*Государственное образовательное учреждение высшего профессионального образования*



ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Системы обработки информации и управления» (ИУ-5)

**ЛАБОРАТОРНАЯ РАБОТА**

**№ 1**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Основные конструкции языка Python\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Группа ИУ5-35Б

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **16.12.2024** /**А. А. Торопыгин/** 

(Подпись, дата) (И.О.Фамилия)

Преподаватель \_\_\_\_\_\_\_\_\_\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** /**Ю. Е. Гапанюк/**

(Подпись, дата) (И.О.Фамилия)

2024

**Задание:**

Разработать программу для решения биквадратного уравнения.

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 ). Если они не заданы, то вводятся с клавиатуры в соответствии с пунктом 2. Описание работы с параметрами командной строки.
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.
5. Дополнительное задание 1 (\*). Разработайте две программы на языке Python - одну с применением процедурной парадигмы, а другую с применением объектно-ориентированной парадигмы.
6. Дополнительное задание 2 (\*). Разработайте две программы - одну на языке Python, а другую на любом другом языке программирования (кроме С++).

**Текст программы (Python):**

import sys

def get\_coef(index, prompt):

try:

coef\_str = sys.argv[index]

except:

print(prompt)

coef\_str = input()

coef = float(coef\_str)

return coef

def solve(a, b, c):

D = b\*b - 4\*a\*c

roots = []

if D > 0:

t1 = (-b + D\*\*0.5) / 2 / a

roots += solve\_square(t1)

t2 = (-b - D\*\*0.5) / 2 / a

roots += solve\_square(t2)

elif D == 0:

t = -b / 2 / a

roots += solve\_square(t)

return roots

def solve\_square(n):

roots = []

if n > 0:

roots.append(n\*\*0.5)

roots.append(-n\*\*0.5)

elif n == 0:

roots.append[n]

return roots

a = get\_coef(1, 'Введите коэффициент А:')

b = get\_coef(2, 'Введите коэффициент B:')

c = get\_coef(3, 'Введите коэффициент C:')

roots = solve(a, b, c)

roots.sort()

match len(roots):

case 0:

print("Не найдено действительных корней")

case 1:

print("Один корень: {}".format(roots[0]))

case 2:

print("Два корня: {}, {}".format(roots[0], roots[1]))

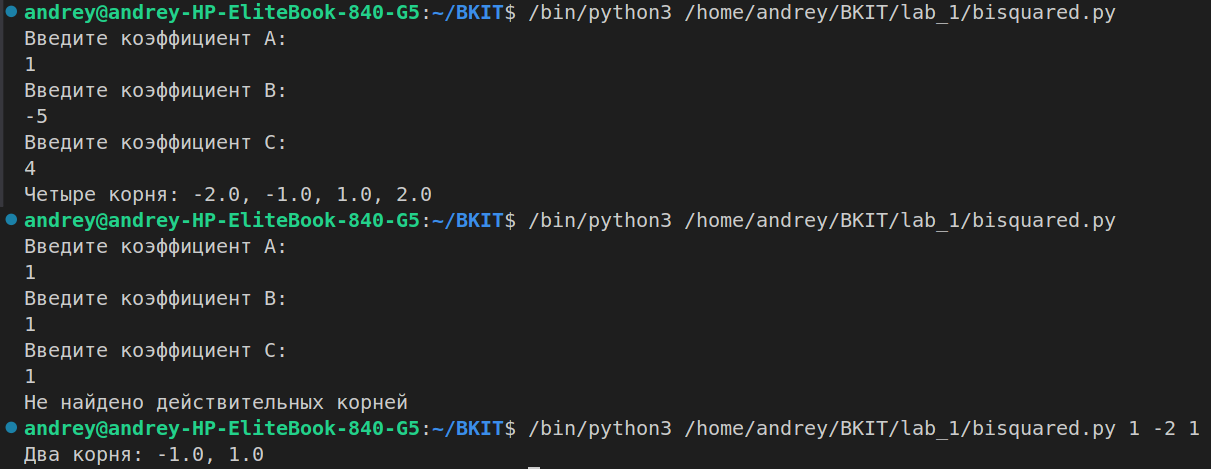
case 3:

print("Три корня: {}, {}, {}".format(roots[0], roots[1], roots[2],))

case 4:

print("Четыре корня: {}, {}, {}, {}".format(roots[0], roots[1], roots[2], roots[3]))

**Вывод:**



**Дополнительное задание 2**

**Текст программы (Rust):**

use std::env;

use std::io;

fn get\_coef(index: usize, name: &str) -> f64{;

match env::args().nth(index){

Some(string) => return string.trim().parse().unwrap\_or\_else(|error| {

panic!("Incorrect argument value")

}),

None => println!("Enter {}:", name)

}

loop{

let mut input: String = String::new();

io::stdin().read\_line(&mut input);

match input.trim().parse(){

Ok(value) => return value,

Err(\_) => println!("Try again")

}

}

}

fn solve(a: f64, b: f64, c: f64) -> [Option<f64>; 4]{

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

let mut roots: [Option<f64>; 4] = [None, None, None, None];

if d < 0.0 {return roots;}

let x1 = (-b - d.sqrt())/ (2.0\*a);

let x2 = (-b + d.sqrt())/ (2.0\*a);

if x1 >= 0.0 {

roots[0] = Some(x1.sqrt());

if x1 > 0.0 {roots[1] = Some(-x1.sqrt());}

}

if x2 == x1 {return roots;}

if x2 >= 0.0 {

roots[2] = Some(x2.sqrt());

if x2 > 0.0 {roots[3] = Some(-x2.sqrt());}

}

return roots;

}

fn main(){

let a = get\_coef(1, "a");

let b = get\_coef(2, "b");

let c = get\_coef(3, "c");

let roots = solve(a, b, c);

let mut count = 0;

for i in 0..roots.len() {

if !roots[i].is\_none(){

count += 1;

println!("x{} = {}",count, roots[i].unwrap());

}

}

if count == 0 {println!("No roots found");}

}

**Вывод:**

