()

{

hour = 0;

minute = 0;

second = 0;

}

void time::set\_time(int h, int m, int s)

{

hour = h;

minute = m;

second = s;

}

int time::get\_hour()

{

return hour;

}

int time::get\_minute()

{

return minute;

}

int time::get\_second()

{

return second;

}

void time::print\_time()

{

if (hour > 24 || hour < 0 || minute < 0 || minute>60 || second < 0 || second>60)

cout << "输入信息有误！";

else if(hour >= 12)

{

hour = hour - 12;

cout << hour << '-' << minute << '-' << second << "(P.M.)";

}

else

{

cout << hour << '-' << minute << '-' << second << "(A.M.)";

}

}

#include"time.h"

void main()

{

int y, m, d;

int h, mi, s;

cout << "请输入 年、月、日：" << endl;

cin >> y;

cin >> m;

cin >> d;

cout << "请输入 时、分、秒；" << endl;

cin >> h;

cin >> mi;

cin >> s;

time A;

A.set\_date(y, m, d);

A.set\_time(h, mi, s);

A.print\_date();

A.print\_time();

cout << endl;}

;