Московский государственный технический университет им. Н.Э. Баумана

Факультет ИУ
Кафедра ИУ5

Курс «Основы информатики»
Отчет по Лабораторной работе №1

Выполнил студент группы ИУ5-33Б: Уфимцев Е.Е. Подпись и дата:

Проверил преподаватель каф.: Гапанюк Ю. Е. Подпись и дата:

Москва, 2024 г

Постановка задачи:

Разработать программу для решения биквадратного уравнения.
Программа должна быть разработана в виде консольного приложения на языке Python.
Программа осуществляет ввод с клавиатуры коэффициентов A, B, C, вычисляет дискриминант и ДЕЙСТВИТЕЛЬНЫЕ корни уравнения (в зависимости от дискриминанта).
Коэффициенты A, B, C могут быть заданы в виде параметров командной строки (вариант задания параметров приведен в конце файла с примером кода). Если они не заданы, то вводятся с клавиатуры в соответствии с пунктом 2. Описание работы с параметрами командной строки.
Если коэффициент A, B, C 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.
Дополнительное задание 1 (*). Разработайте две программы на языке Python - одну с применением процедурной парадигмы, а другую с применением объектно-ориентированной парадигмы.
Дополнительное задание 2 (*). Разработайте две программы - одну на языке Python, а другую на любом другом языке программирования (кроме C++).

Code Realisation (Python):

Main.py:

```
from quadratic_equation import Equation
import sys


def get_ABC() -> list:
    abc = []
    for i in range(1, 4):
        try:
            arg = sys.argv[i]
        except:
            arg = int(input('Enter an integer: '))

        abc.append(arg)
    return abc


if __name__ == '__main__':

    abc = get_ABC()
    equation = Equation(abc)
    roots =
equation.get_biquadratic_roots(equation.get_quadratic_r
oots())
    print(roots)
```

quadratic_equation.py:

```
import math


class Equation:
    def __init__(self, abc: list):
        self.A: int = abc[0]
        self.B: int = abc[1]
        self.C: int = abc[2]
```

```
    def __str__(self):
        return f"{self.A}x^4 + {self.B}x^2  + {self.C}"

    def get_discriminant(self) -> int:
        return self.B**2 - 4*self.A*self.C

    def get_quadratic_roots(self) -> list:
        if self.get_discriminant() == 0:
            return [(-self.B) / 2*self.A]
        elif self.get_discriminant() > 0:
            return [(-self.B +
math.sqrt(self.get_discriminant())) / 2*self.A,
                    (-self.B -
math.sqrt(self.get_discriminant())) / 2*self.A]
        else:
            return []

    def get_biquadratic_roots(self, roots: list) ->
list:
        broots = []
        for i in range(len(roots)):
            if roots[i] > 0:
                broots.append(roots[i]**0.5)
                broots.append((-1)*(roots[i]**0.5))
        return broots
```

Code Realisation (Golan):

```
package main

import (
    "fmt"
    "math"
    "os"
    "strconv"
)

func getValues() (x, y, z float64) {
    fmt.Println("Main: Trying to get from console...")
    consoleArguments := os.Args[1:]
    if len(consoleArguments) == 0 {
        fmt.Println("Main: Failed to get values\nMain:
Please enter arguments: ")
        fmt.Scanln(&x, &y, &z)
    } else {
        x, _ = strconv.ParseFloat(consoleArguments[0],
64)
        y, _ = strconv.ParseFloat(consoleArguments[1],
64)
        z, _ = strconv.ParseFloat(consoleArguments[2],
64)
    }
    return
}

func getDiscriminant(a, b, c float64) float64 {
    return b*b - 4*a*c
}

func getQuadraticRoots(a, b, d float64) (qRoots
[]float64) {
    if d == 0 {
        qRoots = append(qRoots, (-1)*b/2*a)
    } else if d > 0 {
        qRoots = append(qRoots, ((-1)*b+math.Sqrt(d))/
2*a, ((-1)*b-math.Sqrt(d))/2*a)
```

```
    }
    return
}

func getRoots(qRoots []float64) (roots []float64) {
    for _, root := range qRoots {
        if root > 0 {
            roots = append(roots, math.Sqrt(root),
(-1)*math.Sqrt(root))
        }
    }
    return
}

func main() {
    a, b, c := getValues()
    fmt.Println(a, b, c)

    qRoots := getQuadraticRoots(a, b,
getDiscriminant(a, b, c))
    fmt.Println(qRoots)

    roots := getRoots(qRoots)
    fmt.Println(roots)

}
```

Tests:

3.1) Экранные формы с примерами выполнения программы:

```
Введите коэффициент A: 1
Введите коэффициент B: 1
Введите коэффициент C: 5
Нет действительных корней
```

```
Введите коэффициент A: 0
Введите коэффициент B: 0
Введите коэффициент C: 0
x принадлежит R
```

```
Введите коэффициент A: 2
Введите коэффициент B: -6
Введите коэффициент C: 4
Корни уравнения:  1.4142135623730951 -1.4142135623730951 1.0 -1.0

Process finished with exit code 0
```

```
Введите коэффициент A: 3
Введите коэффициент B: -5
Введите коэффициент C: 1
Корни уравнения:  1.1976053381271583 -1.1976053381271583 0.48208725429739563 -0.48208725429739563

Process finished with exit code 0
```

```
Введите коэффициент A: вгррмоав
Некорректный ввод. Введите число
Введите коэффициент A: 3
Введите коэффициент B: 9
Введите коэффициент C: 0
Корни уравнения:  0

Process finished with exit code 0
```

```
Введите коэффициент A: 1
Введите коэффициент B: 0
Введите коэффициент C: 4
Нет действительных корней
```