

《面向对象程序设计》

实验报告

|  |  |
| --- | --- |
| 班 级： | **计222** |
| 学 号： | **22013166** |
| 姓 名： | **杨建鑫** |
| 指导教师： | **梁建宁** |

信息科学与工程学院

2023年11月

**实验五、多态性**

**【实验目的】**

1、理解运算符重载

2、掌握利用友元函数与成员函数实现运算符重载。

**【实验内容】**

1、用操作符重载（采用友元函数）实现复数加、减运算。

Code：

#include <iostream>

using namespace std;

class Complex

{

public:

    Complex(double *real* = 0.0, double *imag* = 0.0) : real(*real*), imag(*imag*) {}

*// 加法运算符重载*

    friend Complex operator+(const Complex &*c1*, const Complex &*c2*)

    {

        Complex result;

        result.real = *c1*.real + *c2*.real;

        result.imag = *c1*.imag + *c2*.imag;

        return result;

    }

*// 减法运算符重载*

    friend Complex operator-(const Complex &*c1*, const Complex &*c2*)

    {

        Complex result;

        result.real = *c1*.real - *c2*.real;

        result.imag = *c1*.imag - *c2*.imag;

        return result;

    }

    void show() const{

        if(imag>0){

            cout<<real<<"+"<<imag<<"i"<<endl;

        }

        else{

            cout<<real<<imag<<"i"<<endl;

        }

    }

private:

    double real;

    double imag;

};

int main()

{

    Complex c1(1.0, 2.0);

    Complex c2(3.0, 4.0);

    Complex addition = c1 + c2;

    Complex subtraction = c1 - c2;

    cout << "addition:";

    addition.show();

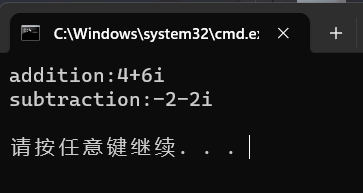
    cout<<"subtraction:";

    subtraction.show();

    return 0;

}

Example：



2、用操作符重载（采用成员函数）实现复数乘、除运算。

Code：

#include <iostream>

using namespace std;

class Complex

{

public:

    Complex(double *real* = 0.0, double *imag* = 0.0) : real(*real*), imag(*imag*) {}

*// 乘法运算符重载*

    Complex operator\*(const Complex &*other*) const

    {

        Complex result;

        result.real = this->real \* *other*.real - this->imag \* *other*.imag;

        result.imag = this->real \* *other*.imag + this->imag \* *other*.real;

        return result;

    }

*// 除法运算符重载*

    Complex operator/(const Complex &*other*) const

    {

        double divisor = *other*.real \* *other*.real + *other*.imag \* *other*.imag;

        Complex result;

        result.real = (this->real \* *other*.real + this->imag \* *other*.imag) / divisor;

        result.imag = (this->imag \* *other*.real - this->real \* *other*.imag) / divisor;

        return result;

    }

    void show() const

    {

        if (imag > 0) { cout << real << "+" << imag << "i" << endl; }

        else { cout << real << imag << "i" << endl; }

    }

private:

    double real;

    double imag;

};

int main()

{

    Complex c1(1.0, 2.0);

    Complex c2(3.0, 4.0);

    Complex multiply = c1 \* c2;

    Complex divide = c1 / c2;

    cout << "multiply:";

    multiply.show();

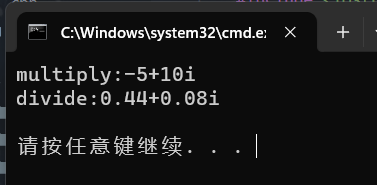
    cout << "divide:";

    divide.show();

    return 0;

}

Example：



3、编程实现一个student类，管理一个学生的基本信息，包括学生姓名、学号、语文成绩、数学成绩、英语成绩等数据成员，然后在main函数中，从键盘输入得到一个学生的基本信息，最后输出所有数据成员值到屏幕上；在以上student类的基础上分别用成员函数和友元函数两种形式重载“= =”运算符，判断二个student对象的成绩是否相等(判定依据是三门平均分)；在以上student类的基础上分别用成员函数和友元函数两种形式重载“>”运算符，判断二个student对象的成绩关系。并在main函数中完成两个对象的“= =”和“>”运算功能。

Code：

#include <iostream>

#include <string>

using namespace std;

class Student

{

public:

    Student(const string &*name*, int *id*, double *chinese*, double *math*, double *english*)

        : name(*name*), id(*id*), chinese(*chinese*), math(*math*), english(*english*) {}

*// 输出学生信息*

    void display() const

    {

        cout << "Name: " << name << endl;

        cout << "ID: " << id << endl;

        cout << "Chinese: " << chinese << endl;

        cout << "Math: " << math << endl;

        cout << "English: " << english << endl;

    }

*// 重载==运算符（成员函数）*

    bool operator==(const Student &*other*) const

    {

        double avg1 = (chinese + math + english) / 3.0;

        double avg2 = (*other*.chinese + *other*.math + *other*.english) / 3.0;

        return avg1 == avg2;

    }

*// 重载>运算符（成员函数）*

    bool operator>(const Student &*other*) const

    {

        double avg1 = (chinese + math + english) / 3.0;

        double avg2 = (*other*.chinese + *other*.math + *other*.english) / 3.0;

        return avg1 > avg2;

    }

*/\**

*由于 == 和 > 运算符被重载了两次，一次是作为成员函数，一次是作为友元函数。这导致了编译器不确定在比较两个Student对象时应该调用哪个函数，产生了警告。因此在这里对友元函数重载进行了注释。*

*// 友元函数重载==运算符*

*friend bool operator==(const Student &s1, const Student &s2)*

*{*

*double avg1 = (s1.chinese + s1.math + s1.english) / 3.0;*

*double avg2 = (s2.chinese + s2.math + s2.english) / 3.0;*

*return avg1 == avg2;*

*}*

*// 友元函数重载>运算符*

*friend bool operator>(const Student &s1, const Student &s2)*

*{*

*double avg1 = (s1.chinese + s1.math + s1.english) / 3.0;*

*double avg2 = (s2.chinese + s2.math + s2.english) / 3.0;*

*return avg1 > avg2;*

*}*

*\*/*

private:

    string name;

    int id;

    double chinese;

    double math;

    double english;

};

int main()

{

    string name;

    int id;

    double chinese;

    double math;

    double english;

*// 输入学生信息*

    cout << "Enter student information: " << endl;

    cout << "Name: ";

    getline(cin, name);

    cout << "ID: ";

    cin >> id;

    cout << "Chinese Score: ";

    cin >> chinese;

    cout << "Math Score: ";

    cin >> math;

    cout << "English Score: ";

    cin >> english;

*// 创建学生对象*

    Student student1(name, id, chinese, math, english);

*// 输出学生信息*

    cout << "Student Information:" << endl;

    student1.display();

    cout << endl;

*// 创建第二个学生对象*

    Student student2("John", 2, 85, 90, 95);

*// 测试"=="运算符重载*

    if (student1 == student2)

    {

        cout << "Average scores are equal." << endl;

    }

    else

    {

        cout << "Average scores are not equal." << endl;

    }

*// 测试">"运算符重载*

    if (student1 > student2)

    {

        cout << "Student1 has higher average scores." << endl;

    }

    else

    {

        cout << "Student2 has higher average scores." << endl;

    }

    return 0;

}

Example：

文本

描述已自动生成