

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Факультет “Информатика и системы управления”
Кафедра ИУ5 “Системы обработки информации и управления”



Дисциплина “Парадигмы и конструкции языков программирования”

Отчет по лабораторной работе
“Основные конструкции языка Python”

Выполнил:
Студент группы ИУ5-33Б
Просвирякова Д. С.

Преподаватель:
Гапанюк Ю. Е.

Москва 2025

Задание

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.
6. Дополнительное задание 2 (*). Разработайте две программы – одну на языке Python, а другую на любом другом языке программирования (кроме C++).

Листинг кода

```
import sys
import math

def get_coefficient_from_user(coefficient_name):
    while True:
        try:
            value = input(f"Введите коэффициент {coefficient_name}: ")
            return float(value)
        except ValueError:
            print(f"Ошибка! Коэффициент {coefficient_name} должен быть числом.")

def parse_command_line_args():
    if len(sys.argv) >= 4:
        try:
            a = float(sys.argv[1])
            b = float(sys.argv[2])
            c = float(sys.argv[3])
            return [a, b, c]
        except ValueError:
            return None
    return None

def solve_biquadratic(a, b, c):
    D = b**2 - 4*a*c
    print(f"\nДискриминант D = {b}^2 - 4*{a}*{c} = {D}")
```

```

roots = []

if D > 0:
    t1 = (-b + math.sqrt(D)) / (2*a)
    t2 = (-b - math.sqrt(D)) / (2*a)
    print(f"t1 = {t1:.4f}, t2 = {t2:.4f}")

    if t1 > 0:
        x1 = math.sqrt(t1)
        x2 = -math.sqrt(t1)
        roots.extend([x1, x2])
        print(f"Корни из замены на t1: x = ±{math.sqrt(t1):.4f}")
    elif t1 == 0:
        roots.append(0)
        print("Корень из замены на t1: x = 0")

    if t2 > 0:
        x3 = math.sqrt(t2)
        x4 = -math.sqrt(t2)
        roots.extend([x3, x4])
        print(f"Корни из замены на t2: x = ±{math.sqrt(t2):.4f}")
    elif t2 == 0 and 0 not in roots:
        roots.append(0)
        print("Корень из замены на t2: x = 0")

elif D == 0:
    t = -b / (2*a)
    print(f"t = {t:.4f}")

    if t > 0:
        x1 = math.sqrt(t)
        x2 = -math.sqrt(t)
        roots.extend([x1, x2])
        print(f"Два корня: x = ±{math.sqrt(t):.4f}")
    elif t == 0:
        roots.append(0)
        print("Один корень: x = 0")

else:
    print("Дискриминант отрицательный.")

return roots

def main():

```

```

print("Решение биквадратного уравнения.")
print("Уравнение вида: A·x⁴ + B·x² + C = 0")

coefficients = parse_command_line_args()

if coefficients:
    a, b, c = coefficients
    print(f"\nКоэффициенты из командной строки:")
    print(f"A = {a}, B = {b}, C = {c}")
else:
    print("\nВведите коэффициенты с клавиатуры:")
    a = get_coefficient_from_user("A")

    while a == 0:
        print("Коэффициент A не может быть равен 0 для биквадратного
уравнения!")
        a = get_coefficient_from_user("A")

    b = get_coefficient_from_user("B")
    c = get_coefficient_from_user("C")

    print(f"\nУравнение: {a}·x⁴ + {b}·x² + {c} = 0")

    roots = solve_biquadratic(a, b, c)

    if roots:
        unique_roots = []
        for root in roots:
            if root not in unique_roots:
                unique_roots.append(root)

        unique_roots.sort()

        print(f"Действительные корни: {len(unique_roots)}")
        for i, root in enumerate(unique_roots, 1):
            print(f"x{i} = {root:.6f}")
    else:
        print("Действительных корней нет.")

if __name__ == "__main__":
    main()
import sys
import math

class BiquadraticEquation:

```

```

def __init__(self, a=0, b=0, c=0):
    self.a = a
    self.b = b
    self.c = c
    self.roots = []
    self.discriminant = None

def calculate_discriminant(self):
    self.discriminant = self.b**2 - 4*self.a*self.c
    return self.discriminant

def solve(self):
    self.calculate_discriminant()
    self.roots = []

    print(f"\nДискриминант D = {self.b}^2 - 4*{self.a}*{self.c} = {self.discriminant}")

    if self.discriminant > 0:
        t1 = (-self.b + math.sqrt(self.discriminant)) / (2*self.a)
        t2 = (-self.b - math.sqrt(self.discriminant)) / (2*self.a)
        print(f"t1 = {t1:.4f}, t2 = {t2:.4f}")

        self._add_roots_from_t(t1, "t1")
        self._add_roots_from_t(t2, "t2")

    elif self.discriminant == 0:
        t = -self.b / (2*self.a)
        print(f"t = {t:.4f}")
        self._add_roots_from_t(t, "t")

    else:
        print("Дискриминант отрицательный.")

    return self.get_unique_roots()

def _add_roots_from_t(self, t, t_name):
    if t > 0:
        x1 = math.sqrt(t)
        x2 = -math.sqrt(t)
        self.roots.extend([x1, x2])
        print(f"Корни из замены на {t_name}: x = ±{math.sqrt(t):.4f}")
    elif t == 0:
        self.roots.append(0)

```

```

print(f"Корень из замены на {t_name}: x = 0")

def get_unique_roots(self):
    unique_roots = []
    for root in self.roots:
        if root not in unique_roots:
            unique_roots.append(root)
    unique_roots.sort()
    return unique_roots

def display_solution(self):
    print(f"\nУравнение: {self.a}·x4 + {self.b}·x2 + {self.c} = 0")
    roots = self.solve()

    if roots:
        print(f"\nДействительные корни: {len(roots)}")
        for i, root in enumerate(roots, 1):
            print(f"x{i} = {root:.6f}")
    else:
        print("Действительных корней нет.")

class EquationSolverApp:

    def __init__(self):
        self.equation = None

    def get_valid_coefficient(self, name):
        while True:
            try:
                value = input(f"Введите коэффициент {name}: ")
                coefficient = float(value)
                if name == 'A' and coefficient == 0:
                    print("Коэффициент A не может быть равен 0!")
                    continue
                return coefficient
            except ValueError:
                print(f"Ошибка! Коэффициент {name} должен быть числом.")

    def parse_arguments(self):
        if len(sys.argv) >= 4:
            try:
                a = float(sys.argv[1])
                b = float(sys.argv[2])
                c = float(sys.argv[3])
            except ValueError:
                print(f"Ошибка! Коэффициенты должны быть числами.")



```

```
        return [a, b, c]
    except ValueError:
        return None
    return None

def run(self):
    print("Решение биквадратного уравнения.")
    print("Уравнение вида: A·x⁴ + B·x² + C = 0")

    coefficients = self.parse_arguments()

    if coefficients:
        a, b, c = coefficients
        print(f"\nКоэффициенты из командной строки:")
        print(f"A = {a}, B = {b}, C = {c}")
        self.equation = BiquadraticEquation(a, b, c)
    else:
        print("\nВведите коэффициенты с клавиатуры:")
        a = self._get_valid_coefficient('A')
        b = self._get_valid_coefficient('B')
        c = self._get_valid_coefficient('C')
        self.equation = BiquadraticEquation(a, b, c)

    self.equation.display_solution()

def main():
    app = EquationSolverApp()
    app.run()

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

Скриншот работы приложения

```
Уравнение вида: A·x4 + B·x2 + C = 0

Введите коэффициенты с клавиатуры:
Введите коэффициент A: 1
Введите коэффициент B: 9
Введите коэффициент C: 2

Уравнение: 1.0·x4 + 9.0·x2 + 2.0 = 0

Дискриминант D = 9.02 - 4*1.0*2.0 = 73.0
t1 = -0.2280, t2 = -8.7720
Действительных корней нет.

● daraprosvirakova@MacBook-Air-Dara-2 Prosviryakova-PCPL % python3 main.py
Решение биквадратного уравнения.

Уравнение вида: A·x4 + B·x2 + C = 0

Введите коэффициенты с клавиатуры:
Введите коэффициент A: 1
Введите коэффициент B: -5
Введите коэффициент C: 4

Уравнение: 1.0·x4 + -5.0·x2 + 4.0 = 0

Дискриминант D = -5.02 - 4*1.0*4.0 = 9.0
t1 = 4.0000, t2 = 1.0000
Корни из замены на t1: x = ±2.0000
Корни из замены на t2: x = ±1.0000
Действительные корни: 4
x1 = -2.000000
x2 = -1.000000
x3 = 1.000000
x4 = 2.000000

● daraprosvirakova@MacBook-Air-Dara-2 Prosviryakova-PCPL % python3 main2.py
Решение биквадратного уравнения.

Уравнение вида: A·x4 + B·x2 + C = 0

Введите коэффициенты с клавиатуры:
Введите коэффициент A: 1
Введите коэффициент B: -5
Введите коэффициент C: 4

Уравнение: 1.0·x4 + -5.0·x2 + 4.0 = 0

Дискриминант D = -5.02 - 4*1.0*4.0 = 9.0
t1 = 4.0000, t2 = 1.0000
Корни из замены на t1: x = ±2.0000
Корни из замены на t2: x = ±1.0000

Действительные корни: 4
x1 = -2.000000
x2 = -1.000000
x3 = 1.000000
x4 = 2.000000
```