

**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:
студент группы ИУ5-32Б
Сайфутдинов Р.И.

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.

Описание задания

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. Если они не заданы, то вводятся с клавиатуры в соответствии с пунктом 2.
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++).

Текст программы

Python

Файл lab1_oop.py

```
import sys
import math

class SquareRoots:

    def __init__(self):
        """
        Конструктор класса
        """
        # Объявление коэффициентов
        self.coef_A = 0.0
        self.coef_B = 0.0
        self.coef_C = 0.0
        # Количество корней
        self.num_roots = 0
        # Список корней
        self.roots_list = []

    def get_coef(self, index, prompt):
        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
    coef = float(coef_str)
    return coef

def get_coefs(self):

    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 корней квадратного уравнения
    """
    a = self.coef_A
    b = self.coef_B
    c = self.coef_C
    # Вычисление дискриминанта и корней
    D = b*b - 4*a*c
    if D == 0.0 and -b / (2.0*a) > 0:
        root1 = math.sqrt(-b / (2.0*a))
        root2 = -root1
        self.num_roots = 2
        self.roots_list.append(root1)
        self.roots_list.append(root2)
    elif D > 0.0:
        sqD = math.sqrt(D)
        temp1 = (-b + sqD) / (2.0*a)
        temp2 = (-b - sqD) / (2.0*a)
        if temp1 > 0:
            self.num_roots += 2
            self.roots_list.append(math.sqrt(temp1))
            self.roots_list.append(-math.sqrt(temp1))
        if temp2 > 0:
            self.num_roots += 2
            self.roots_list.append(math.sqrt(temp2))
            self.roots_list.append(-math.sqrt(temp2))

def print_roots(self):
    # Проверка отсутствия ошибок при вычислении корней
    if self.num_roots != len(self.roots_list):
        print(('Ошибка. Уравнение содержит {} действительных корней, ' + \

```

```

        'но было вычислено {} корней.').format(self.num_roots,
len(self.roots_list))
    else:
        if self.num_roots == 0:
            print('Нет корней')
        elif self.num_roots == 2:
            print('Два корня: {} и {}'.format(self.roots_list[0], \
self.roots_list[1]))
        elif self.num_roots == 4:
            print('Четыре корня: {} и {} и {} и {}'.format(self.roots_list[0], \
self.roots_list[1], \
self.roots_list[2], \
self.roots_list[3]))

def main():
    """
    Основная функция
    """

    # Создание объекта класса
    r = SquareRoots()
    # Последовательный вызов необходимых методов
    r.get_coefs()
    r.calculate_roots()
    r.print_roots()

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

# Пример запуска
# roots_oop.py 1 0 -4

```

Файл lab1_proc.py

```

import sys
import math

def get_coef(index, prompt):
    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
    coef = float(coef_str)

```

```

    return coef

def get_roots(a, b, c):
    result = []
    D = b*b - 4*a*c
    if D == 0.0 and -b / (2.0*a) > 0:
        root1 = math.sqrt(-b / (2.0*a))
        root2 = - root1
        result.append(root1)
        result.append(root2)
    elif D > 0.0:
        sqD = math.sqrt(D)
        temp1 = (-b + sqD) / (2.0*a)
        temp2 = (-b - sqD) / (2.0*a)
        if temp1 > 0:
            result.append(math.sqrt(temp1))
            result.append(-math.sqrt(temp1))
        if temp2 > 0:
            result.append(math.sqrt(temp2))
            result.append(-math.sqrt(temp2))
    return result

def main():
    """
    Основная функция
    """
    a = get_coef(1, 'Введите коэффициент A:')
    b = get_coef(2, 'Введите коэффициент B:')
    c = get_coef(3, 'Введите коэффициент C:')
    # Вычисление корней
    roots = get_roots(a,b,c)
    # Вывод корней
    len_roots = len(roots)
    if len_roots == 0:
        print('Нет корней')
    elif len_roots == 2:
        print('Два корня: {} и {}'.format(roots[0], roots[1]))
    elif len_roots == 4:
        print('Четыре корня: {} и {} и {} и {}'.format(roots[0], roots[1],
roots[2], roots[3]))

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

# Пример запуска

```

```
# roots_proc.py 1 0 -4
```

Примеры выполнения программы

1. Python

```
Биквадратное уравнение  
Введите коэффициент А:  
1  
Введите коэффициент В:  
-5  
Введите коэффициент С:  
4  
Два корня: 2.0, -2.0, 1.0 и -1.0
```