|  |  |  |
| --- | --- | --- |
| Защищено:  Нардид А.Н.    "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2024 г. |  | Демонстрация:  Нардид А.Н.    "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2024 г. |

**Отчет по лабораторной работе № 1 по курсу**

**Парадигмы и конструкции языков программирования**

#### Тема работы: "Основные конструкции языка python"

5

(количество листов)

|  |  |
| --- | --- |
| ИСПОЛНИТЕЛЬ: |  |
| студент группы ИУ5Ц-53Б | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | (подпись) |
| Ахмеров Д.И. | "16" сентября 2024 г. |

Москва, МГТУ – 2024

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**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 А, В, С, вычисляет дискриминант и ДЕЙСТВИТЕЛЬНЫЕ корни уравнения (в зависимости от дискриминанта).
3. Коэффициенты А, В, С 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 А, В, С 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.

**Текст программы**

**Biquad\_FP.py**

import math

import sys

def get\_coef(index, prompt):

try:

coef\_str = sys.argv[index]

except:

print(prompt)

coef\_str = input()

coef = float(coef\_str)

return coef

def get\_roots(a, b, c):

result = []

D = b\*b - 4\*a\*c

if D == 0.0:

t = -b / (2.0\*a)

if t >= 0:

root1 = math.sqrt(t)

root2 = -root1

result.append(root1)

result.append(root2)

elif D > 0.0:

sqD = math.sqrt(D)

t1 = (-b + sqD) / (2.0\*a)

t2 = (-b - sqD) / (2.0\*a)

if t1 >= 0:

root1 = math.sqrt(t1)

root2 = -root1

result.append(root1)

result.append(root2)

if t2 >= 0:

root3 = math.sqrt(t2)

root4 = -root3

result.append(root3)

result.append(root4)

return result

def main():

a = get\_coef(1, 'Введите коэффициент А:')

b = get\_coef(2, 'Введите коэффициент B:')

c = get\_coef(3, 'Введите коэффициент C:')

roots = get\_roots(a,b,c)

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 фкорня: {}, {} и {}'.format(roots[0], roots[1], roots[2]))

elif len\_roots == 4:

print("Четыре корня: {}, {}, {} и {}".format(roots[0], roots[1], roots[2], roots[3]))

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Biquad\_OOP.py**

import sys

import math

class SquareRoots:

    def \_\_init\_\_(self):

        self.coef\_A = 0.0

        self.coef\_B = 0.0

        self.coef\_C = 0.0

        self.num\_roots = 0

        self.roots\_list = []

    def get\_coef(self, index, prompt):

        try:

            coef\_str = sys.argv[index]

        except:

            print(prompt)

            coef\_str = input()

        coef = float(coef\_str)

        return coef

    def get\_coefs(self):

        self.coef\_A = self.get\_coef(1, 'Введите коэффициент А:')

        self.coef\_B = self.get\_coef(2, 'Введите коэффициент B:')

        self.coef\_C = self.get\_coef(3, 'Введите коэффициент C:')

    def calculate\_roots(self):

        a = self.coef\_A

        b = self.coef\_B

        c = self.coef\_C

        D = b\*b - 4\*a\*c

        if D == 0.0:

            t = -b / (2.0\*a)

            if t >= 0:

                self.num\_roots = 2

                root1 = math.sqrt(t)

                root2 = -root1

                self.roots\_list.append(root1)

                self.roots\_list.append(root2)

        elif D > 0.0:

            sqD = math.sqrt(D)

            t1 = (-b + sqD) / (2.0\*a)

            t2 = (-b - sqD) / (2.0\*a)

            if t1 >= 0:

                root1 = math.sqrt(t1)

                root2 = -root1

                self.roots\_list.append(root1)

                self.roots\_list.append(root2)

            if t2 >= 0:

                root3 = math.sqrt(t2)

                root4 = -root3

                self.roots\_list.append(root3)

                self.roots\_list.append(root4)

            self.num\_roots = len(self.roots\_list)

    def print\_roots(self):

        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 == 1:

                print('Один корень: {}'.format(self.roots\_list[0]))

            elif self.num\_roots == 2:

                print('Два корня: {} и {}'.format(self.roots\_list[0], self.roots\_list[1]))

            elif self.num\_roots == 3:

                print('Три фкорня: {}, {} и {}'.format(self.roots\_list[0], self.roots\_list[1], self.roots\_list[2]))

            elif self.num\_roots == 4:

                print("Четыре корня: {}, {}, {} и {}".format(self.roots\_list[0], self.roots\_list[1], self.roots\_list[2], self.roots\_list[3]))

def main():

    r = SquareRoots()

    r.get\_coefs()

    r.calculate\_roots()

    r.print\_roots()

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Экранные формы с примерами выполнения программы**

