Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего

образования «Московский государственный технический университет имени

Н.Э. Баумана (национальный исследовательский университет)»

(МГТУ им. Н.Э. Баумана)

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

**Базовые компоненты интернет-технологий**

**«Основные конструкции языка Python»**

|  |  |
| --- | --- |
| Выполнил: | Проверил: |
| студент группы ИУ5-34Б |  |
| Харитонов Андрей |  |
| Подпись и дата:  29.12.21 | Подпись и дата: |

Москва, 2021

**Постановка задачи**

Разработать программу для решения [биквадратного уравнения.](https://ru.wikipedia.org/wiki/%D0%A3%D1%80%D0%B0%D0%B2%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5_%D1%87%D0%B5%D1%82%D0%B2%D1%91%D1%80%D1%82%D0%BE%D0%B9_%D1%81%D1%82%D0%B5%D0%BF%D0%B5%D0%BD%D0%B8#%D0%91%D0%B8%D0%BA%D0%B2%D0%B0%D0%B4%D1%80%D0%B0%D1%82%D0%BD%D0%BE%D0%B5_%D1%83%D1%80%D0%B0%D0%B2%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5)

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](https://github.com/ugapanyuk/BKIT_2021/blob/main/code/lab1_code) ). Если они не заданы, то вводятся с клавиатуры в соответствии с пунктом 2. [Описание работы с параметрами командной строки.](https://realpython.com/python-command-line-arguments/#the-command-line-interface)
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.

**Текст программы.**

Файл “PyLab\_1.py”

from InputModule import InputCoefficients

from SolvingModule import SolveEquation, OutputSolves

def main():

coefficients = InputCoefficients()

solves = SolveEquation(coefficients)

OutputSolves(solves)

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

main()

Файл “CoefficientsStruct.py”

class Coefficients:

'''

This class is the struct of the coefficients of the quadrate equation.

It's look like A \* x^2 + B \* x + C = 0

'''

A = 0

B = 0

C = 0

COEFF\_COUNT = 3

pushedNum = 0

def PushNext(self, coefficient):

if(self.pushedNum == 0):

self.A = float(coefficient)

elif(self.pushedNum == 1):

self.B = float(coefficient)

elif(self.pushedNum == 2):

self.C = float(coefficient)

else:

raise Exception("All coefficients of this struct have pushed")

self.pushedNum += 1

Файл “InputModule.py”

import sys

from CoefficientsStruct import Coefficients

def isFloat(str):

try:

str = float(str)

return True

except:

return False

def \_InputWithStart():

'''This function input coefficients with program start by console'''

coefficients = Coefficients()

for token in sys.argv:

if(isFloat(token)):

coefficients.PushNext(token)

if(coefficients.pushedNum == coefficients.COEFF\_COUNT):

break

return coefficients

def \_InputCoefficient(coefficientName, correctMess = True):

'''This function print title and input coefficient'''

if(not correctMess):

print("Uncorrect coefficint. Pleas, repeat your input.")

print("Input {} coefficient of the equation: ".format(coefficientName))

return input()

def \_InputFromConsole(coefficients):

'''This fucntion input coefficients from console'''

coeffNames = ('A', 'B', 'C')

correct = True

while(coefficients.pushedNum < coefficients.COEFF\_COUNT):

digit = \_InputCoefficient(coeffNames[coefficients.pushedNum], correct)

if(isFloat(digit)):

coefficients.PushNext(digit)

correct = True

else:

correct = False

return coefficients

def InputCoefficients():

'''This function input coefficients with start and from console if it needs'''

coefficients = \_InputWithStart()

coefficients = \_InputFromConsole(coefficients)

return coefficients

Файл “SolvingModule.py”

from math import sqrt

from CoefficientsStruct import Coefficients

class Solve:

'''This struct is the way to abstract solution of equation.'''

value = 0

wasSolved = False

def \_Biquadrate(coefficients, signs):

'''It's the formula of biquadrate equation'''

return signs[0] \* sqrt( (-coefficients.B + signs[1] \* sqrt(coefficients.B\*\*2 - 4 \* coefficients.A \* coefficients.C)) / (2 \* coefficients.A) )

def \_Quadrate(coefficients, sign):

'''Is's the formula of quadrate equation'''

return (-coefficients.B + sign \* sqrt(coefficients.B\*\*2 - 4 \* coefficients.A \* coefficients.C)) / (2 \* coefficients.A)

def \_TryFindRoot(coefficients, signs, formula):

'''This function abstracts the try-block of solution'''

solve = Solve()

try:

solve.value = formula(coefficients, signs)

solve.wasSolved = True

except (ValueError, ArithmeticError):

solve.wasSolved = False

return solve

def SolveBiquadrate(coefficients):

singsVariation = ((1, 1), (-1, 1), (1, -1), (-1, -1))

roots = []

for sings in singsVariation:

roots.append( \_TryFindRoot(coefficients, sings, \_Biquadrate) )

return roots

def SolveQuadrate(coefficients):

roots = [

\_TryFindRoot(coefficients, 1, \_Quadrate),

\_TryFindRoot(coefficients, -1, \_Quadrate)

]

return roots

def SolveEquation(coefficients):

solvesList = []

if(coefficients.A != 0):

solvesList = SolveBiquadrate(coefficients)

else:

coefficients.A = coefficients.B

coefficients.B = 0

solvesList = SolveQuadrate(coefficients)

return solvesList

def OutputSolves(solvesList):

noOneRoot = True

for solve in solvesList:

if(solve.wasSolved):

print("The root of the equation = ", solve.value)

noOneRoot = False

if(noOneRoot):

print("No one correct root")

**Пример выполнения**

