Московский государственный технический университет им. Н. Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»

Домашнее задание по курсу

«Базовые компоненты интернет-технологий»

Исполнитель: ИУ5-31, Черепанов Е.

Преподаватель: Гапанюк Ю.Е.

«___» _____

_____

Москва 2017г.

Задание

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 программы

```
// Learn
more about
F# at
http://fsha
rp.org
```

```
// See the 'F# Tutorial' project for more help.
open System
//Интерфейс
type BiquadraticRootEmpty = interface end
//Наследуемые классы с вариантами решения
type NoRoots()=
    interface BiquadraticRootEmpty
//Один корень
//Класс содержит параметры, которые присваиваются свойству
type OneRoot(p: double) =
    interface BiquadraticRootEmpty
    //Объявление свойства
    member val root = p: double with get, set
//Два корня
type TwoRoots(p1:double, p2:double)=
    interface BiquadraticRootEmpty
    member val root1 = p1: double with get, set
    member val root2 = p2: double with get, set
//Три корня
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
//Четыре корня
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
//============================================================
============================
//Вычисление корней
let CalculateRoots(a:double, b:double, c:double):
BiquadraticRootEmpty =
    //Вычисление "дискриминанта"
    let D = b*b - 4.0*a*c;
    //Если D равен нулю
    if (D = 0.0) then
         let x = -b/(2.0*a)
         //Явное приведение к интерфейсному типу
         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)
    //Если D больше нуля
    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
            //Четыре корня
            (FourRoots(-Math.Sqrt(x1), -Math.Sqrt(x2),
Math.Sqrt(x1), Math.Sqrt(x2)) :> BiquadraticRootEmpty)
```

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

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

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

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

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

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

        //Нет корней
        else
            (NoRoots() :> BiquadraticRootEmpty)
    //Если D меньше нуля, то нет корней
    else
        (NoRoots() :> BiquadraticRootEmpty)
```

```
//Вывод корней (тип unit - аналог void)
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
        //Оператор сопоставления с образцом по типу - :?
        | :? 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()


        // Если не выполняется ни один из предыдущих шаблонов
        | _ -> ""
    printfn "%s" textResult
[<EntryPoint>]
let main argv =
    //Тестовые корни
    let a1 = 1.0;
    let b1 = 2.0;
    let c1 = 1.0;
    PrintRoots(a1, b1, c1);
    let a2 = 4.0;
    let b2 = -100.0;
    let c2 = 13.0;
    PrintRoots(a2, b2, c2);
    let a3 = 1.0;
    let b3 = 5.0;
```

```
    let c3 = -5.0;

    PrintRoots(a3, b3, c3);

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

    Console.ReadLine() |> ignore

    0

    // возвращение целочисленного кода выхода
```

Примеры работы