**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Объектно-ориентированное программирование»

**Лабораторная работа № 1**

Тема: Простые классы на языке С++

Студент: Чернобаев Андрей Александрович

Группа: 80-208

Преподаватель: Чернышов Л.Н.

Дата: 16.09.2020

Оценка:

Москва, 2020

1. Постановка задачи

|  |  |
| --- | --- |
| 2 | **Комплексное число в тригонометрической форме** представляются парой действительных чисел (r, j), где r – радиус (модуль), j – угол. Реализовать класс Complex для работы с комплексными числами. Обязательно должны быть присутствовать операции  - сложения add, (r1, j1) + (r2, j2);  - вычитания sub, (r1, j1) – (r2, j2);  - умножения mul, (r1, j1) ´ (r2, j2);  - деления div, (r1, j1) / (r2, j2) ;  - сравнение equ, (r1, j1) = (r2, j2), если (r1 = r2) и (j1 = j2);  - сопряженное число conj, conj(r, j) = (r, –j).  Реализовать операции сравнения по действительной части. |

1. Описание программы

* глобальная константа MAX\_ANGLE
* класс Complex
  + атрибуты
  + метод check\_bounds
  + 2 конструктора
  + перегрузки операторов
  + метод conj
  + 2 геттера
* перегрузка оператора вывода на консоль
* функция преобразования градусов в радианы
* функция help для справки взаимодействия
* цикл while (true) для ввода данных

1. Набор тестов

**Из файла test\_01.txt**

input

5

30

6

45

add

input

5

60

20

3

sub

input

7

43

121

90

mult

input

78

20

1

15

div

input

56

23

76

56

comp

input

323

12

412321

15

eq

input

12

5

0

0

conj

**из файла test\_02.txt**

input

30

30

60

15

add

radius: 89.3159 angle: 0.348843

input

30

30

60

16

sub

radius: 89.4039 angle: 0.36052

input

78

78

45

12

mult

radius: 3510 angle: 1.5708

input

144

78

56.907

42.666

div

radius: 2.53044 angle: 0.616695

input

1000

89

1

0.5

comp

1

input

1

0.5

1000

89

comp

0

input

20

20

20

20

eq

1

input

56

132

89

90

conj

radius: 56 angle: 3.97935

1. Результаты выполнения тестов

|  |  |
| --- | --- |
| input | output |
| input  5  30  6  45  add  input  5  60  20  3  sub  input  7  43  121  90  mult  input  78  20  1  15  div  input  56  23  76  56  comp  input  323  12  412321  15  eq  input  12  5  0  0  conj | radius: 10.9067 angle: 0.666466  radius: 23.1069 angle: 0.234847  radius: 847 angle: 2.32129  radius: 78 angle: 0.0872665  1  0  0  radius: 12 angle: 6.19592 |
| input  30  30  60  15  add  input  30  30  60  16  sub  input  78  78  45  12  mult  input  144  78  56.907  42.666  div  input  1000  89  1  0.5  comp  input  1  0.5  1000  89  comp  input  20  20  20  20  eq  input  56  132  89  90  conj | radius: 89.3159 angle: 0.348843  radius: 89.4039 angle: 0.36052  radius: 3510 angle: 1.5708  radius: 2.53044 angle: 0.616695  1  0  1  radius: 56 angle: 3.97935 |

1. Листинг программы

ссылка на github:

https://github.com/ruthenorum/oop\_exercise\_01

rugewit@rugewitPC:/media/rugewit/WDCDATA/MAI/3 sem/OOP/my\_code/task1/oop\_exercise\_01$ cat main.cpp

#include <iostream>

#include <cmath>

const double MAX\_ANGLE = 2 \* M\_PI;

class Complex {

private:

// radius

double r;

// angle

double j;

void check\_bounds(double& angle){

if (angle > MAX\_ANGLE){

angle -= MAX\_ANGLE;

}

if (angle < 0.0){

angle += MAX\_ANGLE;

}

}

public:

Complex(double \_r, double \_j){

r = \_r;

j = \_j;

check\_bounds(j);

}

Complex (){

r = 0;

j = 0;

}

Complex operator +(Complex const &obj){

double a1 = r \* cos(j);

double b1 = r \* sin(j);

double a2 = obj.r \* cos(obj.j);

double b2 = obj.r \* sin(obj.j);

double a3 = a1 + a2;

double b3 = b1 + b2;

double new\_r = sqrt(a3 \* a3 + b3 \* b3);

double new\_j = acos(a3 / new\_r);

return Complex(new\_r, new\_j);

}

Complex operator -(Complex const &obj){

double new\_j = obj.j + M\_PI;

check\_bounds(new\_j);

return (\*this) + obj;

}

Complex operator \*(Complex const &obj){

double new\_r = r \* obj.r;

double new\_j = j + obj.j;

check\_bounds(new\_j);

return Complex(new\_r, new\_j);

}

Complex operator /(Complex const &obj){

double new\_r = r / obj.r;

double new\_j = j - obj.j;

check\_bounds(new\_j);

return Complex(new\_r, new\_j);

}

bool operator ==(Complex const &obj){

return r == obj.r && j == obj.j;

}

bool operator >(Complex const &obj){

double a1 = r \* cos(j);

double a2 = obj.r \* cos(obj.j);

return a1 > a2;

}

Complex conj(){

double new\_j = -j;

check\_bounds(new\_j);

return Complex(r,new\_j);

}

double get\_r() const{

return r;

}

double get\_j() const{

return j;

}

};

std::ostream &operator<<(std::ostream &os, Complex const &c) {

return os << "radius: " << c.get\_r() << " angle: " << c.get\_j();

}

double from\_degrees\_to\_rad(double deg){

return deg \* M\_PI / 180.0;

}

void help(){

std::cout << "write exit for exit" << std::endl;

std::cout << "write help for help" << std::endl;

std::cout << "write input for input two variables (j in degrees) and action, format \nr1\nj1\nr2\nj2\naction" << std::endl;

std::cout << "in input mode add - addition, sub - subtraction, mult - multiplication, "

"div - division, comp - compare, eq for equal, conj for first number conjugate " << std::endl;

}

int main() {

std::string cmd;

help();

while (true){

std::cin >> cmd;

if (cmd == "exit"){

break;

}

else if (cmd == "help"){

help();

}

else if (cmd == "input"){

double r1,j1,r2,j2;

std::string action;

std::cin >> r1;

std::cin >> j1;

std::cin >> r2;

std::cin >> j2;

std::cin >> action;

Complex a = {r1,from\_degrees\_to\_rad(j1)};

Complex b = {r2,from\_degrees\_to\_rad(j2)};

if (action == "add"){

std::cout << a + b << std::endl;

}

else if (action == "sub"){

std::cout << a - b << std::endl;

}

else if (action == "mult"){

std::cout << a \* b << std::endl;

}

else if (action == "div"){

std::cout << a / b << std::endl;

}

else if (action == "comp"){

std::cout << (a > b) << std::endl;

}

else if (action == "eq"){

std::cout << (a == b) << std::endl;

}

else if (action == "conj"){

std::cout << a.conj() << std::endl;

}

}

}

return 0;

}