**Московский государственный технический**

**университет им. Н.Э. Баумана.**

Факультет «Информатика и управление»

Кафедра «Системы обработки информации и управления»

Курс «Основы информатики»

Отчет по лабораторной работе №1

«Трек курса «Функциональное программирование»»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-34 |  | преподаватель каф. ИУ5 |
| Коротков Н. К. |  | Ю.Е. Гапанюк. |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2020 г.

**Описание задания**

Составить программу на функциональном языке программирования для решения биквадратного уравнения с использованием алгоритма рассмотренного в разделе «Биквадратное уравнение» статьи https://ru.wikipedia.org/wiki/Уравнение\_четвёртой\_степени. Программа должна использовать алгебраические типы и механизм сопоставления с образцом.

В случае комплексных корней их вычисление не обязательно, можно выводить информацию о том, что корни комплексные.

**Текст программы**

open System

type BiSquareRootResult =

| NoRoots

| Roots of double \* double \* double \* double

let CalculateRoots(a:double, b:double, c:double):BiSquareRootResult =

if a=0.0 && b=0.0 then

NoRoots

else

let sD = Math.Sqrt(b\*b - 4.0\*a\*c)

let mutable rt1 = Math.Sqrt((-b+sD)/2.0\*a)

let mutable rt2 = Math.Sqrt((-b-sD)/2.0\*a)

let mutable rt3 = -rt1

let mutable rt4 = -rt2

Roots(rt1, rt2, rt3, rt4)

let fn(rt1, rt2, rt3, rt4):string =

//printf "Коэффициенты: rt1=%f, rt2=%f, rt3=%f, rt4=%f. " rt1 rt2 rt3 rt4

let mutable rt1\_2 = "null"

let mutable rt2\_2 = "null"

let mutable rt3\_2 = "null"

let mutable rt4\_2 = "null"

match rt1 with

| f when System.Double.IsNaN(f) -> rt1\_2 <- "комплексное число "

| \_ -> rt1\_2 <- rt1.ToString()

match rt2 with

| f when System.Double.IsNaN(f) -> rt2\_2 <- "комплексное число "

| \_ -> rt2\_2 <- rt2.ToString()

match rt3 with

| f when System.Double.IsNaN(f) -> rt3\_2 <- "комплексное число "

| \_ -> rt3\_2 <- rt3.ToString()

match rt4 with

| f when System.Double.IsNaN(f) -> rt4\_2 <- "комплексное число "

| \_ -> rt4\_2 <- rt4.ToString()

" "+rt1\_2+"\n"+" "+rt2\_2+"\n"+" "+rt3\_2+"\n"+" "+rt4\_2

///Вывод корней (тип unit - аналог void)

let PrintRoots(a:double, b:double, c:double):unit =

printf "Коэффициенты: a=%A, b=%A, c=%A. " a b c

let root = CalculateRoots(a,b,c)

//Оператор сопоставления с образцом

let textResult =

match root with

| NoRoots -> "Уравнение не существует"

| Roots (rt1,rt2,rt3,rt4)-> "Корни: \n" + fn(rt1,rt2,rt3,rt4)

printfn "%s" textResult

[<EntryPoint>]

let main argv =

//Тестовые данные

//4 complex

let a1 = 254.0;

let b1 = 2.0;

let c1 = 1.0;

//1 корень

let a2 = 1.0;

let b2 = 1.0;

let c2 = -1.0;

//4 кореня

let a3 = -25.0;

let b3 = 26.0;

let c3 = -1.0;

PrintRoots(a1,b1,c1)

PrintRoots(a2,b2,c2)

PrintRoots(a3,b3,c3)

//|> ignore - перенаправление потока с игнорирование результата вычисления

Console.ReadLine() |> ignore

0 // возвращение целочисленного кода выхода

**Экранные формы с примерами выполнения программы**

