Московский государственный технический университет имени Н.Э. Баумана (МГТУ им. Н.Э. Баумана)

ОТЧЁТ ПО ДОМАШНЕМУ ЗАДАНИЮ

ПО ДИСЦИПЛИНЕ «БАЗОВЫЕ КОМПОНЕНТЫ ИНТЕРНЕТ ТЕХНОЛОГИЙ»

Выполнил(а): Белов Ф.А.

Проверил(а): Гапанюк Ю.Е.

Москва, 2017

**Формулировка задачи:**

На основе рассмотренного примера 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.

**Код**

open System

type BiquadraticRootEmpty = interface end

type NoRoots()=

interface BiquadraticRootEmpty

type OneRoot(p: double) =

interface BiquadraticRootEmpty

member val root = p: double with get, set

type TwoRoots(p1:double, p2:double)=

interface BiquadraticRootEmpty

member val root1 = p1: double with get, set

member val root2 = p2: double with get, set

type ThreeRoots(p1: double, p2: double, p3:double) =

interface BiquadraticRootEmpty

member val root1 = p1: double with get, set

member val root2 = p2: double with get, set

member val root3 = p3: double with get, set

type FourRoots(p1:double, p2:double, p3:double, p4:double) =

interface BiquadraticRootEmpty

member val root1 = p1: double with get, set

member val root2 = p2: double with get, set

member val root3 = p3: double with get, set

member val root4 = p4: double with get, set

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

let D = b\*b - 4.0\*a\*c;

if (D = 0.0) then

let x = -b/(2.0\*a)

if (x < 0.0) then (NoRoots() :> BiquadraticRootEmpty)

else if (x = 0.0) then (OneRoot(Math.Sqrt(x)) :> BiquadraticRootEmpty)

else (TwoRoots(-Math.Sqrt(x), Math.Sqrt(x)) :> BiquadraticRootEmpty)

else if (D > 0.0) then

let sqrtD = Math.Sqrt(D)

let x1 = (-b-sqrtD)/(2.0\*a);

let x2 = (-b+sqrtD)/(2.0\*a);

if (x1 > 0.0 && x2 > 0.0) then (FourRoots(-Math.Sqrt(x1),-Math.Sqrt(x2),Math.Sqrt(x1),Math.Sqrt(x2)) :> BiquadraticRootEmpty)

else if (x1 > 0.0 && x2 = 0.0) then (ThreeRoots(-Math.Sqrt(x1),x2,Math.Sqrt(x1)) :> BiquadraticRootEmpty)

else if (x1 > 0.0 && x2 < 0.0) then (TwoRoots(-Math.Sqrt(x1),Math.Sqrt(x1)) :> BiquadraticRootEmpty)

else if (x1 = 0.0 && x2 > 0.0) then (ThreeRoots(-Math.Sqrt(x2),x1,Math.Sqrt(x2)) :> BiquadraticRootEmpty)

else if (x1 = 0.0 && x2 < 0.0) then (OneRoot(x1) :> BiquadraticRootEmpty)

else if (x1 < 0.0 && x2 > 0.0) then (TwoRoots(-Math.Sqrt(x2),Math.Sqrt(x2)) :> BiquadraticRootEmpty)

else if (x1 < 0.0 && x2 = 0.0) then (OneRoot(x2) :> BiquadraticRootEmpty)

else if (x1 = 0.0 && x2 = 0.0) then (OneRoot(x1) :> BiquadraticRootEmpty)

else (NoRoots() :> BiquadraticRootEmpty)

else

(NoRoots() :> BiquadraticRootEmpty)

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

printfn "Коэффициенты: a = %A, b = %A, c = %A" a b c

let root = CalculateRoots(a, b, c)

let textResult =

match root with

| :? NoRoots -> "Корней нет"

| :? OneRoot as r -> "Один корень: " + r.root.ToString()

| :? TwoRoots as r -> "Два корня: " + r.root1.ToString() + ", " + r.root2.ToString()

| :? ThreeRoots as r -> "Три корня: " + r.root1.ToString() + ", " + r.root2.ToString() + ", " + r.root3.ToString()

| :? FourRoots as r -> "Четыре корня: " + r.root1.ToString() + ", " + r.root2.ToString() + ", " + r.root3.ToString() + ", " + r.root4.ToString()

| \_ -> ""

printfn "%s" textResult

[<EntryPoint>]

let main argv =

PrintRoots(10.0, 20.0, 30.0);

PrintRoots(3.0, -25.0, 5.0);

PrintRoots(1.0, 25.0, -1.0);

0