

## 1. Цель работы

Изучить механизм перегрузки операторов для типов, определенных пользователем посредством использования методов класса и дружественных функций.

## 2. Задание

Согласно варианту №5:

Разработать класс «Комплексное число». Определить в нем конструктор, перегрузить арифметические операции, операции сравнения, операцию преобразования в строку, метод получения комплексного числа из строки.

## 3. Листинг программы

```
#include <iostream>
#include <math.h>
#include <string>
#include <sstream>

using namespace std;

class Complex
{
private:
    float real;
    float imag;
public:
    void setReal(float);
    void setImag(float);
    void fromString(string);
    float Module();
    string toString();
    Complex(float, float);
    friend Complex operator +(const Complex&, const Complex&);
    Complex operator -(const Complex&);
    Complex operator *(const Complex&);
    Complex operator /(const Complex&);
    bool operator >(Complex&);
    bool operator <(Complex&);
    bool operator ==(Complex&);
    bool operator >=(Complex&);
    bool operator <=(Complex&);
};

void Complex::setReal(float real)
{
    this->real = real;
}

void Complex::setImag(float imag)
{
    this->imag = imag;
}

float Complex::Module()
{
    return pow(pow(real, 2) + pow(imag, 2), 0.5);
}

Complex operator +(const Complex&c1, const Complex& c2)
{
    return Complex(c1.real + c2.real, c1.imag + c2.imag);
}
```

```

}

Complex Complex::operator -(const Complex& c1)
{
    return Complex(this->real - c1.real, this->imag - c1.imag);
}

Complex Complex::operator *(const Complex& c1)
{
    return Complex((this->real * c1.real - this->imag * c1.imag), (this->real *
c1.imag + this->imag * c1.real));
}

Complex Complex::operator /(const Complex& c1)
{
    return Complex((this->real * c1.real + this->imag * c1.imag) /
(pow(c1.real,2)+pow(c1.imag,2)), (this->imag * c1.real - this->real * c1.imag) /
(pow(c1.real, 2) + pow(c1.imag, 2)));
}

bool Complex::operator >(Complex& c1)
{
    return this->Module() > c1.Module();
}

bool Complex::operator <(Complex& c1)
{
    return this->Module() < c1.Module();
}

bool Complex::operator ==(Complex& c1)
{
    return this->Module() == c1.Module();
}

bool Complex::operator >=(Complex& c1)
{
    return this->Module() >= c1.Module();
}

bool Complex::operator <=(Complex& c1)
{
    return this->Module() <= c1.Module();
}

Complex::Complex(float r = 0, float i = 0)
{
    real = r;
    imag = i;
}

string Complex::toString()
{
    stringstream result;
    if ((imag != 0) && (real != 0))
        if (imag > 0)
        {
            result << real;
            result << "+";
            if (imag != 1)
                result << imag;
            result << "i";
            return result.str();
        }
}

```

```

        else
        {
            result << real;
            if (imag == -1)
                result << "-";
            else
                result << imag;
            result << "i";
            return result.str();
        }
    else if ((imag == 0) && (real != 0))
    {
        result << real;
        return result.str();
    }
    else if ((imag != 0) && (real == 0))
    {
        if (imag != 1)
            result << imag;
        result << "i";
        return result.str();
    }
    else
        return "0";
}

string RequestComplex()
{
    string alp = "0123456789i+-.";
    string comp;

    getline(cin, comp);

    while (comp.find(" ") != string::npos)
    {
        comp.replace(comp.find(" "), 1, "");
    }

    bool find = false;

    for (int i = 0; i < comp.length(); ++i)
    {
        if (alp.find(comp[i]) > alp.length())
        {
            find = true;
        }
    }

    if ((find) || ((comp.find("+") != string::npos || comp.find("-") != string::npos) && comp.find("i") == string::npos))
    {
        cout << "Строка " << comp << endl;
        cout << "Неверная форма записи числа" << endl;;
        return RequestComplex();
    }

    return comp;
}

void Complex::fromString(string comp)
{
    int plusPos = comp.find('+');
    int minusPos = comp.find('-');
    int i_Pos = comp.find('i');
    string num = "";
    string num2 = "";

```

```

if (minusPos == 0)
{
    minusPos = comp.find('-', 1);
}
else
{
    plusPos = comp.find('+', 1);
}
if (plusPos < comp.length())
{
    for (int i = 0; i < plusPos; ++i)
    {
        num += comp[i];
    }
    if (plusPos + 1 == i_Pos)
        num2 = "1";
    for (int i = plusPos + 1; i < i_Pos; ++i)
    {
        num2 += comp[i];
    }
}
else if (minusPos < comp.length())
{
    for (int i = 0; i < minusPos; ++i)
    {
        num += comp[i];
    }

    if (minusPos + 1 == i_Pos)
        num2 = "1";
    else
        num2 = "-";
    for (int i = minusPos + 1; i < i_Pos; ++i)
    {
        num2 += comp[i];
    }
}
else if (i_Pos < comp.length())
{
    for (int i = 0; i < i_Pos; ++i)
    {
        num2 += comp[i];
    }
}
else
{
    for (int i = 0; i < comp.length(); ++i)
    {
        num += comp[i];
    }
}

if (num == "")
    num = "0";
if (num2 == "")
    num2 = "0";

real = stof(num);
imag = stof(num2);
}

int main()

```

```

{
    setlocale(LC_ALL, "RUS");
    Complex Comp_1(2, 3);
    Complex Comp_2;

    string comp;

    cout << "Исходные числа" << endl;

    cout << "Первое: ";
    cout << Comp_1.toString() << endl;

    cout << "Модуль первого: ";
    cout << Comp_1.Module() << endl;

    cout << "Введите второе комплексное числа вида a+bi" << endl;
    comp = RequestComplex();
    Comp_2.fromString(comp);

    cout << "Второе: ";
    cout << Comp_2.toString() << endl;

    cout << "Модуль второго: ";
    cout << Comp_2.Module() << endl;

    if (Comp_1 == Comp_2)
        cout << "Числа равны" << endl;
    else if (Comp_1 > Comp_2)
        cout << "Первое число больше второго" << endl;
    else
        cout << "Второе число больше первого" << endl;

    cout << "Сложение: ";
    cout << (Comp_1 + Comp_2).toString() << endl;

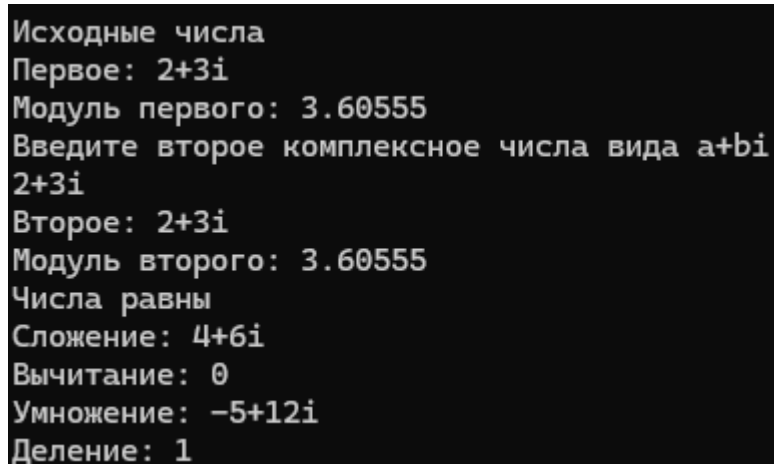
    cout << "Вычитание: ";
    cout << (Comp_1 - Comp_2).toString() << endl;

    cout << "Умножение: ";
    cout << (Comp_1 * Comp_2).toString() << endl;

    cout << "Деление: ";
    cout << (Comp_1 / Comp_2).toString() << endl;
}

```

#### 4. Пример работы программы



```

Исходные числа
Первое: 2+3i
Модуль первого: 3.60555
Введите второе комплексное числа вида a+bi
2+3i
Второе: 2+3i
Модуль второго: 3.60555
Числа равны
Сложение: 4+6i
Вычитание: 0
Умножение: -5+12i
Деление: 1

```

Рисунок 1 – Пример работы

```
Исходные числа
Первое: 2+3i
Модуль первого: 3.60555
Введите второе комплексное числа вида a+bi
2+5i
Второе: 2+5i
Модуль второго: 5.38516
Второе число больше первого
Сложение: 4+8i
Вычитание: -2i
Умножение: -11+16i
Деление: 0.655172-0.137931i
```

Рисунок 2 – Пример работы

```
Исходные числа
Первое: 2+3i
Модуль первого: 3.60555
Введите второе комплексное числа вида a+bi
1i
Второе: i
Модуль второго: 1
Первое число больше второго
Сложение: 2+4i
Вычитание: 2+2i
Умножение: -3+2i
Деление: 3-2i
```

Рисунок 3 – Пример работы

## 5. Выводы

В процессе лабораторной работы были изучены принципы перегрузки операторов, определённых пользователем посредством использования классов и дружественных функций.