



|  |  |  |  |
| --- | --- | --- | --- |
| **学号** | 17130130312 | **姓名** | 曹晨瑶 |
| **班级** | **1713013** | **任课教师** | **王献青** |
| **实验名称** | 类与操作符重载 | | |
| **实验学期** | **2018 – 2019 学年第2学期** | | |
| **实验日期** | 2019年6月1日 | **实验地点** | **G334** |
| **报告成绩** |  |  |  |

西安电子科技大学软件学院

# 实验目的

类使得自定义类型表现和基本类型相似，操作符重载机制则可使得自定义类型的一些操作的使用与习惯一致。本次实验通过使用类、操作符重载机制来设计并实现一些程序，了解并熟悉使得自定义类型更便于使用的手段，加深学生对基于类的操作符重载机制的理解，进一步提高面向对象编程能力。

# 实验环境

操作系统：Windows10

开发工具: Code::Blocks

# 实验内容

## 定义Complex及其操作

Complete and test class Complex of §11.3 ;

## 定义Vec4及操作

Define a type ***Vec4***as a vector of four *float*s. Define ***operator*[]** for *Vec4*. Define operators +, -, \*, /, =, +=, -=, \*=, and /= for combinations of vectors and floating point numbers.

## 定义指针类

Complete class Ptr\_to\_T from §11.11 and test it. To be complete, Ptr\_to\_T must have at least the operators \*, ->, =, ++, and -– define. Do not cause a run-time error until a wild pointer is actually dereferenced.

# 数据结构与算法设计

## 定义了complex及其操作

**模块结构及文件组织设计：**

模块1：仅包括main.cpp文件,定义了main（）函数，对复数的相关操作进行了测试。

模块2：仅包含complex.hpp文件，定义了复数类complex及其重载操作符

模块3：仅包含complex.cpp文件，实现复数类complex及其操作，包括操作符重载

**关键数据结构设计：**

**class complex**

**{**

**double re,im;**

**public:**

**complex(double r=0,double i=0) : re(r),im(i) {}**

**double real() const {return re;}**

**double imag() const {return im;}**

**void setter\_real(double a){ re=a;}**

**void setter\_imag(double a){ im=a;}**

**complex& operator+=(complex a);**

**complex& operator+=(double a);**

**complex& operator-=(complex a);**

**complex& operator-=(double a);**

**complex& operator\*=(complex a);**

**complex& operator\*=(double a);**

**complex& operator/=(complex a);**

**complex& operator/=(double a);**

**friend istream& operator>>(istream& is,complex& a);//输入**

**friend ostream& operator<<(ostream& os,complex& a);//输出**

**};**

**算法1.1 int main()**

作 用：主控函数,，测试函数，测试相关操作

参 数：无

返回值：总是返回0。

计算过程：

1. 定义类complex a,并输入a，b
2. 调用算法1.4计算和差
3. 调用算法1.5计算积商
4. 返回0

**算法1.2 complex& complex::operator+=() 、complex& complex::operator-=()**

作 用：根据参数的不同重载运算符+=、-=

参 数：conplex a/double a

返回值：返回complex自身的引用

计算过程：

1. 若参数为complex a,则re加（减）a.re并赋值给re，im加（减）a.im并赋值给im，返回cpmplex自身引用；
2. 若参数为双精度浮点型a,则实部re加（减）a并赋值给re，im不变，返回complex自身引用。

**算法1.3 complex& complex::operator\*=()、complex& complex::operator/=()**

作 用：根据参数的不同重载操作符\*=、/=

参 数：conplex a/double a

返回值：返回complex自身的引用

计算过程：

1. \*=：若参数为complex a，定义一个变量r并赋值re，re等于re\*a.re-im\*a.im,im等于r\*a.im+a.re\*im，返回complex自身的引用
2. /=：若参数为complex a，定义一个变量r并赋值re，re等于(re\*a.re+im\*a.im)/(a.re\*a.re+a.im\*a.im),im等于(im\*a.re-r\*a.im)/(a.re\*a.re+a.im\*a.im)，返回complex自身的引用
3. 若参数为双精度浮点型a，则re为re乘（除）a，im不变，返回complex自身的引用

**算法1.4 complex operator+()、complex operator-()**

作 用：根据参数的不同重载操作符+、-（分为一元操作符和二元操作符）

参 数：complex a,complex b/complex a,double b/double a ,complex a/complex a(一元操作符)

返回值：返回complex

计算过程：

1. complex a,complex b：定义一个新的complex r，并将a赋值给r，返回r+=b(r-=b)

(调用算法1.2 complex：：（complex）)

1. complex a,double b：定义一个新的complex r，并将a赋值给r，返回r+=b(r-=b)

(调用算法1.2 complex：：（double）)

1. double a ,complex b：定义一个新的complex r，并将b赋值给r，返回r+=a(r-=a)

(调用算法1.2 complex：：（double）)

1. complex a：将complex的实部和虚部同时乘-1或乘一，并返回a

**算法1.5 complex operator\*()、complex operator/()**

作 用：根据参数的不同重载操作符\*、/

参 数：complex a,complex b/complex a,double b/double a,complex a

返回值：返回complex

计算过程：

1. complex a,complex b：定义一个新的complex r，并将a赋值给r，返回r\*=b(r/=b)

(调用算法1.3complex：：（complex）)

1. complex a,double b：定义一个新的complex r，并将a赋值给r，返回r\*=b(r/=b)

(调用算法1.3 complex：：（double）)

1. double a ,complex b：定义一个新的complex r，并将b赋值给r，返回r\*=a(r/=a)

(调用算法1.3complex：：（double）)

**算法1.6 bool operator==(complex a,complex b)、bool operator！=(complex a,complex b)**

作 用：重载操作符==，！=

参 数：complex a,complex b

返回值：true 或者 false

计算过程：

1. ==：若复数a和复数b的实部和虚部分别相同，则返回true，否则返回false
2. ！=若复数a和复数b的实部不相同或者虚部不相同，则返回true，否则返回false

**算法1.7 std::istream& operator>>(istream& is,complex& a)**

作 用：重载操作符>>

参 数：istream& is,complex& a

返回值：istream的引用

计算过程：输入复数实部和虚部，并调用setter\_real()和setter\_imag()置数

**算法1.8 std::ostream& operator<<(ostream& os,complex &a)**

作 用：重载操作符<<

参 数：ostream& os,complex &a

返回值：ostream的引用

计算过程：调用real（）和image（）将实部和虚部输出

## 定义了Vec4及其操作

**模块结构及文件组织设计：**

模块1：仅包括main.cpp文件,定义了main（）函数。

模块2：仅包含Vec4.hpp文件，定义了类Vec4及其接口

模块3：仅包含Vec4.cpp文件，实现了类Vec4及其操作

**关键数据结构设计：**

class Vec4

{

float vec[4];

public:

Vec4(float a=0,float b=0,float c=0,float d=0){}//default constructor

Vec4(const Vec4& a);//copy constructor

Vec4 operator=(const Vec4 a);//复制赋值

float& operator[](int n);

//一元操作

Vec4 operator-(Vec4 a);//一元操作

Vec4 operator+(Vec4 a);

Vec4& operator+=(Vec4 a);

Vec4& operator+=(float a);

Vec4& operator-=(Vec4 a);

Vec4& operator-=(float a);

Vec4& operator\*=(Vec4 a);

Vec4& operator\*=(float a);

Vec4& operator/=(Vec4 a);

Vec4& operator/=(float a);

friend Vec4 operator+(Vec4 a,Vec4 b);

friend Vec4 operator+(Vec4 a,float b);

friend Vec4 operator-(Vec4 a,Vec4 b);

friend Vec4 operator-(Vec4,float);

friend Vec4 operator\*(Vec4 a,Vec4 b);

friend Vec4 operator\*(Vec4 a,float b);

friend Vec4 operator/(Vec4 a,Vec4 b);

friend Vec4 operator/(Vec4 a,float b);

friend std::istream& operator>>(istream& in, Vec4& a ); // 输入操作

friend std::ostream& operator<<(ostream& out,const Vec4& a); // 输出操作

};

**算法2.1 int main()**

作 用: 主控函数 测试函数

参 数： 无参数

返回值： 0

计算过程：

1. 定义Vec4 a,b
2. 调用算法2.8输入a
3. 调用算法2.8输入a
4. 进行a，b间的运算，调用算法2.6 2.7
5. 调用算法2.9输出

**算法2.2 Vec4::Vec4()**

作 用：拷贝构造函数

参 数：**const Vec4& a**

返回值：无

计算过程：将a中数组的四个浮点数依次赋值给Vec4中的数组中的四个浮点数

**算法2.3 Vec4 Vec4::operator=（）**

作 用：重载操作符=（复制赋值）

参 数：const Vec4 a

返回值: Vec4自身引用

计算过程：将a中数组的四个浮点数依次赋值给Vec4中的数组中的四个浮点数

**算法2.4 float& Vec4::operator[]（）**

作 用：重载操作符【】

参 数：int n

返回值：数组vec[]中某个浮点数的引用

计算过程：

1. 若n<0,返回vec[0]的引用
2. 若n>3,返回vec[3]的引用
3. 否则，返回vec[n]的引用

**算法2.5 Vec4 Vec4::operator-() 、Vec4 Vec4::operator+()**

作 用：重载操作符-、+（一元操作符）

参 数：Vec4 a

返回值：Vec4自身引用

计算过程：将a中数组中的四个浮点数分别变为相反数（乘-1）或者不变，依次赋值给vec中的四个浮点数，返回Vec4自身引用

**算法2.6 Vec4& Vec4::operator+=（）、Vec4& Vec4::operator-=（）、**

**Vec4& Vec4::operator\*=（）、Vec4& Vec4::operator/=（）**

作 用：根据参数的不同，重载操作符+=、-=、\*=、/=

参 数：Vec4 a/float a

返回值：Vec4自身引用

计算过程：

1. 若参数为Vec4 a,则vec[0..3]=vec[0..3]+a.[0..3](对应相加、相减、相乘、相除)
2. 若参数为float a，则数组中四个浮点数都加（减、乘、除）a，并依次赋值给之前

原来的变量

**算法2.7 Vec4 operator+（）、Vec4 operator-（）、Vec4 operator\*()、Vec4 operator/（）**

作 用：重载操作符+、-、\*、/

参 数：Vec4 a,Vec4 b / Vec4 a,float b

返回值：Vec4

计算过程：

定义一个新的Vce4 r

1. 若参数为Vec4 a，Vec4 b,则r.vec[0..3]=a.vec[0..3]+b.[0..3](对应相加、相减、相乘、相除)
2. 若参数为Vec4 a,float b，则数组中四个浮点数都加（减、乘、除）b，并依次赋值r.vec[0..3]

**算法2.8 std::istream& operator>>（）**

作 用：重载操作符>>

参 数：istream& in, Vec4& a

返回值：istream&

计算过程：输入Vec4 a中数组的四个浮点数

**算法2.9** **std::ostream& operator << ()**

作 用：重载操作符<<

参 数：ostream& out,const Vec4& a

返回值：ostream&

计算过程：输出Vec4 a中数组的四个浮点数

## 定义指针类

**模块结构及文件组织设计：**

模块1：主控模块，仅包括文件main.cpp ，定义了 main()函数和结构体T,用于对指针类进行测试。

模块2： 仅包含ptr.hpp，定义了指针类和其接口

模块3： 仅包含ptr.cpp，实现了指针类

**关键数据结构设计：**

struct T

{

int m;

};

class Ptr\_to\_T

{

private:

T\* p;

T\* array;

int size;

public:

Ptr\_to\_T(T\* t,T\* v,int s);//约束到大小s的数组v，初始值为P

Ptr\_to\_T(const Ptr\_to\_T & a);//约束到单个对象，初始值为p

Ptr\_to\_T& operator=( const Ptr\_to\_T& a);

Ptr\_to\_T& operator++();//前缀

Ptr\_to\_T operator++(int);//后缀

Ptr\_to\_T& operator--();

Ptr\_to\_T operator--(int);

T& operator\*();

T\* operator->();

};

**算法3.1 int main()**

作 用：主控函数，也实现对题目所需其他内容的测试。

参 数：无

返回值：总是返回0。

计算过程：

1. 定义T v[200],h[200]
2. 给数组v[200]循环赋值
3. 调用算法3.2，定义 Ptr\_to\_T TV并初始化
4. 调用算法3.2，定义 Ptr\_to\_T TH并初始化
5. 进行数组访问越界控制，若发现越界，抛出错误，捕获错误，停止程序
6. TV调用算法3.6
7. 调用算法3.8，输出第二个数
8. 调用算法3.4和算法3.6 TH=TV+1
9. 调用算法3.7 输出TH中的第二个数
10. 调用算法3.8. 输出TV中的第三个数

**算法3.2 Ptr\_to\_T::Ptr\_to\_T()**

作 用：初始化 约束到大小为s的数组v，初始值为p

参 数：T\* t,T\* v,int s

返回值： 无

计算过程：分别将t、v、s赋值给类Ptr\_to\_T 中的p、array、size

**算法3.3 Ptr\_to\_T::Ptr\_to\_T()**

作 用：初始化 约束到单个对象，初始值为p

参 数：const Ptr\_to\_T &a

返回值：无

计算过程：分别将a中的t、v、s赋值给类Ptr\_to\_T 中的p、array、size

**算法3.4 Ptr\_to\_T& Ptr\_to\_T::operator=()**

作 用：重载操作符=（复制赋值）

参 数：const Ptr\_to\_T& a

返回值：返回Ptr\_to\_T&

计算过程：分别将a中的t、v、s赋值给类Ptr\_to\_T 中的p、array、size，并返回Ptr\_to\_T的自身引用

**算法3.5 Ptr\_to\_T& Ptr\_to\_T::operator++()、Ptr\_to\_T& Ptr\_to\_T::operator--()**

作 用：重载前缀操作符++、--

参 数：无

返回值： 返回Ptr\_to\_T&

计算过程：p=p+1（p=p-1）,其余不变，返回自身引用

**算法3.6 Ptr\_to\_T Ptr\_to\_T::operator++(int)、Ptr\_to\_T Ptr\_to\_T::operator--(int)**

作 用：重载后缀操作符++、--

参 数：int

返回值：类Ptr\_to\_T

计算过程：

1. 定义一个新的类Ptr\_to\_T r
2. 调用算法1.4，将原来的类赋值给它
3. p=p+1或者p=p-1
4. 返回r

**算法3.7** **T& Ptr\_to\_T::operator\*()**

作 用：重载操作符\*

参 数：无

返回值：T&

计算过程：直接返回\*p

**算法3.8 T\* Ptr\_to\_T::operator->()**

作 用：重载操作符->

参 数：无

返回值：T\*

计算过程：直接返回p

# 测试用例与测试结果

## 5.1定义complex及其操作

|  |  |
| --- | --- |
| 序号 | 输出 |
| 1 | \*\*\*\*\*复数运算\*\*\*\*\*  输入实部和虚部：（以空格分开）  1 2  输入实部和虚部：（以空格分开）  3 5  和为：(4,7)  差为：(-2,-3)  积为：(-7,11)  商为：(0.382353,0.0294118) |
| 2 | \*\*\*\*\*复数运算\*\*\*\*\*  输入实部和虚部：（以空格分开）  5 -3  输入实部和虚部：（以空格分开）  -6 9  和为：(-1,6)  差为：(11,-12)  积为：(-3,63)  商为：(-0.487179,-0.230769) |

## 5.3定义Vec4及其操作

|  |  |  |
| --- | --- | --- |
| 序号 | 测试数据 | 输出 |
| 1 | Vec4 a, b;  cin >> a;  cin >> b;  cout<<"结果为："<<endl;  a += b;  cout << a << endl;  a = b + 1;  cout << a << endl;  a -= 2.0;  cout << a << endl;  a = b \* 2;  cout << a << endl;  a = a \* b;  cout << a << endl;  a /= b;  cout << a << endl; | 四个数之间用空格分开：  1 2 3 4  四个数之间用空格分开：  5 9 7 3  结果为：  (6,11,10,7)  (6,10,8,4)  (4,8,6,2)  (10,18,14,6)  (50,162,98,18)  (10,18,14,6) |
|  |  | 四个数之间用空格分开：  -8 -6 1 3  四个数之间用空格分开：  5 1 9 45  结果为：  (-3,-5,10,48)  (6,2,10,46)  (4,0,8,44)  (10,2,18,90)  (50,2,162,4050)  (10,2,18,90) |

## 5.3定义算术表达式类型

|  |  |
| --- | --- |
| 序号 | 输出结果 |
| 1 | 1  1  2 |

# 实验总结

通过这次实验，我掌握了类的使用和操作符重载。在实验中，对于大型的结果，我们要考虑优化返回方式；对于大型运算对象，我们要使用 const 引用参数类型；对于对称的运算符，我们要采用非成员函数的方法等等。我们要通过大量的实践，将课本上的理论知识转化为自己的，熟能生巧。类对于我们来说是一个新知识，我们需要加深对其的理解和使用，了解并熟悉其中的机制和规范，进一步提高面向对象编程的能力。