P3- Imaginary Numbers

Final

3.1 Implementing Complex Class

```
int main() {
    Comp n(5, 12);
    Comp m(2, 1);
    Comp x(1, 1);

    cout << "Original Value n: ";
    n.printNumber();

    cout << "Original Value m: ";
    m.printNumber();
    cout << endl;


    x = n + m;
    cout << "n + m: ";
    x.printNumber();

    x = n + 5;
    cout << "n + 5: ";
    x.printNumber();
    cout << endl;


    x = n - m;
    cout << "n - m: ";
    x.printNumber();

    x = n - 5;
    cout << "n - 5: ";
    x.printNumber();
    cout << endl;


    x = n * m;
    cout << "n * m: ";
    x.printNumber();

    x = n * 5;
    cout << "n * 5: ";
    x.printNumber();
    cout << endl;


    x = n / m;
    cout << "n / m: ";
    x.printNumber();

    x = n / 5;
    cout << "n / 5: ";
    x.printNumber();
    cout << endl;
```

```
D:\2020+21\ELEC1204 - Advanced Programming\P3\src\class>main.exe
Original Value n: 5 + 12i
Original Value m: 2 + 1i

n + m: 7 + 13i
n + 5: 10 + 12i

n - m: 3 + 11i
n - 5: 0 + 12i

n * m: -2 + 29i
n * 5: 25 + 60i

n / m: 4.4 + 3.8i
n / 5: 1 + 2.4i
```

3.2 Converting RLC to impedance

```
int main() {

    float freq = 50;          //50Hz
    float R = 12;             //12 R
    float L = 0.15;           //0.15H
    float C = 0.0001;         //100uF

    float ang = 2 * M_PI * freq;

    float Xl = ang * L;
    float Xc = 1 / (ang * C);


    Comp r(12);
    Comp l(0.0, Xl);
    Comp c(0.0, Xc);
    Comp Xt;


    Xt = r + (l - c);

    cout << "Total Impedence: ";
    Xt.printNumber();

    return 0;
}
```

```
D:\2020+21\ELEC1204 - Advanced Programming\P3\src\RLC>main.exe
Total Impedence: 12 + 15.2929i
```

$$\omega = 2\pi f = 100\pi = 314.159$$

$$Z_T = R + \left(\omega L - \frac{1}{\omega C}\right) i = 12 + \ 15.2928354i$$

The answers match so I can confirm the code works.

3.3 RLC Circuit Simulation

To better implement this I created a new class besides Comp which is Polar. I then allowed seamless conversion between Comp and Polar using overloaded operators.

```
class Polar {

    public:
    Polar() {
        radius = 0.0;
        angle = 0.0;
    }

    Polar(float r, float o) {
        radius = r;
        angle = o;
    }

    Polar operator= (const Comp &n) {
        radius = sqrt((n.re * n.re) + (n.im * n.im));
        angle = atan(n.im / n.re);
    }


    Comp toComplex (void) {
        Comp x;
        x.re = radius * cos(angle);
        x.im = radius * sin(angle);
        return x;
    }


    void printNumber(void) {
        cout << radius << ", " << angle << "r" << endl;
    }

    float radius;
    float angle;
};
```

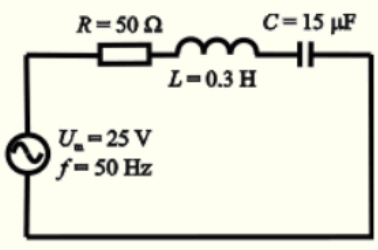
```
D:\2020+21\ELEC1204 - Advanced Programming\P3\src\RLC_Complex>main.exe
Complex: 3 + 4i
Polar: 5, 0.927295r
```

This makes it easier to deal with voltage and current which is in polar form and impedance which is in imaginary number form.

```cpp
int main() {
    float r, l, c, vm, va, freq, angv;


    cout << "Resistance (R): ";
    cin >> r;

    cout << "Inductance (H): ";
    cin >> l;

    cout << "Capacitance (F): ";
    cin >> c;



    cout << "Voltage Magnitude (V): ";
    cin >> vm;

    cout << "Voltage Phase (degrees): ";
    cin >> va;

    cout << "Circuit Frequency: ";
    cin >> freq;


    angv = 2 * M_PI * freq;

    Comp R(r);
    Comp L(0.0, l * angv);
    Comp C(0.0, 1 / (c * angv));


    Polar V(vm, (va * M_PI) / 180);
    Polar I;


    RLC circuit(R, L, C, V);

    I = circuit.calculateCurrent();

    I.printNumber();


    return 0;
}
```

```cpp
class RLC {
    public:

    RLC(Comp res, Comp ind, Comp cap, Polar volt) {
        r = res;
        l = ind;
        c = cap;
        v = volt;
    }

    Polar calculateCurrent() {
        Comp xt, ic, vc;
        Polar i;


        xt = r + (l - c);           //calculate impedence
        vc = v.toComplex();                     //convert V to complex form

        ic = vc / xt;               //I = V / Z
        i = ic;                     //convert I to polar form

        return i;
    }

    private:
    Comp r, l, c;
    Polar v;
};
```

```
D:\2020+21\ELEC1204 - Advan
Resistance (R): 50
Inductance (H): 0.3
Capacitance (F): 0.000015
Voltage Magnitude (V): 25
Voltage Phase (degrees): 0
Circuit Frequency: 50
0.195132, 1.16988r
```



http://physicstasks.eu/

In the series RLC circuit the amplitude of the current is approximately:

$$I_m = 0.2 \text{ A}.$$

The phase difference between the voltage and the current is about:

$$\varphi = -67°.$$

4 Optional Additional Work

I had already included the ability to do this in my Polar class. This class can take in either and angle and a radius or can be set equal to a complex number (Comp class). This is displayed working below.

```
class Polar {

    public:
    Polar() {
        radius = 0.0;
        angle = 0.0;
    }

    Polar(float r, float o) {
        radius = r;
        angle = o;
    }

    Polar operator= (const Comp &n) {
        radius = sqrt((n.re * n.re) + (n.im * n.im));
        angle = atan(n.im / n.re);
    }

    Comp toComplex (void) {
        Comp x;
        x.re = radius * cos(angle);
        x.im = radius * sin(angle);
        return x;
    }


    void printNumber(void) {
        cout << radius << ", " << angle << "r" << endl;
    }

    float radius;
    float angle;
};
```

```
D:\2020+21\ELEC1204 - Advanced Programming\P3\src\RLC_Complex>main.exe
Complex: 3 + 4i
Polar: 5, 0.927295r
```

Complete Class Code

```cpp
#include <iostream>
//#include "PolarNumbers.cpp"
#include <cmath>

using namespace std;

class Comp;
class Polar;



class Comp {
    public:

    Comp() {
        re = 0.0;
        im = 0.0;
    }

    Comp(float real) {
        re = real;
        im = 0.0;
    }

    Comp(float real, float imaginary) {
        re = real;
        im = imaginary;
    }


    Comp operator= (const Comp &n) {
        //a + bi = c + di
        re = n.re;
        im = n.im;
    }

    Comp operator= (const float n) {
        //a + bi = n + 0i
        re = n;
        im = 0.0;
    }




    Comp operator+ (const Comp &n) {
        //(a + bi) + (c + di) = (a + c) + (b + d)i
```

```
    Comp x;
    x.re = this->re + n.re;
    x.im = this->im + n.im;
    return x;
}

Comp operator+ (const float n) {
    //(a + bi) + n = (a + n) + (b)i
    Comp x;
    x.re = this->re + n;
    x.im = this->im;
    return x;
}


Comp operator- (const Comp &n) {
    //(a + bi) - (c + di) = (a - c) + (b - d)i
    Comp x;
    x.re = this->re - n.re;
    x.im = this->im - n.im;
    return x;
}

Comp operator- (const float n) {
    //(a + bi) - n = (a - n) + (b - n)i
    Comp x;
    x.re = this->re - n;
    x.im = this->im;
    return x;
}


Comp operator* (const Comp &n) {
    //(a + bi) * (c + di) = (ac - bd) + (ad + bc)i
    Comp x;
    x.re = (this->re * n.re) - (this->im * n.im);
    x.im = (this->re * n.im) + (this->im * n.re);
    return x;
}

Comp operator* (const float n) {
    //(a + bi) * n = (an) + (bn)i
    Comp x;
    x.re = this->re * n;
    x.im = this->im * n;
    return x;
}
```

```
    Comp operator/ (const Comp &n) {
        //(a + bi) / (c + di) = (ac + bd) / (c^2 + d^2) + (bc - ad)i / (c^2 +
d^2)
        float denom = (n.re * n.re) + (n.im * n.im);        //common denominat
or makes reading easier
        Comp x;
        x.re = ((this->re * n.re) + (this->im * n.im)) / denom;
        x.im = ((this->im * n.re) - (this->re * n.im)) / denom;
        return x;
    }

    Comp operator/ (const float n) {
        //(a + bi) / n = (a / n) + (b / n)i
        Comp x;
        x.re = this->re / n;
        x.im = this->im / n;
        return x;
    }



    void printNumber(void) {
        cout << re;
        cout << ((im < 0) ? "- " : " + ");
        cout << im << "i" << endl;
    }

    float re;
    float im;
};



class Polar {

    public:
    Polar() {
        radius = 0.0;
        angle = 0.0;
    }

    Polar(float r, float o) {
        radius = r;
        angle = o;
    }

    Polar operator= (const Comp &n) {
        radius = sqrt((n.re * n.re) + (n.im * n.im));
        angle = atan(n.im / n.re);
```

```
    }

    Comp toComplex (void) {
        Comp x;
        x.re = radius * cos(angle);
        x.im = radius * sin(angle);
        return x;
    }


    void printNumber(void) {
        cout << radius << ", " << angle << "r" << endl;
    }

    float radius;
    float angle;
};


class RLC {
    public:

    RLC(Comp res, Comp ind, Comp cap, Polar volt) {
        r = res;
        l = ind;
        c = cap;
        v = volt;
    }

    Polar calculateCurrent() {
        Comp xt, ic, vc;
        Polar i;


        xt = r + (l - c);           //calculate impedence
        vc = v.toComplex();                     //convert V to complex form

        ic = vc / xt;               //I = V / Z
        i = ic;                     //convert I to polar form

        return i;
    }

    private:
    Comp r, l, c;
    Polar v;
};
```