МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСТИТЕТ)

ЛАБОРАТОРНАЯ РАБОТА №2

по курсу объектно-ориентированное программирование I семестр, 2021/22 уч. год

Студент *Ханнанов Руслан Маратович, группа М8О-208Б-20*

Преподаватель *Дорохов Евгений Павлович*

Условие

Задание: Вариант 2: Комплексное число в тригонометрической форме. Разработать программу на языке C++ согласно варианту задания. Программа на C++ должна собираться с помощью системы сборки CMake. Программа должна получать данные из стандартного ввода и выводить данные в стандартный вывод. Реализовать над объектами реализовать в виде перегрузки операторов. Реализовать пользовательский литерал для работы с константами объектов созданного класса. Исходный код лежит в файле main.cpp

Протокол работы

3 60
2 15
Complex num in trigonometric form: 3*(cos60 + i * sin60)
Complex num in trigonometric form: 2*(cos15 + i * sin15)

Complex num in trigonometric form: 10*(cos40 + i * sin40)
Сложение: Complex num in trigonometric form: 4.39362*(cos-185.045 + i * sin-185.045)

Вычитание: Complex num in trigonometric form: 2.58769*(cos-121.102 + i * sin-121.102)

Умножение: Complex num in trigonometric form: 6*(cos75 + i * sin75)

Деление: Complex num in trigonometric form: 1.5*(cos45 + i * sin45)

Проверка на равенство: 0
Сопряженное число: Complex num in trigonometric form: 3*(cos-60 + i * sin-60)

Сравнение по модулю: 1

Дневник отладки

Проблем и ошибок при написании данной работы не возникло.

Недочёты

Выводы

В процессе выполнения данной лабораторной работы, я познакомился с пользовательскими литералими. Литерал - это некоторое выражение создающее объект. Я считаю, что литерал - полезный в некоторых случаях инструмент, но излишнее его использование может навредить.

Исходный код:

main.cpp

```
#include <iostream>
#include <cmath>

class Complex{

 public:
  Complex();
  Complex(double r, double j);

  double modul(Complex x);
  Complex conj(); //Сопряженное число
  Complex to_alg(Complex x); //Перевод в алгебраический вид
  Complex to_trig(Complex x); //Перевод в тригонометрический вид

  friend std::istream& operator>> (std::istream &in, Complex &num); //Перегрузка операто
  friend std::ostream& operator<< (std::ostream &out, const Complex &num); //Перегрузка
  Complex operator + (const Complex &x);
  Complex operator - (const Complex &x);
  Complex operator * (const Complex &x);
  Complex operator / (const Complex &x);
  bool operator == (const Complex &x);
  bool operator < (const Complex &x);
  bool operator > (const Complex &x);
 private:
  double r,j;
};

Complex::Complex(double r, double j) {
  this->r = r;
  this->j = j;
}

Complex Complex::to_alg(Complex x) {
  return {x.r * cos(x.j), x.r * sin(x.j)};
}

Complex Complex::to_trig(Complex x) {
  double a = x.r, b = x.j;
  double z = sqrt(a * a + b * b);
```

```
  double argZ = (-3.14 + atan(b / a)) * (180 / 3.14);
  return {z, argZ};
}

Complex Complex::conj() {
  return {this->r, -this->j};
}

Complex::Complex() {
  this->r = 0;
  this->j = 0;
}

std::ostream &operator<<(std::ostream &out, const Complex &num) {
  out << "Complex num in trigonometric form: " << num.r << "*(cos" << num.j << " + i * s
  return out;
}

std::istream &operator>>(std::istream &in, Complex &num) {
  in >> num.r >> num.j;
  return in;
}
Complex Complex::operator+(const Complex &x) {
  Complex alg_form1 = to_alg(*this);
  Complex alg_form2 = to_alg(x);
  Complex result(alg_form1.r + alg_form2.r, alg_form1.j + alg_form2.j);
  return to_trig(result);
}
Complex Complex::operator-(const Complex &x) {
  if(this->r == x.r && this->j == x.j) return {0, 0};
  Complex alg_form1 = to_alg(*this);
  Complex alg_form2 = to_alg(x);
  Complex result(alg_form1.r - alg_form2.r, alg_form1.j - alg_form2.j);
  return to_trig(result);
}
Complex Complex::operator*(const Complex &x) {
  return {this->r * x.r, this->j + x.j};
}
Complex Complex::operator/(const Complex &x) {
  if(x.r == 0 && x.j == 0){
    std::cout << "На 0 делить нельзя!" << std::endl;
    return *this;
```

```
  }
  return {this->r / x.r, this->j - x.j};
}
bool Complex::operator==(const Complex &x) {
  return this->r == x.r && this->j == x.j;
}
double Complex::modul(Complex x) {
  Complex sub = to_alg(x);
  return sqrt(sub.r * sub.r + sub.j * sub.j);
}
bool Complex::operator<(const Complex &x) {
  return modul(*this) < modul(x);
}
bool Complex::operator>(const Complex &x) {
  return modul(*this) > modul(x);
}

Complex operator "" _complex(const char* str, size_t size) {
  int cnt = 0;
  std::string s;
  while (str[cnt] != ' '){
    s += str[cnt++];
  }
  double r = 0, j = 0;
  for (char i : s) {
    r *= 10;
    r += i - '0';
  }
  s = "";
  while (str[cnt++] != '\0') {
    s += str[cnt];
  }
  for (int i = 0; i < s.size() - 1; ++i) {
    j *= 10;
    j += s[i] - '0';
  }
  Complex res(r, j);
  return res;
}

int main() {
  Complex test_num1, test_num2;
```

```
  std::cin >> test_num1;
  std::cin >> test_num2;

  std::cout << test_num1 << test_num2 << std::endl;
  //Литерал
  std::cout << "10 40"_complex << std::endl;
  //Все операции
  std::cout << "Сложение: " << test_num1 + test_num2 << std::endl;
  std::cout << "Вычитание: " << test_num1 - test_num2 << std::endl;
  std::cout << "Умножение: " << test_num1 * test_num2 << std::endl;
  std::cout << "Деление: " << test_num1 / test_num2 << std::endl;
  std::cout << "Проверка на равенство: " << bool(test_num1 == test_num2) << std::endl;
  std::cout << "Сопряженное число: " << test_num1.conj() << std::endl;
  std::cout << "Сравнение по модулю: " << bool(test_num1 > test_num2) << std::endl;

  return 0;
} //Сложение: Complex num in trigonometric form: 4.63849*(cos-109.508 + i * sin-109.508)
```