**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Радиотехника»

Кафедра «Информатика и вычислительная техника»

Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе №1

«Основные конструкции языка Python»

Вариант «Усовершенствованная реализация на Haskell»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы РТ5-31Б: |  | преподаватель каф. ИУ5 |
| Щербинин А.О. |  | Гапанюк Ю.Е. |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2023 г.

**ОПИСАНИЕ ЗАДАНИЯ.**

Разработать программу для решения [биквадратного уравнения.](https://ru.wikipedia.org/wiki/%D0%A3%D1%80%D0%B0%D0%B2%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5_%D1%87%D0%B5%D1%82%D0%B2%D1%91%D1%80%D1%82%D0%BE%D0%B9_%D1%81%D1%82%D0%B5%D0%BF%D0%B5%D0%BD%D0%B8#%D0%91%D0%B8%D0%BA%D0%B2%D0%B0%D0%B4%D1%80%D0%B0%D1%82%D0%BD%D0%BE%D0%B5_%D1%83%D1%80%D0%B0%D0%B2%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5)

1. Программа должна быть разработана в виде консольного приложения на языке Python.
2. Программа осуществляет ввод с клавиатуры коэффициентов А, В, С, вычисляет дискриминант и ДЕЙСТВИТЕЛЬНЫЕ корни уравнения (в зависимости от дискриминанта).
3. Коэффициенты А, В, С могут быть заданы в виде параметров командной строки ( [вариант задания параметров приведен в конце файла с примером кода](https://github.com/ugapanyuk/BKIT_2022/blob/main/code/lab1_code) ). Если они не заданы, то вводятся с клавиатуры в соответствии с пунктом 2. [Описание работы с параметрами командной строки.](https://realpython.com/python-command-line-arguments/#the-command-line-interface)
4. Если коэффициент А, В, С введен или задан в командной строке некорректно, то необходимо проигнорировать некорректное значение и вводить коэффициент повторно пока коэффициент не будет введен корректно. Корректно заданный коэффициент - это коэффициент, значение которого может быть без ошибок преобразовано в действительное число.

**Текст программы**

module Main where

import System.Environment (getArgs)

import Control.Exception

--АДТ Отвечающий за кол-во корней уравнения

data Roots = NoRoots

| OneRoot Double

| TwoRoots Double Double

| ThreeRoots Double Double Double

| FourRoots Double Double Double Double

numberOfRoots :: Roots -> String

numberOfRoots NoRoots = "Нет корней или уравнение не является биквадратным!"

numberOfRoots (OneRoot x1) = "Один корень: " ++ show x1

numberOfRoots (TwoRoots x1 x2) = "Два корня: " ++ show x1 ++ ", " ++ show x2

numberOfRoots (ThreeRoots x1 x2 x3) = "Три корня: " ++ show x1 ++ ", " ++ show x2 ++ ", " ++ show x3

numberOfRoots (FourRoots x1 x2 x3 x4) = "Четыре корня: " ++ show x1 ++ ", " ++ show x2 ++ ", " ++ show x3 ++ ", " ++ show x4

solve :: (Double, Double, Double) -> Roots

solve (a, b, c)

| discr < 0 || discr == 0 && x < 0 || a == 0 = NoRoots

| discr == 0 && x == 0 = OneRoot 0

| discr == 0 && x > 0 = TwoRoots (sqrt x) ((-1) \* sqrt x)

| x1 > 0 && x2 < 0 = TwoRoots (sqrt x1) ((-1) \* sqrt x1)

| x1 < 0 && x2 > 0 = TwoRoots (sqrt x2) ((-1) \* sqrt x2)

| x1 > 0 && x2 == 0 = ThreeRoots (sqrt x1) ((-1) \* sqrt x1) 0

| x1 == 0 && x2 > 0 = ThreeRoots (sqrt x2) ((-1) \* sqrt x2) 0

| x1 > 0 && x2 > 0 = FourRoots (sqrt x1) ((-1) \* sqrt x1) (sqrt x2) ((-1) \* sqrt x2)

where

discr = b ^ 2 - 4 \* a \* c

x = (-b) / (2 \* a)

x1 = (-b + sqrt discr) / (2 \* a)

x2 = (-b - sqrt discr) / (2 \* a)

-- Функция для безопасного считывания коэффициентов из строки

readSafe :: Read a => IO a

readSafe = do

input <- getLine

case reads input of

[(value, "")] -> return value

\_ -> putStrLn "Ошибка: Некорректный ввод. Попробуйте снова." >> readSafe

readCoefficients :: IO (Double, Double, Double)

readCoefficients = do

args <- getArgs

case args of

[a, b, c] -> return (read a, read b, read c)

\_ -> do

putStrLn "Введите значение a:"

a <- readSafe `catch` (\(ex :: SomeException) -> print ex >> readSafe)

putStrLn "Введите значение b:"

b <- readSafe `catch` (\(ex :: SomeException) -> print ex >> readSafe)

putStrLn "Введите значение c:"

c <- readSafe `catch` (\(ex :: SomeException) -> print ex >> readSafe)

return (a, b, c)

main :: IO ()

main = do

(a, b, c) <- readCoefficients

putStr $ numberOfRoots $ solve (a, b, c)

**Примеры выводов**





