山东大学 软件 学院

c++程序设计 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：201800301249 | 姓名：王帅 | | 班级：软件4班 |
| 实验题目：实验四 | | | |
| 实验学时：4 | | 实验日期： 2019.5.9 | |
| 实验目的：  熟悉c++的类、对象。  熟悉c++的重载、继承。 | | | |
| 硬件环境：  PC机 | | | |
| 软件环境：  DEV C++ | | | |
| 实验步骤与内容：  实验四（1）：  建立一个学生类，一个对象数组，内放5个学生的数据(学号、成绩)，用指针指向数组首元素，输出第1,3,5学生的数据。  建立一个函数bestStudent，用指向对象的指针作函数参数，在函数中找出5个学生中成绩最高者，并输出其学号。  在主函数main()中，调用bestStudent。  代码如下：  --begin--cpp--code--  #include <iostream>  using namespace std;  class student  {  public:  student(int n, double s): idnumber(n), score(s){}  int idnumber;  double score;  };  //用来找最高分的方法bestStudent  void bestStudent(student \*p)  {  student \*best=p;  for(int i=0;i<5;i++)  {  if((\*p).score<(\*(p+1)).score&&(\*best).score<(\*(p+1)).score) { best=(p+1);}  p=(p+1);  }  cout<<"最高成绩为："<<(\*best).score<<endl<<"学生学号为："<<(\*best).idnumber<<endl;  }  int main()  {  //对象数组student放学生数据  student a[5]={  student(11111,97),  student(22222,92),  student(33333,96),  student(44444,94),  student(55555,95)};    //输出第1，3，5学生的数据  student \*p;  for(int i=0;i<5;i=i+2)  {  p=&a[i];  cout<<"第"<<i+1<<"个学生的学号和成绩："<<endl;  cout<<(\*p).idnumber<<" "<<(\*p).score<<endl;  }  cout<<endl;  bestStudent(a);  return 0;  }  --end--cpp--code--  运行结果如下：    实验四（2）：  2. 定义一个复数类Complex，并实现：   1. 重载运算符“+”，使之能用于复数的加法运算。将运算符函数重载为非成员、非友元的普通函数。 2. 重载运算符“-”，使之能用于复数的减法运算。将运算符函数重载为类的友元函数。 3. 重载运算符“\*”,“/”，使之能用于复数的乘、除。运算符重载函数作为类的成员函数。 4. 重载运算符“+”，使之能用于复数的加法运算。参加运算的两个运算量有一个是类对象，有一个是整数，顺序任意。例如i+c1, c1+i均合法(设i为整数,c1,c2为复数)。 5. 实现实部自增、自减（都可以前置、后置）的运算 6. 重载流插入运算符“<<”和流提取运算符“>>”，使之能用于复数的输入和输出。 7. 处理一个复数与一个double数相加的运算(与复数的实部相加)，结果存放在一个double型的变量d1中，输出d1的值。在成员函数中包含重载类型转换运算符: operator double( ) { return real;} 8. 在主程序中分别调用以上复数运算，并输出运算结果。在调用的地方加上注释，说明要验证什么运算。   代码如下：  --begin--cpp--code--  #include <iostream>  using namespace std;  class Complex  {  public:  double real;  double imag;  Complex() {real=0; imag=0;}  Complex(double r,double i) {real=r; imag=i;}  void output() {cout<<real<<" + "<<imag<<"i"<<endl;}    //在成员函数中包含重载类型转换运算符  operator double(){return real;}    //声明友元函数  friend Complex operator-(Complex &a,Complex &b);  friend ostream& operator<<(ostream&,Complex&);  friend istream& operator>>(istream&,Complex&);    //重载运算符"\*","/"，运算符重载函数作为类的成员函数。  Complex operator\*(Complex &a)  {  Complex n;  n.real=real\*a.real-imag\*a.imag;  n.imag=real\*a.imag+imag\*a.real;  return n;  }  Complex operator/(Complex &a)  {  Complex n;  n.real=(real\*a.real+imag\*a.imag)/(a.real\*a.real+a.imag\*a.imag);  n.imag=(imag\*a.real-real\*a.imag)/(a.real\*a.real+a.imag\*a.imag);  return n;  }    };  //重载运算符"+",将运算符函数重载为非成员、非友元的普通函数  Complex operator+(Complex &a,Complex &b)  {  Complex n;  n.real=a.real+b.real;  n.imag=a.imag+b.imag;  return n;  }  //重载运算符"+"，参加运算的两个运算量有一个是类对象，有一个是整数，顺序任意  Complex operator+(Complex &a,int b)  {  Complex n;  n.real=a.real+b;  n.imag=a.imag;  return n;  }  Complex operator+(int b,Complex &a)  {  Complex n;  n.real=a.real+b;  n.imag=a.imag;  return n;  }  //处理一个复数与一个double数相加的运算(与复数的实部相加)  Complex operator+(Complex &a,double b)  {  Complex n;  n.real=a.real+b;  n.imag=a.imag;  return n;  }  //重载运算符"-"，将运算符函数重载为类的友元函数  Complex operator-(Complex &a,Complex &b)  {  Complex n;  n.real=a.real-b.real;  n.imag=a.imag-b.imag;  return n;  }  //实现实部自增、自减（前置、后置）的运算  Complex operator++(Complex &a,int b)  {  Complex n;  n.real=a.real++;  n.imag=a.imag;  return n;  }  Complex operator++(Complex &a)  {  Complex n;  n.real=a.real++;  n.imag=a.imag;  return n;  }  Complex operator--(Complex &a,int b)  {  Complex n;  n.real=a.real--;  n.imag=a.imag;  return n;  }  Complex operator--(Complex &a)  {  Complex n;  n.real=a.real--;  n.imag=a.imag;  return n;  }  //重载流插入运算符"<<"和流提取运算符">>"，使之能用于复数的输入和输出  ostream& operator<<(ostream& a,Complex& b)  {  a<<b.real<<" + "<<b.imag<<"i"<<endl;  return a;  }  istream& operator>>(istream& a,Complex& b)  {  cout<<"请依次输入实部和虚部："<<endl;  a>>b.real>>b.imag;  return a;  }  int main()  {  Complex c1(4,5),c2(1,2.5),c3;  int b=1;  double d1;  c3=c1+c2; c3.output(); //验证复数加法  c3=c1-c2; c3.output(); //验证复数减法  c3=c1\*c2; c3.output(); //验证复数乘法  c3=c1/c2; c3.output(); //验证复数除法  c3=c1+b; c3.output(); //验证复数加整数  c3=b+c1; c3.output(); //验证整数加复数  c1++; c1.output(); //验证复数自加（后置）  ++c1; c1.output(); //验证复数自加（前置）  c1--; c1.output(); //验证复数自减（后置）  --c1; c1.output(); //验证复数自减（前置）  cout<<c1; //验证复数输出  cin>>c3; //验证复数输入  cout<<"输入的复数为："<<c3;  c3=c1+1.5; //验证复数与double数相加  d1=double(c3);  cout<<"d1的值为："<<d1<<endl; //输出d1的值  }  --end--cpp--code--  输出结果如下：    实验四（3）：  编写一个程序，声明抽象基类Shape，由它派生出3个派生类： Circle(圆形)、Rectangle(矩形)、Triangle(三角形)，用一个函数printArea分别输出以上三者的面积，3个图形的数据在定义对象时给定。  代码如下：  --begin--cpp--code--  #include <iostream>  using namespace std;  //声明抽象基类Shape  class Shape  {  public:  virtual double area()=0;  };  //圆形派生类  class Circle:public Shape  {  public:  Circle(double r):radius(r){}  double area() {return radius\*radius\*3.14;}  private:  double radius;  };  //矩形派生类  class Rectangle:public Shape  {  public:  Rectangle(double l,double w):length(l),width(w){}  double area() {return length\*width;}  private:  double length;  double width;  };  //三角形派生类  class Triangle:public Shape  {  public:  Triangle(double b,double h):base(b),height(h){}  double area() {return 0.5\*base\*height;}  private:  double base;  double height;  };  //用来输出面积的printArea函数  void printArea(Shape &a) {cout<<a.area()<<endl;}  int main()  {  Circle c(5);  Rectangle r(1,2);  Triangle t(3,5);  cout<<"圆形的面积为：";  printArea(c);  cout<<"矩形的面积为：";  printArea(r);  cout<<"三角形的面积为：";  printArea(t);    return 0;  }  --end--cpp--code--  输出结果如下： | | | |
| 结论分析与体会：  本次实验让我更加熟悉了C++的类与对象，也了解了C++的重载和继承。总体而言，C++和java的差别还是比较大的，需要一定的时间来多加练习，才能熟练掌握。 | | | |