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
Евдокимов М. С.
Преподаватель:
Гапанюк Ю. Е.

Москва 2025

Цель лабораторной работы

Цель лабораторной работы: изучение основных конструкций языка Python.

Задание лабораторной работы

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 программы

Файл biquadratic_functional.py – решение задачи процедурной парадигмой

```
import sys
import math

def read_coefficient(name, args, index):
    if len(args) > index:
        token = args[index]
        try:
            return float(token)
        except ValueError:
            print(f"Ошибка! Коэффициент {name} в командной строке ('{token}') не является числом.")
            while True:
                s = input(f"Введите коэффициент {name}: ")
                try:
                    return float(s)
                except ValueError:
                    print(f"Ошибка! Коэффициент {name} должен быть числом.")
    else:
        while True:
            s = input(f"Введите коэффициент {name}: ")
            try:
                return float(s)
            except ValueError:
                print(f"Ошибка! Коэффициент {name} должен быть числом.")

def solve_quadratic(a, b, c):
    D = b*b - 4*a*c
    print(f"Дискриминант: {D}")
    if D < 0:
        return D, []
    elif D == 0:
        x = -b / (2*a)
        return D, [x]
    else:
        sqrtD = math.sqrt(D)
        x1 = (-b + sqrtD) / (2*a)
        x2 = (-b - sqrtD) / (2*a)
        return D, sorted([x1, x2])

args = sys.argv[1:]
a = read_coefficient("A", args, 0)
while a == 0:
    print("Коэффициент A не может быть равен нулю для квадратного уравнения.")
    a = read_coefficient("A", args, 0)

b = read_coefficient("B", args, 1)
c = read_coefficient("C", args, 2)

D, roots = solve_quadratic(a, b, c)
if roots:
    print("Действительные корни:", roots)
else:
    print("Действительных корней нет.")
```

Файл biquadratic_class.py – решение задачи объекто-ориентированной парадигмой

```
import sys
import math

class CoefficientReader:
    def set_args(self, args):
        self.args = args
```

```
    def read(self, name, index):
        if len(self.args) > index:
            token = self.args[index]
            try:
                return float(token)
            except ValueError:
                print(f"Ошибка! Коэффициент {name} в командной строке ('{token}') не является числом.")
                return self._read_from_input(name)
        else:
            return self._read_from_input(name)

    def _read_from_input(self, name):
        while True:
            s = input(f"Введите коэффициент {name}: ")
            try:
                return float(s)
            except ValueError:
                print(f"Ошибка! Коэффициент {name} должен быть числом.")

class QuadraticEquation:
    def set_coefficients(self, a, b, c):
        if a == 0:
            raise ValueError("Коэффициент A не может быть равен нулю для квадратного уравнения.")
        self.a = a
        self.b = b
        self.c = c

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

    def solve(self):
        D = self.discriminant()
        print(f"Дискриминант: {D}")
        if D < 0:
            return []
        elif D == 0:
            x = -self.b / (2 * self.a)
            return [x]
        else:
            sqrtD = math.sqrt(D)
            x1 = (-self.b + sqrtD) / (2 * self.a)
            x2 = (-self.b - sqrtD) / (2 * self.a)
            return sorted([x1, x2])


args = sys.argv[1:]

reader = CoefficientReader()
reader.set_args(args)

a = reader.read("A", 0)
while a == 0:
    print("Коэффициент A не может быть равен нулю для квадратного уравнения.")
    a = reader.read("A", 0)

b = reader.read("B", 1)
c = reader.read("C", 2)

equation = QuadraticEquation()
equation.set_coefficients(a, b, c)

roots = equation.solve()
if roots:
    print("Действительные корни:", roots)
else:
    print("Действительных корней нет.")
```

Скриншоты работы программы