ECE 2305

Introduction to C Programming

Programming Project 10

Object-oriented Programming

Program Features: Object-oriented Programming, Class Declarations, Constructor Functions, Accessor Functions.

Write a C++ program that creates a class of objects called complex. Use these objects to represent complex numbers. The two Data Members of the class shall be the real part and the imaginary part of the complex number.

The Member Functions of the class shall include:

A default constructor

A constructor that accepts the real part and the imaginary part as parameters

A function that sets the value of the real part

A function that sets the value of the imaginary part

A function that returns the magnitude of the complex number

A function that returns the angle (in degrees) of the complex number

A function that returns the complex conjugate of the complex number

A function that displays the number in the form “real + j imaginary”

A function that displays the number in the form “magnitude at angle degrees”

Overload the following mathematical operators to accept complex objects as operands:

+ Complex Number Addition

– Complex Number Subtraction

* Complex Number Multiplication

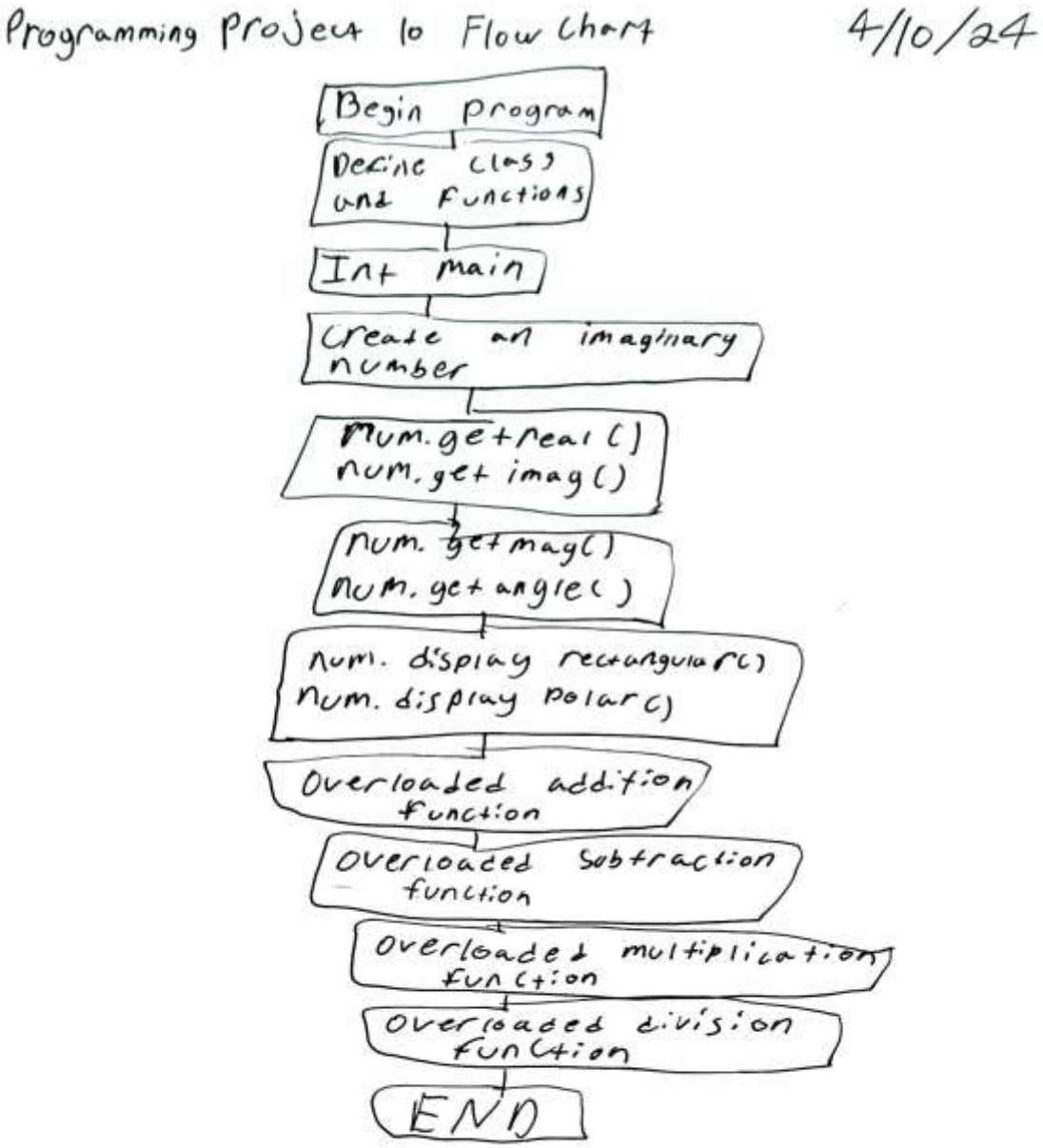
/ Complex Number Division

Write a main function that demonstrates all of the capabilities of the class definition.

Document the project with the following:

A. A written description of the program.
This program takes the real and imaginary parts of complex number and displays the rectangular form, polar form, and does different functions such as addition and multiplication on them. It uses class definitions and functions to do so.

B. A diagram to graphically illustrate the structure of the program.



C. The code listing.

```
1   //ECE 2305-Programming Project 10-Kaleb Badgett
2   #include <iostream>
3   #include <cmath>
4   using namespace std;
5
6   double const PI = 4 * atan(1);
7
8   class imaginaryNum
9   {
10  private:
11      double real;
12      double imag;
13  public:
14      imaginaryNum() { real = 0; imag = 0; }
15      imaginaryNum(double r, double i)
16      {
17          real = r; imag = i;
18      }
19      double getReal() const { return real; }
20      double getImag() const { return imag; }
21      void setReal(double r);
22      void setImag(double i);
23      double getMag(double r, double i);
24      double getAngle(double r, double i);
25      void conjugate(double r, double i);
26      void displayRect(imaginaryNum& r);
27      void displayPolar(imaginaryNum& r);
28      imaginaryNum operator + (imaginaryNum& r);
29      imaginaryNum operator - (imaginaryNum& r);
30      imaginaryNum operator * (imaginaryNum& r);
31      imaginaryNum operator / (imaginaryNum& r);
32  };
33
34  void imaginaryNum::setReal(double r)
35  {
36      real = r;
37  }
```

```
void imaginaryNum::setReal(double r)
{
    real = r;
}
void imaginaryNum::setImag(double i)
{
    imag = i;
}
double imaginaryNum::getMag(double r, double i)
{
    double mag = 0;
    mag = sqrt(pow(r, 2) + pow(i, 2));
    return mag;
}
double imaginaryNum::getAngle(double r, double i)
{
    double angle = 0;
    angle = atan2(i, r) * 180 / PI;
    return angle;
}
void imaginaryNum::conjugate(double r, double i)
{
    cout << "num1.conjugate = " << r << " - j * " << i << endl;
}
void imaginaryNum::displayRect(imaginaryNum& r)
{
    cout << r.real << " + " << r.imag;
}
void imaginaryNum::displayPolar(imaginaryNum& r)
{
    cout << "Polar form = Magnitude of " << r.getMag(r.getReal(), r.getImag()) << " at " << r.getAngle(r.getReal(), r.getImag()) << " degrees" << endl;
}
imaginaryNum imaginaryNum::operator + (imaginaryNum& r)
{
    double tempr = real + r.real;
    double tempi = imag + r.imag;
    imaginaryNum tempc(tempr, tempi);
    return tempc;
}
```

```
imaginaryNum imaginaryNum::operator * (imaginaryNum& r)
{
    double tempr = real*(r.real+r.imag);
    double tempi = imag * (r.real+r.imag);
    imaginaryNum tempc(tempr, tempi);
    return tempc;
}
imaginaryNum imaginaryNum::operator - (imaginaryNum& r)
{
    double tempr = real - r.real;
    double tempi = imag - r.imag;
    imaginaryNum tempc(tempr, tempi);
    return tempc;
}

imaginaryNum imaginaryNum::operator / (imaginaryNum& r)
{
    double tempr = (real * r.real)/(r.real * r.real);
    double tempi = (imag * r.imag) / (r.imag * (-r.imag));
    imaginaryNum tempc(tempr, tempi);
    return tempc;
}

```

```
97  int main()
98  {
99      double mag = 0;
100
101     cout << "Imaginary Number" << endl << endl;
102     imaginaryNum num1(9, 5);
103     cout << "imaginaryNum num1 ==> " << &num1 << "\t\t";
104     cout << "num1.getReal = " << num1.getReal() << "\t";
105     cout << "num1.getImag = " << num1.getImag() << endl << endl;
106
107     num1.setReal(3);
108     num1.setImag(12);
109
110     cout << "imaginaryNum num1 ==> " << &num1 << "\t\t";
111     cout << "num1.getReal = " << num1.getReal() << "\t";
112     cout << "num1.getImag = " << num1.getImag() << endl << endl;
113
114     cout << "num1.getMag = " << num1.getMag(num1.getReal(), num1.getImag()) << endl << endl;
115     cout << "num1.getAngle = " << num1.getAngle(num1.getReal(), num1.getImag()) << endl << endl;
116
117     num1.conjugate(num1.getReal(), num1.getImag());
118     cout << endl;
119     cout << "Rectangular form = ";
120     num1.displayRect(num1);
121     cout << " * j  " << endl;
122     cout << endl;
123     num1.displayPolar(num1);
124
125     cout << endl;
126
127     imaginaryNum num2(4, 8);
128     imaginaryNum num3(13, 26);
129     imaginaryNum num4 = num2 + num3;
130
```

```
    cout << "num2 for overloaded addition operator:" << endl;
    cout << "Rectangular form = ";
    num2.displayRect(num2);
    cout << " * j  " << endl;
    cout << endl << "num3 for overloaded addition operator:" << endl;
    cout << "Rectangular form = ";
    num2.displayRect(num2);
    cout << " * j  " << endl;
    cout << endl << "num4 for overloaded addition operator:" << endl;
    cout << "Rectangular form = ";
    num3.displayRect(num3);
    cout << " * j  " << endl;
    cout << endl;

    imaginaryNum num5(15, 2);
    imaginaryNum num6(3, 7);
    imaginaryNum num7 = num5 - num6;

    cout << "num5 for overloaded subtraction operator:" << endl;
    cout << "Rectangular form = ";
    num5.displayRect(num5);
    cout << " * j  " << endl;
    cout << endl << "num9 for overloaded subtraction operator:" << endl;
    cout << "Rectangular form = ";
    num6.displayRect(num6);
    cout << " * j  " << endl;
    cout << endl << "num7 for overloaded subtraction operator:" << endl;
    cout << "Rectangular form = ";
    num7.displayRect(num7);
    cout << " * j  " << endl;
    cout << endl;
```

```
161     imaginaryNum num8(4, 8);
162     imaginaryNum num9(13, 26);
163     imaginaryNum num10 = num8 * num9;
164
165     cout << "num8 for overloaded multilpication operator:" << endl;
166     cout << "Rectangular form = ";
167     num8.displayRect(num8);
168     cout << endl << endl << "num9 for overloaded multilpication operator:" << endl;
169     cout << "Rectangular form = ";
170     num9.displayRect(num9);
171     cout << endl << endl << "num10 for overloaded multilpication operator:" << endl;
172     cout << "Rectangular form = ";
173     num10.displayRect(num10);
174     cout << endl << endl;
175
176     imaginaryNum num11(4, 7);
177     imaginaryNum num12(3, 8);
178     imaginaryNum num13 = num11 / num12;
179
180     cout << "num11 for overloaded multilpication operator:" << endl;
181     cout << "Rectangular form = ";
182     num11.displayRect(num11);
183     cout << " * j  " << endl;
184     cout << endl << endl << "num12 for overloaded multilpication operator:" << endl;
185     cout << "Rectangular form = ";
186     num12.displayRect(num12);
187     cout << " * j  " << endl;
188     cout << endl << endl << "num13 for overloaded multilpication operator:" << endl;
189     cout << "Rectangular form = ";
190     num13.displayRect(num13);
191     cout << " * j  " << endl;
192     cout << endl;
193 }
```

D. Screen shots that show the operation of the program.

```
Imaginary Number

imaginaryNum num1 ==> 000000B3D9FCF518          num1.getReal = 9        num1.getImag = 5

imaginaryNum num1 ==> 000000B3D9FCF518          num1.getReal = 3        num1.getImag = 12

num1.getMag = 12.3693

num1.getAngle = 75.9638

num1.conjugate = 3 - j * 12

Rectangular form = 3 + 12 * j

Polar form = Magnitude of 12.3693 at 75.9638 degrees

num2 for overloaded addition operator:
Rectangular form = 4 + 8 * j

num3 for overloaded addition operator:
Rectangular form = 4 + 8 * j

num4 for overloaded addition operator:
Rectangular form = 13 + 26 * j

num5 for overloaded subtraction operator:
Rectangular form = 15 + 2 * j

num9 for overloaded subtraction operator:
Rectangular form = 3 + 7 * j

num7 for overloaded subtraction operator:
Rectangular form = 12 + -5 * j
```

```
num8 for overloaded multilpication operator:
Rectangular form = 4 + 8

num9 for overloaded multilpication operator:
Rectangular form = 13 + 26

num10 for overloaded multilpication operator:
Rectangular form = 156 + 312

num11 for overloaded division operator:
Rectangular form = 4 + 7 * j


num12 for overloaded division operator:
Rectangular form = 3 + 8 * j


num13 for overloaded division operator:
Rectangular form = 1.33333 + -0.875 * j
```

Submit the document on Blackboard.