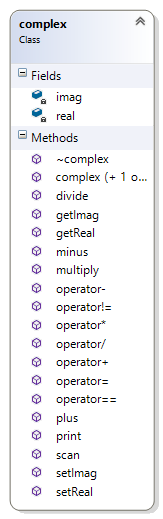
# Lab 7

Problem: Improve complex number class

UML Diagram:



## Source Files:

main.cpp:

/\*main.cpp

Here to text complex class

\*/

#include "complex.h"

int main()

{

complex c;

complex d;

cout<<"Enter a complex number in the form a+bi: ";

cin>>c;

cout<<"Enter another complex number in the form a+bi: ";

cin>>d;

cout<<endl<<"The first complex multiplied by the second complex is: ";

complex e=c\*d;

cout<<e;

cout<<endl<<"The first complex divided by the second complex is ";

complex f=c/d;

cout<<f;

cout<<endl<<"The first complex plus the second complex is ";

complex g=c+d;

cout<<g;

cout<<endl<<"The first complex minus the second complex is ";

complex h=c-d;

cout<<h;

cout<<endl<<endl;

return 0;

}

complex.h:

#pragma once

#include <iostream>

#include <fstream>

using namespace std;

class complex

{

public:

complex(void); //default constructor

complex(double rp, double ip); //initializing constructor

~complex(void); //destructor

private:

double real;

double imag;

public:

void print(ostream & out);

void scan(istream & in);

double getReal(void);

void setReal(double rv);

double getImag(void);

void setImag(double iv);

complex multiply(const complex & second)const;

complex divide(const complex & second)const;

complex plus(const complex & second)const;

complex minus(const complex & second)const;

complex operator=(const complex & second);

complex operator\*(const complex & second)const;

complex operator/(const complex & second)const;

complex operator+(const complex & second)const;

complex operator-(const complex & second)const;

bool operator==(const complex & second)const;

bool operator!=(const complex & second)const;

friend ostream & operator<<(ostream & out,const complex & c);

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

};

complex.cpp:

#include "complex.h"

complex::complex(void)

: real(0)

, imag(0)

{

}

complex::~complex(void)

{

cout<<"Destructor called for ";

print(cout);

cout<<endl;

}

complex::complex(double rp, double ip)

{

real=rp;

imag=ip;

}

void complex::print(ostream & out)

{

if(imag>=0)

out<<real<<"+"<<imag<<"i";

else

out<<real<<imag<<"i";

}

void complex::scan(istream & in)

{

char junk;

in>>real>>imag>>junk;

}

double complex::getReal(void)

{

return real;

}

void complex::setReal(double rv)

{

real=rv;

}

complex complex::multiply(const complex & second)const

{

return complex(real\*second.real-imag\*second.imag ,real\*second.imag+imag\*second.real);

}

complex complex::divide(const complex & second)const

{

return complex((real\*second.real+imag\*second.imag)/(second.real\*second.real+second.imag\*second.imag),(imag\*second.real-real\*second.imag)/(second.real\*second.real+second.imag\*second.imag));

}

complex complex::plus(const complex & second)const

{

return complex(real+second.real,imag+second.imag);

}

complex complex::minus(const complex & second)const

{

return complex(real-second.real,imag-second.imag);

}

double complex::getImag(void)

{

return imag;

}

void complex::setImag(double iv)

{

imag=iv;

}

complex complex::operator=(const complex & second)

{

real=second.real;

imag=second.imag;

return complex(\*this);

}

complex complex::operator\*(const complex & second)const

{

return complex(real\*second.real-imag\*second.imag ,real\*second.imag+imag\*second.real);

}

complex complex::operator/(const complex & second)const

{

return complex((real\*second.real+imag\*second.imag)/(second.real\*second.real+second.imag\*second.imag),(imag\*second.real-real\*second.imag)/(second.real\*second.real+second.imag\*second.imag));

}

complex complex::operator+(const complex & second)const

{

return complex(real+second.real,imag+second.imag);

}

complex complex::operator-(const complex & second)const

{

return complex(real-second.real,imag-second.imag);

}

bool complex::operator==(const complex & second)const

{

return fabs(real-second.real)<.0000001 && fabs(imag-second.imag)<.0000001;

}

bool complex::operator!=(const complex & second)const

{

return fabs(real-second.real)>.0000001 && fabs(imag-second.imag)>.0000001;

}

ostream & operator<<(ostream & out,const complex & c)

{

if(c.imag>=0)

out<<c.real<<"+"<<c.imag<<"i";

else

out<<c.real<<c.imag<<"i";

return out;

}

istream & operator>>(istream & in, complex & c)

{

double rv,iv;

char junk;

in>>rv;

c.real=rv;

in>>iv;

c.imag=iv;

in>>junk;

return in;

}

Sample Run:

Enter a complex number in the form a+bi: 6-7i

Enter another complex number in the form a+bi: 4+2i

The first complex multiplied by the second complex is: 38-16i

The first complex divided by the second complex is 0.5-2i

The first complex plus the second complex is 10-5i

The first complex minus the second complex is 2-9i

Destructor called for 2-9i

Destructor called for 10-5i

Destructor called for 0.5-2i

Destructor called for 38-16i

Destructor called for 4+2i

Destructor called for 6-7i

Press any key to continue . . .