КАФЕДРА №

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №3

Перегрузка операторов

по курсу: Объектно-ориентированное программирование

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №

подпись, дата	инициалы, фамилия

Санкт-Петербург 2022

Условие

6 вариант

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

Листинг программы

main.cpp

```
#include <iostream>
#include <string>
using namespace std;

class Complex {
private:
    double r, im; //-Â‡Î¸Ì‡˘ Ë ,ÓÓ·□‡Ê‡ÂÏ‡˘ ˜‡ÒÚ¸ ÍÓÏÔÎÂÍÒÌÓ„Ó
˜ËÒÎ‡

public:
    Complex();
    Complex(double, double); //ÌÛ Ú° ÔÓÌ˘Î
    ~Complex();

    double get_real(); //ÔÓÎÛ˜ËÚ¸ □Â‡Î¸ÌÛ¸ ˜‡ÒÚ¸
    double get_imaginary(); //ÔÓÎÛ˜ËÚ¸ ˚ ,ÓÓ·□ ˜‡ÒÚ¸

    Complex operator + (const Complex& z);
    Complex operator - (const Complex& z);
    friend bool operator == (Complex& z1, Complex& z2);
    friend bool operator != (Complex& z1, Complex& z2);

    string complex_to_str(); //ÔÂ□Â,ÂÒÚËË , ÙÓ□ÏÛ x + yi
    void str_to_complex(string z); // ÒÓÁ‰‡Ú¸ ËÁ ÒÚ□ÓÍË ÌÓ,°È
ÍÎ‡ÒÒ „‰Â r = x, im = y
};

Complex::Complex() {
    r = 0;
    im = 0;
}

Complex::Complex(double x, double y) {
    r = x;
    im = y;
}

Complex::~Complex() {
    cout << "”ÌË˜ÚÓÊÂÌ" << endl;
}

double Complex::get_real() {
    return r;
}

double Complex::get_imaginary() {
    return im;
}
```

```
Complex Complex::operator + (const Complex& z) {
    Complex new_z;
    new_z.r = r + z.r;
    new_z.im = im + z.im;
    return new_z;
}

Complex Complex::operator - (const Complex& z) {
    Complex new_z;
    new_z.r = r - z.r;
    new_z.im = im - z.im;
    return new_z;
}

bool operator == (Complex& z1, Complex& z2) {
    return ((z1.get_real() == z2.get_real()) && (z1.get_imagi-
nary() == z2.get_imaginary()));
}

bool operator != (Complex& z1, Complex& z2) {
    return ((z1.get_real() != z2.get_real()) && (z1.get_imagi-
nary() != z2.get_imaginary()));
}

string Complex::complex_to_str() {
    string z = to_string(r) + " + " + to_string(im) + "i";
    return z;
}

void Complex::str_to_complex(string z) {
    string x = "";
    int i = 0;
    while (z[i] != ' ') {
     x += z[i];
     i++;
    }
    r = stod(x);
    i += 3;
    x = "";
    while (z[i] != 'i') {
     x += z[i];
     i++;
    }
    im = stod(x);
}

int main() {
    setlocale(LC_ALL, "ru");
    Complex z1(5, 8);
    Complex z2(12, 10);
    cout << "ŒÂ□,ÓÂ ˜ËÒÎÓ z1 = " + z1.complex_to_str() << endl;
    cout << "¬ÚÓ□ÓÂ ˜ËÒÎÓ z1 = " + z2.complex_to_str() << endl;
    Complex z3 = z1 + z2;
    cout << "œ□Ë ÒÎÓÊÂÌËË ÔÓÎÛ˜ÂÌÓ ˜ËÒÎÓ " + z3.complex_to_str()
<< endl;
    Complex z4 = z2 - z1;
    cout << "œ□Ë ,˚˜ËÚ‡ÌËË ÔÓÎÛ˜ÂÌÓ ˜ËÒÎÓ " + z4.complex_to_str()
<< endl;
    if (z1 == z2) cout << "z1 Ë z2 □‡,Ì˚" << endl;
```

```
    if (z1 != z2) cout << "z1 Ë z2 ÌÂ □‡,Ì˚" << endl;
    string z;
    cout << "¬,Â‰ËÚÂ ÍÓÏÔÎÂÍÒÌÓÂ ˜ËÒÎÓ , ÙÓ□Ï‡ÚÂ x + yi" << endl;
    getline(cin, z);
    z1.str_to_complex(z);
    cout << "“ÂÔÂ□¸ z1 = " + z1.complex_to_str() << endl;
}
```

Скриншоты

Вывод

Мы изучили механизм перегрузки операторов для типов, определённых пользователем посредством использования методов класса и дружественных функций.