МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Факультет "Информатика и системы управления"
Кафедра "Системы обработки информации и управления"

Дисциплина "Парадигмы и конструкции языков программирования"

Отчет по лабораторной работе №1
"Основные конструкции языка Python"

Выполнил:
Студент группы ИУ5-35Б
Шаньгин Н.А.
Преподаватель:
Гапанюк Ю.Е.

Москва 2025

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 A, B, C, вычисляет дискриминант и ДЕЙСТВИТЕЛЬНЫЕ корни уравнения (в зависимости от дискриминанта).
3. 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.
4. 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.
5. Дополнительное задание 1 (*). Разработайте две программы на языке Python - одну с применением процедурной парадигмы, а другую с применением объектно-ориентированной парадигмы.

Листинг кода

main.py

```
import sys
import math


def ConvertAbstractRootsToReal(abstractRoots, realRoots):
    for tempRoot in abstractRoots:
        if tempRoot >= 0:
            tempRoot = math.sqrt(tempRoot)
            negativeRoot = - tempRoot
            realRoots.append(tempRoot)
            if negativeRoot != tempRoot:
                realRoots.append(negativeRoot)


def GetAbstractRoots(a, b, c):
    abstractRoots = []
    D = b*b - 4*a*c
    if D == 0.0:
        root = -b / (2.0*a)
        abstractRoots.append(root)
    elif D > 0.0:
        sqD = math.sqrt(D)
        root1 = (-b + sqD) / (2.0*a)
        root2 = (-b - sqD) / (2.0*a)
        abstractRoots.append(root1)
        abstractRoots.append(root2)
    return abstractRoots

def PrintRoots(roots):
    # Вывод корней
    len_roots = len(roots)
    if len_roots == 0:
        print('Нет корней')
    elif len_roots == 1:
        print('Один корень: {}'.format(roots[0]))
    elif len_roots == 2:
        print('Два корня: {} и {}'.format(roots[0], roots[1]))
    elif len_roots == 3:
        print('Три корня: {}, {}, {}'.format(roots[0], roots[1], roots[2]))
    elif len_roots == 4:
        print('Четыре корня: {}, {}, {}, {}'.format(roots[0], roots[1], roots[2],
roots[3]))

def GetCoef(index, prompt):
    try:
        coefStr = sys.argv[index]
    except:
```

```
        print(prompt, end=' ')
        coefStr = input()

    isImputCorrect = False
    while (isImputCorrect == False):
        try:
            coef = float(coefStr)
            isImputCorrect = True
        except:
            print("Invalid input, try again")
            print(prompt, end=' ')
            coefStr = input()
    return coef


def MenuInput(coefs):
    coefs.append(GetCoef(1, 'Введите коэффициент A:'))
    coefs.append(GetCoef(2, 'Введите коэффициент B:'))
    coefs.append(GetCoef(3, 'Введите коэффициент C:'))
    print('\n\nРешаем биквадратное уравнение: ({})x^4+({})x^2+({}) =
0\n\n'.format(coefs[0], coefs[1], coefs[2]))


def BiquadralSolver():
    coefs = []
    MenuInput(coefs)

    realRoots = []
    abstractRoots = GetAbstractRoots(coefs[0], coefs[1], coefs[2])
    ConvertAbstractRootsToReal(abstractRoots, realRoots)

    PrintRoots(realRoots)

def main():
    BiquadralSolver()


if __name__ == "__main__":
    main()
```

main_OOP.py

```
import sys
import math


class BiquadraticEquation:
    def __init__(self, A: float, B: float, C: float):
        self.A = A
        self.B = B
        self.C = C
        self.solutions = []


    def CalculateDiscriminant(self):
        return self.B * self.B - 4 * self.A * self.C

    def CalculateSolutions(self):
        abstractRoots = self.GetAbstractRoots()
        self.ConvertAbstractRootsToReal(abstractRoots)


    def GetAbstractRoots(self):
        abstractRoots = []
        D = self.CalculateDiscriminant()
        if D == 0.0:
            root = -self.B / (2.0 * self.uation.A)
            abstractRoots.append(root)
        elif D > 0.0:
            root1 = (-self.B + D ** 0.5) / (2.0 * self.A)
            root2 = (-self.B - D ** 0.5) / (2.0 * self.A)
            abstractRoots.append(root1)
            abstractRoots.append(root2)
        return abstractRoots

    def ConvertAbstractRootsToReal(self, abstractRoots):
        for tempRoot in abstractRoots:
            if tempRoot >= 0:
                tempRoot = tempRoot ** 0.5
                negativeRoot = - tempRoot
                self.solutions.append(tempRoot)
                if negativeRoot != tempRoot:
                    self.solutions.append(negativeRoot)


class BiquadraticSolver:
    def __init__(self):
        pass


    def Start(self):
        coefs = []
```

```
        self.SolvingMenu(coefs)
        equation = BiquadraticEquation(coefs[0], coefs[1], coefs[2])
        equation.CalculateSolutions()

        self.PrintRoots(equation.solutions)


    def GetCoef(self, index, prompt):
        try:
            coefStr = sys.argv[index]
        except:
            print(prompt, end=' ')
            coefStr = input()
        isInputCorrect = False
        while (isInputCorrect == False):
            try:
                coef = float(coefStr)
                isInputCorrect = True
            except:
                print("Invalid input, try again")
                print(prompt, end=' ')
                coefStr = input()
        return coef


    def SolvingMenu(self, coefs):
        coefs.append(self.GetCoef(1, 'Введите коэффициент A:'))
        coefs.append(self.GetCoef(2, 'Введите коэффициент B:'))
        coefs.append(self.GetCoef(3, 'Введите коэффициент C:'))
        print('\n\nРешаем биквадратное уравнение: ({})x^4+({})x^2+({}) =
0\n\n'.format(coefs[0], coefs[1], coefs[2]))


    def PrintRoots(self, roots):
        # Вывод корней
        lenRoots = len(roots)
        if lenRoots == 0:
            print('Нет корней')
        elif lenRoots == 1:
            print('Один корень: {}'.format(roots[0]))
        elif lenRoots == 2:
            print('Два корня: {} и {}'.format(roots[0], roots[1]))
        elif lenRoots == 3:
            print('Три корня: {}, {}, {}'.format(roots[0], roots[1], roots[2]))
        elif lenRoots == 4:
            print('Четыре корня: {}, {}, {}, {}'.format(roots[0], roots[1],
roots[2], roots[3]))


def main():
    solver = BiquadraticSolver()
```

```
    solver.Start()


if __name__ == "__main__":
    main()
```

Результат выполнения программы

```
PS D:\Учёба\3 семестр\ПиКЯП\Shangin-PCPL-Labs-2025\lab_1> py main.py
Введите коэффициент A: 1
Введите коэффициент B: 4
Введите коэффициент C: -3


Решаем биквадратное уравнение: (1.0)x^4+(4.0)x^2+(-3.0) = 0


Два корня: 0.8035865299173393 и -0.8035865299173393
PS D:\Учёба\3 семестр\ПиКЯП\Shangin-PCPL-Labs-2025\lab_1> py main_OOP.py
Введите коэффициент A: 2
Введите коэффициент B: 5
Введите коэффициент C: -3


Решаем биквадратное уравнение: (2.0)x^4+(5.0)x^2+(-3.0) = 0


Два корня: 0.7071067811865476 и -0.7071067811865476
PS D:\Учёба\3 семестр\ПиКЯП\Shangin-PCPL-Labs-2025\lab_1>
```