实验六 类与对象（四）

学号 姓名 班级

一、实验目的

掌握静态成员的定义与使用，掌握友元。

二、实验内容

1、定义员工类（Employee），求员工的工资总额及平均工资。要求用静态成员实现 。

2、求两个复数（Complex）之和。

（1）用成员函数实现求和。

（2）用普通函数实现求和。

三、实验步骤 （记录）

1、

#include <iostream>

#include <string>

#define Maxsize 999

using namespace std;

class Employee

{

private:

string name;

int salary;

public:

Employee()

{

name = "0";

salary= 0;

}

Employee(string name,int salary)

{

this->salary= salary;

this->name = name;

allper++;

Sum+=salary;

}

static void Getall()

{

cout<<"总人数为："<<allper<<endl;

cout<<"总工资为："<<Sum<<endl;

cout<<"所有员工的平均工资为："<<Getaver()<<endl;

}

static int Sum; //声明静态数据成员

static double Getaver()

{

return 1.0\*Sum/allper;

}

static int allper;

};

int Employee::Sum=0; //定义并初始化静态数据成员

int Employee::allper=0;

int main(void)

{

int q=1;

string name;

int sal;

cout<<"请输入员工的姓名和工资：(最多输入999人)"<<endl;

for(int i=0;i<Maxsize;i++)

{

cin>>name>>sal;

Employee employee(name,sal);

cout<<"按1继续输入，按0结束输入：";

cin>>q;

if(q==0)

break;

}

Employee::Getall();

system("pause");

return 0;

}

2、

#include <iostream>

using namespace std;

class Complex

{

public:

Complex()

{

real = 0;

imag = 0;

}

Complex(double r, double i)

{

real = r;

imag = i;

}

// friend Complex operator + (Complex &c1, Complex &c2); //成员函数实现求和 声明重载运算符

void add(Complex c2) //普通函数实现求和

{

this->real=this->real+c2.real;

this->imag=this->imag+c2.imag;

}

void Display()

{

if(this->real==0&&this->imag!=0)

cout<<this->imag<<"i"<<endl;

else if(this->real!=0&&this->imag==0)

cout<<this->real<<endl;

else if(this->real==0&&this->imag==0)

cout<<0<<endl;

else if(this->real>0&&this->imag<0)

cout<<this->real<<this->imag<<"i"<<endl;

else

cout<<this->real<<"+"<<this->imag<<"i"<<endl;

}

private:

double real;

double imag;

};

void NorAdd(int a,int b,int c,int d)

{

int x,y;

x=a+c;

y=b+d;

if(x==0&&y!=0)

{

cout<<y<<"i"<<endl;

}

else if(x!=0&&y==0)

{

cout<<x<<endl;

}

else if(x==0&&y==0)

{

cout<<0<<endl;

}

else if(x!=0&&y>0)

{

cout<<x<<"+"<<y<<"i"<<endl;

}

}

//Complex operator +(Complex &c1,Complex &c2)

//{ return Complex(c1.real + c2.real, c1.imag + c2.imag);//直接调用复制构造函数

//}

int main()

{

int a,b,c,d;

for(int i=0;i<4;i++)

{

cout<<"请输入第一个复数的实部、虚部：";

cin>>a>>b;

cout<<"请输入第二个复数的实部、虚部：";

cin>>c>>d;

Complex c1(a,b), c2(c,d), c3;

// c3 = c1 + c2; //运算符+ 用于复数运算

cout<<"成员函数实现求和："<<endl;

c1.add(c2);

c1.Display();

// c3.Display();

cout<<"普通函数实现求和："<<endl;

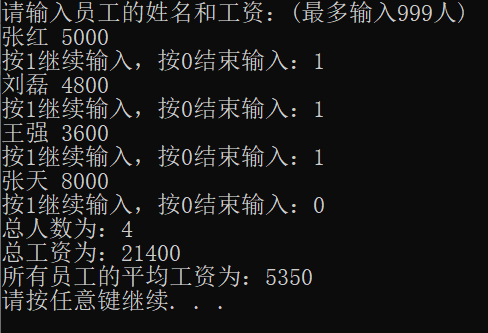
NorAdd(a,b,c,d);

}

return 0;

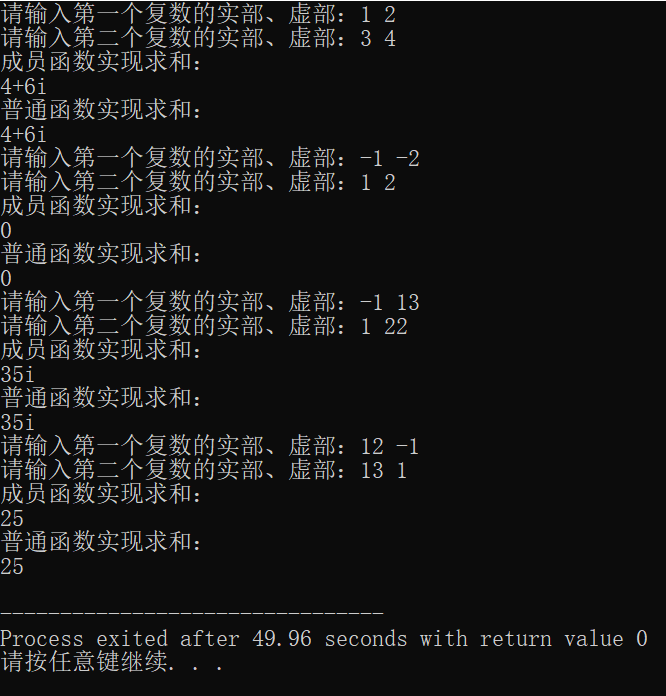
}四、实验结果及分析

1、



分析：员工的姓名和工资均为私有成员，所要求的人数、平均工资、总工资需要每一个对象的成员、故选用静态成员变量。

2、



分析：

五、实验总结

友元关系不能被传递或继承的，且是单向的。它可以破坏数据隐蔽和封装的机制，这是他的用处，也是他的坏处。