**Московский государственный технический университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления» Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования» Отчет по лабораторной работе №1

«Основные конструкции языка Python.»

|  |  |
| --- | --- |
| Выполнил: | Проверил: |
| студент группы ИУ5-31Б | преподаватель каф. ИУ5 |
| Зобнин Александр |  |
| Подпись и дата: | Подпись и дата: |

# Постановка задачи

**Задание:**

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).

1. 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](https://realpython.com/python-command-line-arguments/#the-command-line-interface)

[параметрами командной строки.](https://realpython.com/python-command-line-arguments/#the-command-line-interface)

1. 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.

1. Дополнительное задание 1 (\*). Разработайте две программы на языке Python - одну с применением процедурной парадигмы, а другую с применением объектно-

ориентированной парадигмы.

1. Дополнительное задание 2 (\*). Разработайте две программы - одну на языке Python, а другую на любом другом языке программирования (кроме С++).

# Текст программы

**main.py**

import sys import math

def get\_coef(index, prompt):

*'''*

*Читаем коэффициент из командной строки или вводим с клавиатуры*

*Args:*

*index (int): Номер параметра в командной строке prompt (str): Приглашение для ввода коэффицента*

*Returns:*

*float: Коэффициент квадратного уравнения*

*'''*

while True:

try:

# Пробуем прочитать коэффициент из командной строки coef\_str = sys.argv[index]

except:

# Вводим с клавиатуры print(prompt) coef\_str = input()

# Переводим строку в действительное число try:

coef = float(coef\_str) except ValueError:

print("Некорректный ввод. Попробуйте ещё раз.") continue

else:

break

return coef

def get\_roots(a, b, c):

*'''*

*Вычисление корней квадратного уравнения*

*Args:*

* 1. *(float): коэффициент А*
  2. *(float): коэффициент B*
  3. *(float): коэффициент C*

*Returns:*

*list[float]: Список корней*

*'''*

result = []

D = b \* b - 4 \* a \* c if D == 0.0:

root = -b / (2.0 \* a) result.append(root)

elif D > 0.0:

sqD = math.sqrt(D)

root1 = (-b + sqD) / (2.0 \* a) root2 = (-b - sqD) / (2.0 \* a) result.append(root1)

result.append(root2) return result

def get\_real\_roots(result): real\_result = []

for root in result: if root > 0:

real\_result.append(math.sqrt(root)) real\_result.append(-math.sqrt(root))

elif root == 0: real\_result.append(abs(root))

return real\_result

def main():

*'''*

*Основная функция '''*

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

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

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

roots = get\_roots(a, b, c) real\_roots = get\_real\_roots(roots) # Вывод корней

len\_roots = len(real\_roots) if len\_roots == 0:

print('Нет корней') elif len\_roots == 1:

print('Один корень: {}'.format(real\_roots[0])) elif len\_roots == 2:

print('Два корня: {} и {}'.format(real\_roots[0], real\_roots[1])) elif len\_roots == 3:

print('Три корня: {}, {} и {}'.format(real\_roots[0], real\_roots[1], real\_roots[2]))

elif len\_roots == 4:

print('Четыре корня: {}, {}, {} и {}'.format(real\_roots[0], real\_roots[1], real\_roots[2], real\_roots[3]))

# Если сценарий запущен из командной строки if name == " main ":

main()

# Примеры запуска

# roots\_proc.py 1 0 10 (Нет корней)

# roots\_proc.py 1 0 -4 (Два корня)

# roots\_proc.py -4 16 0 (Три корня)

# roots\_proc.py 1 -13 36 (Четыре корня)

**main.swift**

import Foundation

// # Linear Equation Solver

func linearSolve(a: Double, b: Double) -> [Double] {

if a == 0 {

return []

}

return [Double(-b/a)]

}

// # Quadratic Equation Solver

func quadraticSolve(a: Double, b: Double, c: Double, threshold: Double = 0.0001) -> [Double] {

if a == 0 { return linearSolve(a: b, b: c) }

var roots = [Double]()

var d = pow(b, 2) - 4\*a\*c

// Check if discriminate is within the 0 threshold

if -threshold < d && d < threshold { d = 0 }

if d > 0 {

let x\_1 = Double((-b + sqrt(d))/(2\*a))

let x\_2 = Double((-b - sqrt(d))/(2\*a))

roots = [x\_1, x\_2]

} else if d == 0 {

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

roots = [x, x]

}

return roots

}

// # Biquadratic Equation Solver

func biquadrateSolve(a: Double, b: Double, c: Double) -> [Double] {

var result = [Double]()

let solutions = quadraticSolve(a: a, b: b, c: c)

for root in solutions {

if root > 0 {

result.append(-root.squareRoot())

result.append(root.squareRoot())

} else if root == 0 {

result.append(0)

}

}

return result

}

// Example of usage

let solutions = biquadrateSolve(a: 1, b: -13, c: 36)

print(solutions)

**Анализ результатов**





