

中国矿业大学计算机学院实验报告

| | | | |
|--|------------------------------|------|----------|
| 课程名称 | 高级语言程序设计 | | |
| 实验名称 | 高级语言程序设计实验七(12月10日5-8节)-薛猛老师 | | |
| 班级 | 信息安全 19-01 班 | 姓名 | 许万鹏 |
| | | 学号 | 05191643 |
| 仪器组号 | 66 | 实验日期 | 12月10日 |
| 实验报告要求：1. 实验目的 2. 实验内容（题目描述，源代码，运行截图，调试情况） 3. 实验体会 | | | |
| <p>一、实验目的</p> <p>1) 掌握类的声明、对象定义以及类对象的访问权限；</p> <p>2) 掌握访问类的数据成员和成员函数的方法；</p> <p>3) 熟悉构造函数和析构函数的定义及使用。</p> <p>二、实验内容</p> <p>1、第一题</p> <p>1.1 题目描述</p> <p>【题目描述】</p> <p>利用继承性与派生类来管理学生教师档案:由 Person(人员)类出发(作为基类), 派生出 Student(学生)及 Teacher(教师)类;而后再由 Student(学生)类出发(作为基类), 派生出 GraduateStudent(研究生)类。可假定这几个类各自具有的数据成员为:</p> <p>Person(人员)类: 姓名、性别、年龄;</p> <p>Student(学生)类: 姓名、性别、年龄、学号、系别;</p> <p>Teacher(教师)类: 姓名、性别、年龄、职称、担任课程;</p> <p>GraduateStudent(研究生)类: 姓名、性别、年龄、学号、系别、导师。</p> <p>为简化起见, 每个类可只设立构造函数以及显示类对象数据的成员函数 Print。而后编制简单的主函数, 说明上述有关的类对象, 并对其类成员函数进行简单使用(调用)</p> <p>【样例输出】</p> <pre>== per1.Display() => name,age,sex sun 42 M == stu1.Display() => name,age,sex,Reg_Number,department guo 22 F 1001 comp</pre> | | | |

```
== teach1.Display() => name,age,sex,course,post  
fang 38 M english professor  
== gStu.Display() => name,age,sex,Reg_Number,department,advi  
sor  
wu 25 M 1021 comp wei
```

1.2 源代码

```
#include<iostream>  
#include<string>  
using namespace std;  
  
class Person{  
    protected:  
        string name;  
        string sex;  
        int age;  
    public:  
        Person(){}  
        Person(string n,int a,string s)  
        {  
            name=n;age=a;sex=s;  
        }  
        void Display()  
        {  
            cout << name << " " << age << " " << sex;  
        }  
};  
  
class Student:public Person{  
    protected:  
        int Reg_Number;  
        string department;  
    public:  
        Student(){}  
        Student(string n,int a,string s,int r,string d)  
        {  
            name=n;age=a;sex=s;Reg_Number=r;department=d;  
        }  
        void Display()  
        {  
            Person::Display();  
            cout << " " << Reg_Number << " " << department;  
        }  
};
```

```

class Teacher:public Person{
protected:
    string course;
    string post;
public:
    Teacher(string n,int a,string s,string c,string p):P
erson(n,a,s),course(c),post(p){}
    void Display()
    {
        Person::Display();
        cout << " " << course << " " << post;
    }
};


class GraduateStudent:public Student{
protected:
    string advisor;
public:
    GraduateStudent(){}
    GraduateStudent(string n,int a,string s,int r,string
d,string ad):Student(n,a,s,r,d),advisor(ad){}
    void Display()
    {
        Student::Display();
        cout << " " << advisor;
    }
};

int main()
{
    cout << "== per1.Display() => name,age,sex" << endl;
    Person per1("sun",42,"M");
    per1.Display();cout << endl;
    cout << "== stu1.Display() => name,age,sex,Reg_Number,de
partment" << endl;
    Student stu1("guo",22,"F",1001,"comp");
    stu1.Display();cout << endl;
    cout << "== teach1.Display() => name,age,sex,course,post
" << endl;
    Teacher teach1("fang",38,"M","english","professor");
    teach1.Display();cout << endl;
    cout << "== gStu.Display() => name,age,sex,Reg_Number,de
partment,advisor" << endl;
    GraduateStudent gStu("wu",25,"M",1021,"comp","wei");
    gStu.Display();
}

```

```
    return 0;
}
```

1.3 运行截图



```
问题  输出  调试控制台  终端
> Executing task: D:\Codefield\CODE_Cpp\Cpp_Single\No.7\bin\1.exe <

== per1.Display() => name,age,sex
sun 42 M
== stu1.Display() => name,age,sex,Reg_Number,department
guo 22 F 1001 comp
== teach1.Display() => name,age,sex,course,post
fang 38 M english professor
== gStu.Display() => name,age,sex,Reg_Number,department,advisor
wu 25 M 1021 comp wei
按任意键关闭终端。
```

1.4 调试情况

Accepted, 如上图。

2、第二题

2.1 题目描述

【题目描述】

自定义一个日期时间类 DateTimeType, 它含有类 DateType 与类 TimeType 的类对象作为其数据成员, 并具有所列的其他几个成员函数。而后编制主函数, 说明 DateTimeType 的类对象, 并对其成员函数以及二对象成员所属类的公有成员函数进行使用。

```
class DateTimeType { //自定义的日期时间类 DateTimeType
DateType date; //类 DateType 的类对象 date 作为其数据成员
TimeType time; //类 TimeType 的类对象 time 作为其另一个数据成员
public:
DateTimeType(int y0=1, int m0=1, int d0=1, int hr0=0, int mi
0=0, int se0=0);
//构造函数, 设定 DateTimeType 类对象的日期时间, 并为各参数设置了默认值
DateType& GetDate(){ return date; } //返回本类的私有数据对
象 data
TimeType& GetTime(){ return time; } //返回本类的私有数据对
象 time
void IncrementSecond(int s); //增加若干秒, 注意“进位”问题
void PrintDateTime(); //屏幕输出日期时间对象的有关数据
};
```

注意,每一个 DateTimeType 类对象中总包含有一个 DateType 类对象(对象成员) 以及一个 TimeType 类对象(对象成员), 编制与实现本程序时, 也必须包含 DateType 与 TimeType 自定义类(类型)。

之所以设置了公有的类成员函数 GetDate 与 GetTime, 是为类外如主函数处使用该类的私有数据成员 date 与 time 提供方便(否则的话, 类外无法直接访问该类的私有数据成员)。另外, 两成员函数返回的都为引用, 为的是可将返回对象当作一个独立变量来使用(如可以继续作左值等)。例如, 假设编制了如下形式的主函数:

```
void main() {
DateTimeType dttml(1999,12,31,23,59,59), dttm2;
(dttml.GetDate()).PrintDate(); //调用对象成员所属类的公有成员函数
cout<<endl;
dttml.PrintDateTime(); //调用本派生类的成员函数 PrintDateTime
dttm2.PrintDateTime();
dttml.IncrementSecond(30) ; //调用本派生类成员函数
dttml.PrintDateTime();
}
```

【样例输出】

1999-12-31

1999-12-31 23:59:59

1-1-1 0:0:0

2000-1-1 0:0:29

2.2 源代码

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

class DateType{
protected:
    int y;
    int m;
    int d;
public:
    DateType(int yy=0,int mm=0,int dd=0):y(yy),m(mm),d(dd){}
    void PrintDate()
    {
        cout<<y<<"-"<<m<<"-"<<d;
    }
    void IncrementOneDay()
```

```

{
    int dpm[13]={0,31,28,31,30,31,30,31,31,30,31,30,31};
    if(m==2)
    {
        if((y%4==0 && y%100!=0)||y%400==0)
        {
            if(d==29)
            {
                m++;
                d=1;
            }
            else d++;
        }
        else
        {
            if(d==28)
            {
                m++;
                d=1;
            }
            else d++;
        }
    }
    else
    {
        if(d==dpm[m])
        {
            if(m==12)
            {
                y++;
                m=1;
                d=1;
            }
            else
            {
                m++;
                d=1;
            }
        }
        else d++;
    }
}

void Carrying(int dd)
{

```

```

        for(int i=0;i<dd;i++)
            IncrementOneDay();
    }
};

class TimeType{
protected:
    int hr;
    int mi;
    int se;
public:
    TimeType(int hh=0,int mm=0,int ss=0):hr(hh),mi(mm),se(ss)
    {}
    void PrintTime()
    {
        cout<<hr<<':'<<mi<<':'<<se;
    }
    int Carrying(int ss)
    {
        se+=ss;
        int ctemp=se/60;
        if(ctemp)
        {
            mi+=ctemp;
            se-=ctemp*60;
            ctemp=mi/60;
            if(ctemp)
            {
                hr+=ctemp;
                mi-=ctemp*60;
                ctemp=hr/24;
                if(ctemp)
                {
                    hr-=ctemp*24;
                    return ctemp;
                }
            }
        }
        return 0;
    }
};

class DateTimeType {
    DateType date;

```

```

        TimeType time;
public:
    DateTimeType(int y0=1, int m0=1, int d0=1, int hr0=0, int
mi0=0, int se0=0):date(y0,m0,d0),time(hr0,mi0,se0){}
    DateType& GetDate(){ return date; }
    TimeType& GetTime(){ return time; }
    void IncrementSecond(int s)
    {
        date.Carrying(time.Carrying(s));
    }
    void PrintDateTime()
    {
        date.PrintDate();
        cout<<' ';<br>
        time.PrintTime();
        cout<<endl;
    }
};

int main() {
    DateTimeType dttm1(1999,12,31,23,59,59), dttm2;
    (dttm1.GetDate()).PrintDate();
    cout<<endl;
    dttm1.PrintDateTime();
    dttm2.PrintDateTime();
    dttm1.IncrementSecond(30);
    dttm1.PrintDateTime();
    return 0;
}

```

2.3 运行截图

问题 输出 调试控制台 终端

> Executing task: D:\Codefield\CODE_Cpp\Cpp_Single\No.7\bin\2new.exe <

```

1999-12-31
1999-12-31 23:59:59
1-1-1 0:0:0
2000-1-1 0:0:29

```

按任意键关闭终端。

2.4 调试情况

Accepted，如上图。

3、第三题

3.1 题目描述

【题目描述】

设计一个 Point(点)类，数据信息包含 x 轴和 y 轴的坐标。设计一个 Circle(圆)类，数据信息包含圆心和半径。

要求：（1）数据部分都采用整型；

（2）圆心作为 Circle 类中的子对象；

（3）每个类都包含带有参数的构造函数；

（4）重载运算符“<<”和“>>”，用于输入输出每个类对象的数据信息；

（5）主函数内验证各个功能。

主函数参考代码：

```
int main()
{
    Point p(0,0);
    cin>>p;
    cout<<p;
    Circle c(0,0,0);
    cin>>c;
    cout<<c;
    return 0;
}
```

【输入】

输入有两行，第一行两个整数，分别代表 x 轴和 y 轴的坐标值；第二行三个整数，分别代表 x 轴坐标值、y 轴坐标值和半径值。

【输出】

输出三行，第一行是点的坐标，形式为 (x, y)；第二行是圆心坐标，形式仍为 (x, y)；第三行是半径值，最后有换行。注意：输出可以和输入交叉出现。

【样例输入】

1 2

3 4 5

【样例输出】

(1,2)

(3,4)

5

3.2 源代码

```
#include <iostream>
using namespace std;
class Point{
private:
    int x,y;
public:
    Point(int xx,int yy):x(xx),y(yy){}
    friend istream& operator>>(istream& in,Point& p);
    friend ostream& operator<<(ostream& out,Point& p);
};
istream& operator>>(istream& in,Point& p)
{
    in>>p.x>>p.y;
    return in;
}
ostream& operator<<(ostream& out,Point& p)
{
    out<<' ('<<p.x<<','<<p.y<<') '<<endl;
    return out;
}
class Circle{
private:
    Point O;
    int r;
public:
    Circle(int xx,int yy,int rr):O(xx,yy){r=rr;}
    friend istream& operator>>(istream& in,Circle& c);
    friend ostream& operator<<(ostream& out,Circle& c);
};
istream& operator>>(istream& in,Circle& c)
{

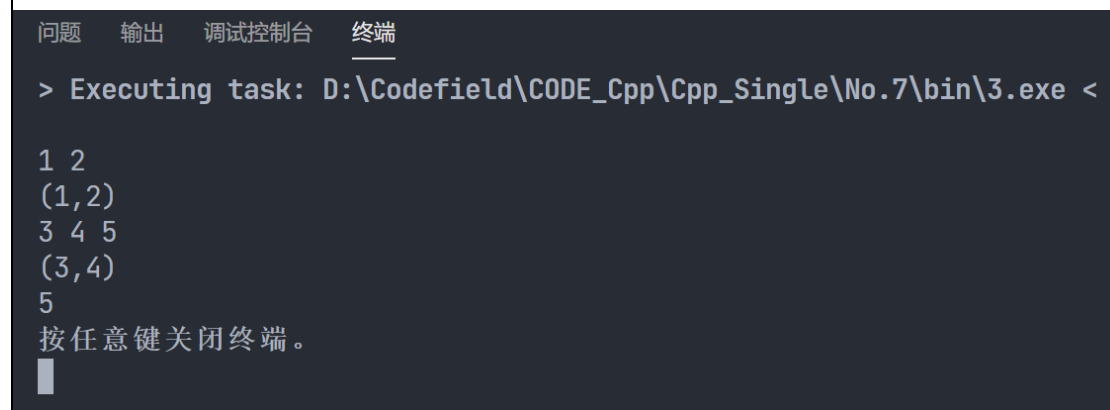
```

```

        in>>c.O;
        in>>c.r;
        return in;
    }
ostream& operator<<(ostream& out,Circle& c)
{
    out<<c.O<<c.r;
    return out;
}
int main()
{
    Point p(0,0);
    cin>>p;
    cout<<p;
    Circle c(0,0,0);
    cin>>c;
    cout<<c;
    return 0;
}

```

3.3 运行截图



```

问题  输出  调试控制台  终端
> Executing task: D:\Codefield\CODE_Cpp\Cpp_Single\No.7\bin\3.exe <

1 2
(1,2)
3 4 5
(3,4)
5
按任意键关闭终端。

```

3.4 调试情况

Accepted, 如上图。

三、实验体会

通过本次实验，巩固了类与对象的相关知识，学习了面向对象编程（OOP）的这种新的模式，体会到了面向对象编程方法的应用性，相信通过学习这种编程模式，可以使自己的编程能力有很大的提升。