Московский государственный технический
университет им. Н.Э. Баумана

Факультет информатика и системы управления
Кафедра системы обработки информации и управления

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

Выполнил:
студент группы ИУ5-32Б:
Шпакова Д. Л.
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Нардид А.Н.
Подпись и дата:

Москва, 2025 г.

Задание:

Разработать программу для решения биквадратного уравнения.

рограмма должна быть разработана в виде консольного приложения на языке

рограмма осуществляет ввод с клавиатуры коэффициентов A, B, C, вычисляет дискриминант и ДЕЙСТВИТЕЛЬНЫЕ корни уравнения (в зависимости от дискриминанта).

оэффициенты A, B, C могут быть заданы в виде параметров командной строки они не заданы, то вводятся с клавиатуры в соответствии с пунктом 2. Описание работы с параметрами командной строки.

сли 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.

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

Текст программы

`lab_1.py`

```
import sys
import math

def get_coef(index, prompt):
    while True:
        try:
            # Пробуем прочитать коэффициент из командной строки
            coef_str = sys.argv[index]
        except IndexError:
            # Вводим с клавиатуры
            coef_str = input(prompt)
        try:
            coef = float(coef_str)
            return coef
        except ValueError:
             print("Ошибка: нужно ввести число.")



def get_biquadratic_roots(a, b, c):
    result = []

    # Решаем квадратное уравнение относительно y = x^2
    D = b*b - 4*a*c
    if D < 0:
        return result  # нет действительных решений

    if D == 0.0:
        y = -b / (2.0*a)
        if y >= 0:
            result.append(math.sqrt(y))
            result.append(-math.sqrt(y))
    else:
        sqD = math.sqrt(D)
        y1 = (-b + sqD) / (2.0*a)
        y2 = (-b - sqD) / (2.0*a)
        for y in (y1, y2):
            if y >= 0:
                result.append(math.sqrt(y))
                result.append(-math.sqrt(y))

    # убираем дубликаты (например, если корень равен 0)
    result = sorted(set(result))
    return result


def main():
```

```
    a = get_coef(1, 'Введите коэффициент A: ')
    b = get_coef(2, 'Введите коэффициент B: ')
    c = get_coef(3, 'Введите коэффициент C: ')

    if a == 0:
        print("Ошибка: коэффициент A не может быть равен 0.")
        return

    roots = get_biquadratic_roots(a, b, c)

    if not roots:
        print('Действительных корней нет')
    elif len(roots) == 1:
        print('Один действительный корень: {}'.format(roots[0]))
    else:
        print('Действительные корни:', ', '.join(map(str, roots)))


# Если сценарий запущен из командной строки
if __name__ == "__main__":
    main()
```

`lab1_additional.py`

```
import sys
import math

class BiquadraticSolver:
    """
    Класс для решения биквадратных уравнений вида ax^4 + bx^2 + c = 0.
    """

    def __init__(self):
        """
        Конструктор класса. Инициализирует коэффициенты и список корней.
        """
        self.coef_A = 0.0
        self.coef_B = 0.0
        self.coef_C = 0.0
        self.roots_list = []

    def get_coef(self, index, prompt):
        # 1. Пытаемся прочитать из аргументов командной строки
        if index < len(sys.argv):
            try:
                coef = float(sys.argv[index])
                print(f'{prompt} {coef} (из командной строки)')
                return coef
            except ValueError:
```

```
                print(f"Некорректное значение '{sys.argv[index]}' в командной
строке.")

        # 2. Если аргументов нет, запрашиваем ввод в цикле
        while True:
            try:
                # Ввод будет на той же строке
                coef_str = input(f"{prompt} ")
                return float(coef_str)
            except ValueError:
                print("Ошибка: Введите корректное действительное число.")

    def get_coefs(self):
        """
        Чтение трёх коэффициентов A, B и C.
        """
        print("Решение биквадратного уравнения ax^4 + bx^2 + c = 0")
        self.coef_A = self.get_coef(1, 'Введите коэффициент A:')
        self.coef_B = self.get_coef(2, 'Введите коэффициент B:')
        self.coef_C = self.get_coef(3, 'Введите коэффициент C:')

    def calculate_roots(self):
        """
        Вычисление действительных корней биквадратного уравнения.
        """
        a, b, c = self.coef_A, self.coef_B, self.coef_C

        # Если A=0, уравнение становится квадратным: bx^2 + c = 0
        if a == 0:
            if b != 0:
                val = -c / b
                if val > 0:
                    root = math.sqrt(val)
                    self.roots_list = [-root, root]
                elif val == 0:
                    self.roots_list = [0.0]
            # Если b=0 и c=0, корней бесконечно много (любое число),
            # если b=0 и c!=0, корней нет. Для простоты оставляем список пустым.
            return

        # Решаем вспомогательное квадратное уравнение at^2 + bt + c = 0, где t =
x^2
        D = b*b - 4*a*c

        t_roots = []
        if D == 0.0:
            t_roots.append(-b / (2.0*a))
        elif D > 0.0:
            sqD = math.sqrt(D)
            t_roots.append((-b + sqD) / (2.0*a))
            t_roots.append((-b - sqD) / (2.0*a))
```

```
        # Если D < 0, действительных корней для t нет, значит и для x их тоже
нет.

        # Находим корни x из положительных корней t
        final_roots = set()
        for t in t_roots:
            if t > 0:
                x = math.sqrt(t)
                final_roots.add(x)
                final_roots.add(-x)
            elif t == 0:
                final_roots.add(0.0)

        # Сохраняем отсортированный список уникальных корней
        self.roots_list = sorted(list(final_roots))

    def print_roots(self):
        """
        Вывод вычисленных корней в консоль.
        """
        if not self.roots_list:
            print('Результат: Нет действительных корней.')
        else:
            num_roots = len(self.roots_list)
            roots_str = ', '.join(map(str, self.roots_list))
            if num_roots == 1:
                print(f'Результат: Найден один корень: {roots_str}')
            else:
                print(f'Результат: Найдено корней ({num_roots}): {roots_str}')

def main():
    """
    Основная функция.
    """
    # Создание объекта класса
    solver = BiquadraticSolver()
    # Последовательный вызов необходимых методов
    solver.get_coefs()
    solver.calculate_roots()
    solver.print_roots()

# Если сценарий запущен из командной строки
if __name__ == "__main__":
    main()
```

Скриншот работы приложения

```
dasha@Magic14:~/LABS_2025_3TERM/LABS_2025_3TERM-1/lab1$ python lab_1.py
Введите коэффициент A: 1
Введите коэффициент B: 0
Введите коэффициент C: -4
Действительные корни: -1.4142135623730951, 1.4142135623730951
```

Рисунок 1 ООП подход

```
(.venv) dasha@Magic14:~/LABS_2025_3TERM/LABS_2025_3TERM-1/lab1$ python lab1_additional.py
Решение биквадратного уравнения ax^4 + bx^2 + c = 0
Введите коэффициент A: 3
Введите коэффициент B: 4
Введите коэффициент C: 5
Результат: Нет действительных корней.
```

Рисунок 2 Процедурный подход