**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Радиотехнический»

Кафедра «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе №3

«Решение биквадратного уравнения»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы РТ5-31Б: |  | преподаватель кафедры ИУ5 |
| Савельева В. О. |  | Гапанюк Ю. Е. |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2024 г.

Постановка задачи

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 Result =

    | NoRoots

    | OneRoot of double

    | TwoRoots of double \* double

    | ThreeRoots of double \* double \* double

    | FourRoots of double \* double \* double \* double

let FindRoots (a:double, b:double, c:double):Result =

    let D = b\*b-4.0\*a\*c;

    if D < 0.0 then NoRoots

    else if D = 0.0 then

        let rt = -b/(2.0\*a)

        if rt < 0.0 then NoRoots

        else if rt = 0.0 then OneRoot rt

        else

            let sqrtRT = Math.Sqrt(rt);

            TwoRoots (rt, -sqrtRT)

    else

        let sqrtD = Math.Sqrt(D);

        let rt1 = (-b+sqrtD)/(2.0\*a);

        let rt2 = (-b-sqrtD)/(2.0\*a);

        if rt1 > 0.0 then

            if rt2 > 0.0 then

                let sqrtRT1 = Math.Sqrt(rt1);

                let sqrtRT2 = Math.Sqrt(rt2);

                let rt5 = (-1.0)\*sqrtRT1;

                let rt6 = (-1.0)\*sqrtRT2;

                FourRoots(sqrtRT1, rt5, sqrtRT2, rt6)

            else if rt2 = 0.0 then

                let sqrtRT1 = Math.Sqrt(rt1);

                ThreeRoots(sqrtRT1, -sqrtRT1, rt2)

            else

                let sqrtRT1 = Math.Sqrt(rt1);

                TwoRoots (sqrtRT1, -sqrtRT1)

        else if rt1 < 0.0 then

            if rt2 > 0.0 then

                let sqrtRT2 = Math.Sqrt(rt2);

                TwoRoots (sqrtRT2, -sqrtRT2)

            else if rt2 = 0.0 then

                OneRoot rt2

            else NoRoots

        else

            if rt2 > 0.0 then

                let sqrtRT2 = Math.Sqrt(rt2);

                ThreeRoots (rt1, sqrtRT2, -sqrtRT2)

            else

                OneRoot rt1

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

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

    let root = FindRoots(a, b, c)

    let TextResult =

        match root with

        |NoRoots -> "No"

        |OneRoot(x) -> "x = " + x.ToString()

        |TwoRoots(x1, x2) -> "x1 = " + x1.ToString() + " x2 = " + x2.ToString()

        |ThreeRoots(x1, x2, x3) -> "x1 = " + x1.ToString() + " x2 = " + x2.ToString() + " x3 = " + x3.ToString()

        |FourRoots(x1, x2, x3, x4) -> "x1 = " + x1.ToString() + " x2 = " + x2.ToString() + " x3 = " + x3.ToString() + " x4 = " + x4.ToString()

    printfn "%s" TextResult

[<EntryPoint>]

let main argv =

    let a = 1.0;

    let b = -2.0;

    let c = 1.0;

    PrintRoots(a, b, c)

    0

Результат

