实验八 运算符重载

学号 20181101047 姓名 范少帅 班级 18计科3班

一、实验目的

理解运算符重载的概念，掌握常用运算符重载。

二、实验内容

1、定义复数类，假定c1，c2代表复数对象，实现如下功能：

（1）两个复数相加 c1+c2

（2）复数和实数相加 c1+5.0 5.0+c1

（3）复数对象自增（规则：实部、虚部均自增） c1++ ++c1

（4）重载“<<”、”>>” 运算符实现复数对象输入输出。

2、定义一个数组类Array（封装数组及对数组的操作，数组元素空间动态分配），实现如下功能：

（1）数组复制

（2）数组赋值

（3）重载”[]”访问数组元素

（4）重载“<<”、”>>” 运算符实现数组对象输入输出。

三、实验步骤 （记录）

1、

#include <iostream>

using namespace std;

class complex

{

public:

double real,image;

public:

complex(double a=0,double b=0)

{

real=a;image=b;

}

friend complex add(complex &x,complex &y);

friend complex operator+(complex &c1,double &b);

complex operator++();

complex operator++(int);

friend istream & operator>>(istream & in, complex & A);

friend ostream & operator<<(ostream & out, complex & A);

void Show()

{

cout<<"("<<real<<","<<image<<")"<<endl;

}

};

//重载输入运算符

istream & operator>>(istream & in, complex & A){

in >> A.real >> A.image;

return in;

}

//重载输出运算符

ostream & operator<<(ostream & out, complex & A){

out << A.real << " + " << A.image << " i ";;

return out;

}

complex complex::operator++()

{

return complex(++real,image);

}

complex complex::operator++(int)

{

return complex(real++,image);

}

complex operator+(complex c1,double b)

{

return complex(c1.real+b,c1.image);

}

complex add(complex &x,complex &y)

{

complex z;

z.real=y.real+x.real;

z.image=y.image+x.image;

return z;

}

int main() //主函数

{

complex c1(1.5,2.8),c2(-2.3,3.4),c3; //声明复数类的对象

c3=add(c1,c2);//友元函数调用

cout<<"c3=";

c3.Show();

c3=c1+5.0;

cout<<"c1+5.0=";

c3.Show();

c1++;

cout<<endl;

cout<<"c1++=";

c1.Show();

++c1;

c1.Show();

cout<<c1;

return 0;

}

2、

#include <iostream>

using namespace std;

class Array{

public:

int \*data;

int size;

static const int dSize = 10; // 数组默认大小

public:

Array()

{ // 无参构造

size = dSize;

data = new int[size]();

}

Array(int n ){ // 有参构造

size = n;

data = new int[size]( );

}

Array( Array & arr)//拷贝构造函数,深拷贝

{

if(arr.size>0)

{

size = arr.size;

data = new int[size]( );

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

{

data[i] = arr.data[i];

}

}

}

~Array()

{

if (this->data != NULL)//不为空才释放

{

delete []data;

}

}

int& operator [] (int k)

{

return data[k-1];

}

friend ostream& operator << (ostream& o, const Array& a); // 运算符 << 重载，以方便输出

friend istream& operator>>(istream & in, Array & A);

};

ostream& operator << (ostream& o, const Array& a){

o << '[' ;

for(int i=0; i<a.size-1; i++)

o << a.data[i] << ',' ;

o << a.data[a.size-1] << ']';

return o;

}

istream& operator >>(istream &in,Array &A)

{

for(int i=0;i<A.size;i++)

in >> A.data[i];

return in;

}

int main(){

int n, k;

cout<<"输入数组大小："<<endl;

cin >>n;

Array a(n); // 构造数组，大小为 n

cout<<"请输入数组："<<endl;

cin>>a;

Array b(a); // 拷贝构造数组

cout <<"a=" <<a << endl;

cout <<"数组复制后："<<endl;

cout<<"b = "<< b << endl;

cout<<"请输入您要访问第几个数组："<<endl;

cin>>k;

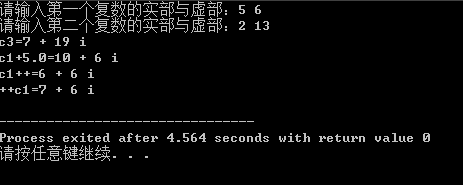
cout <<"a[k]="<< a[k] << endl;

return 0;

}

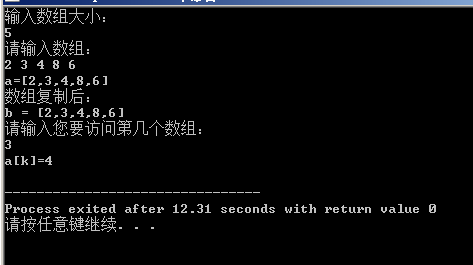
四、实验结果及分析

1、



分析：c3 = c1+c2;>>符号经过重载直接cout>>c3 即可输出7+19i；

2、



分析：<< 和>>符号经过重载后直接输出数组cout<<a时即可得到[2，3,4,5,6]格式的数组

五、实验总结

运算符重载就是对已有的运算符重新进行定义，赋予其另一种功能，以达到适应不同的数据类型。运算符重载不能改变它本来的寓意(也就是 加法不能变更为减法)运算符重载只是一种 “语法上的方便” ， 它只是一种函数调用的方式。