ГУАП  
КАФЕДРА № 41

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| доцент |  |  |  | А.В. Аграновский |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ |
| ЛИНЕЙНЫЕ АЛГОРИТМЫ |
| по курсу: ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | М023 |  |  |  | Д.А. Трегуб |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2022

ПОСТАНОВКА ЗАДАЧИ

Задача №19. Даны коэффициенты квадратного уравнения a, b, c. Рассчитайте сумму и произведение корней по формулам Виета.

Нахождение суммы и произведения корней по теореме Виета осуществляется по формулам для суммы и для произведения. Теорема Виета справедлива, если корни действительны, то есть .

На вход подаются коэффициенты a, b, c квадратного уравнения ax2+bx+x=0.

Конечным результатом программы являются значения суммы и произведения корней.

Предела вводимых значений для коэффициентов b и с нет. Значение a не может быть равно 0. Дискриминант должен быть больше либо равен нулю.

МАТЕМАТИЧЕСКАЯ МОДЕЛЬ

.

Область определения данных:

ОПИСАНИЕ РАЗРАБОТАННОЙ ПРОГРАММЫ

Таблица 1 Описание переменных

|  |  |  |
| --- | --- | --- |
| Имя переменной | Тип переменной | Назначение |
| a | Int/Float | Коэффициент a |
| b | Int/Float | Коэффициент b |
| c | Int/Float | Коэффициент с |
| sumX | Int/Float/Str | Конечный результат (сумма корней) |
| multiply | Int/Float/Str | Конечный результат (произведение корней) |

На вход подаётся 3 числа: коэффициенты “a”, “b”, “c”.

После получения входных данных программа последовательно проводить проверку на: возможность привести полученные значения к типу float, то, что коэффициент “a” не равен нулю, существование корней уравнения. В случае не прохождения проверки возвращается соответствующее сообщение об ошибке.

В случае прохождения в переменную sumX записывается результат выражения (-b/a), а в переменную multiply выражения (c/a). После этого sumX и multiply возвращаются как результат работы программы.

Алгоритм является линейным, число входящих данных всегда константа – сложность O(1).

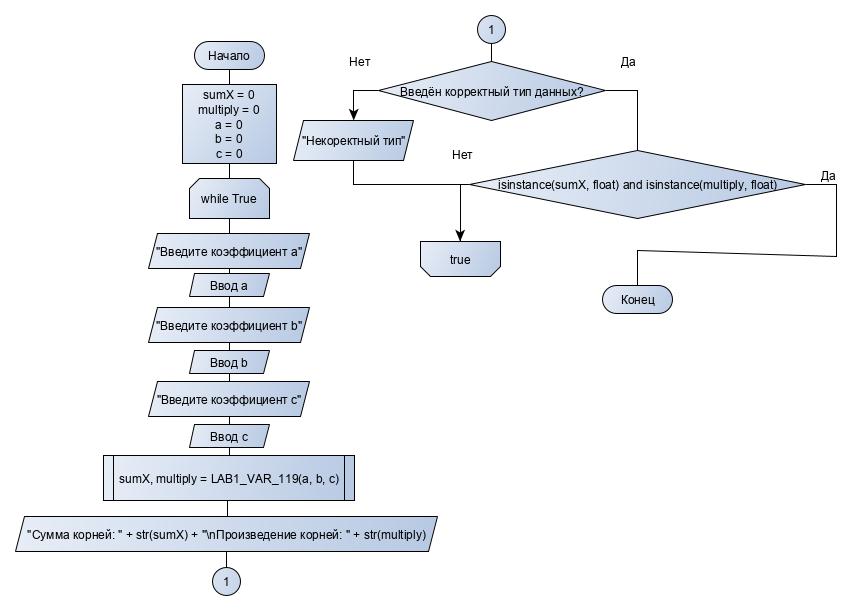


Рисунок 1 – функция main

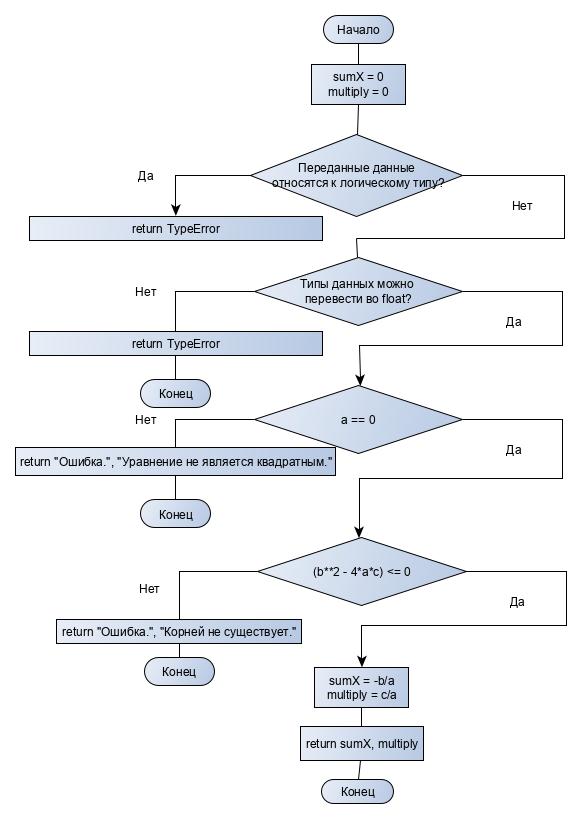


Рисунок 2 - функция LAB1\_VAR\_19

ОПИСАНИЕ ТЕСТОВОГО НАБОРА

Таблица 2 Тестовый набор

|  |  |  |  |
| --- | --- | --- | --- |
| Описание тестового случая | Входные данные | Ожидаемый результат | Результат теста |
| Переданы целые числа | a = 1, b = 4, c = 3 | -4, 3 | OK |
| Переданы дробные числа | a = 1.0, b = 4.5, c = -0.3 | -4.5, -0.3 | OK |
| Введен дробный тип данных | a = “a”, b = 2, c = 3 | TypeError | OK |
| Введен логический тип данных | a = 1, b = False, c = 3 | TypeError | OK |
| Уравнение не является квадратным. | a = 0, b = 2, c = 3, | "Ошибка. Уравнение не является квадратным." | OK |
| Корней не существует | a = 1, b = 0, c = 4, | "Ошибка. Корней ней существует” | OK |

В случае непрохождения теста выводится результат работы программы после сообщения о непрохождении.

ПРИМЕР РАБОТЫ ПРОГРАММЫ

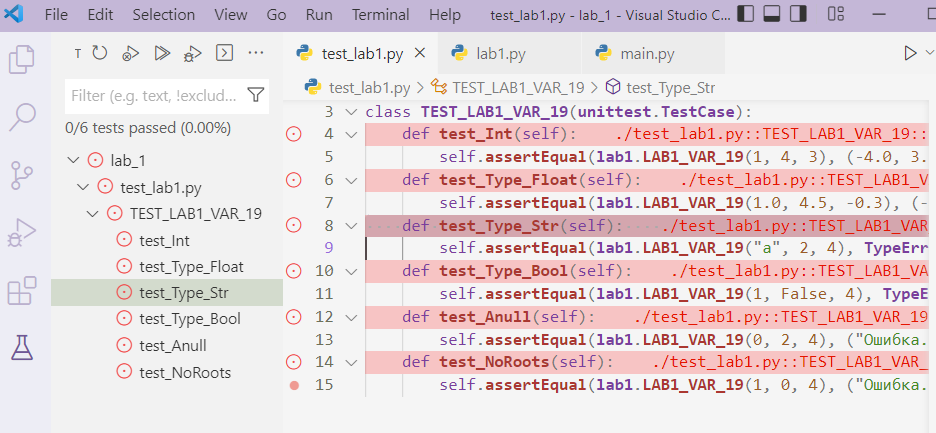


Рисунок 3 - Результат работы тестов до написания программы



Рисунок 4 - Результат работы тестов после написания программы

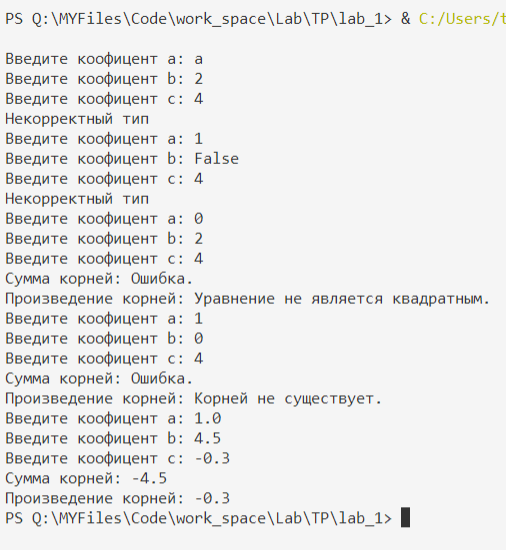


Рисунок 5 - Результат работы программы

ВЫВОД

Были получены навыки работы с python. Способы решения задач путем программирования алгоритмов их решения. Были изучены и применены на практике способы тестирования программ на корректность работы. На практике были применены способы оформления кода для повышения его читаемости: отступы, комментарии, пустые строки и т.д..

ПРИЛОЖЕНИЯ

Приложение 1

Код программы

Код тестов:

import **unittest**

import **lab1**

class **TEST\_LAB1\_VAR\_19**(**unittest**.**TestCase**):

    def **test\_Int**(self):

        self.**assertEqual**(**lab1**.**LAB1\_VAR\_19**(1, 4, 3), (-4.0, 3.0) )

    def **test\_Type\_Float**(self):

        self.**assertEqual**(**lab1**.**LAB1\_VAR\_19**(1.0, 4.5, -0.3), (-4.5, -0.3))

    def **test\_Type\_Str**(self):

        self.**assertEqual**(**lab1**.**LAB1\_VAR\_19**("a", 2, 4), **TypeError**)

    def **test\_Type\_Bool**(self):

        self.**assertEqual**(**lab1**.**LAB1\_VAR\_19**(1, False, 4), **TypeError**)

    def **test\_Anull**(self):

        self.**assertEqual**(**lab1**.**LAB1\_VAR\_19**(0, 2, 4), ("Ошибка.", "Уравнение не является квадратным."))

    def **test\_NoRoots**(self):

        self.**assertEqual**(**lab1**.**LAB1\_VAR\_19**(1, 0, 4), ("Ошибка.", "Корней не существует."))

Код функции LAB1\_VAR\_19:

def **LAB1\_VAR\_19**(a, b, c):

    sumX = 0

    multiply = 0

*# Проверка на возможность обработать данные как числа:*

*# Проверка, что тип переменных не логический*

    if **type**(a) == **bool** or **type**(b) == **bool** or **type**(c) == **bool**:

        return **TypeError**

*# Преведение введеных значений к типу float*

    try:

        a = **float**(a)

        b = **float**(b)

        c = **float**(c)

    except:

        return **TypeError**

*#Проверка на неравенства коэффициента "a" нулю*

    if (a == 0):

        return "Ошибка.", "Уравнение не является квадратным."

*#Проверка на существование корней*

    if (b\*\*2 - 4\*a\*c) <= 0:

        return "Ошибка.", "Корней не существует."

*#Вычисление суммы и произведения корней*

    sumX = -b/a

    multiply = c/a

    return sumX, multiply

Код функции main:

import **lab1**

*#иницилизация переменных*

a = 0

b = 0

c = 0

sumX = 0

multiply = 0

*#начало бесконечного цикла*

while True:

*#обработка исключения для корректного ввода*

    try:

        a = **input**("Введите коофицент a: ")*#Ввод коофицента a*

        b = **input**("Введите коофицент b: ") *#Ввод коофицента b*

        c = **input**("Введите коофицент c: ") *#Ввод коофицента c*

        sumX, multiply = **lab1**.**LAB1\_VAR\_19**(a, b, c) *#вызов функции*

**print**("Сумма корней: " + **str**(sumX) + "\nПроизведение корней: " + **str**(multiply)) *#Вывод полученных значений*

    except **TypeError**:

**print**("Некорректный тип")

*#Если полученные коректные значения - прервать цикл*

    if **isinstance**(sumX, **float**) and **isinstance**(multiply, **float**):

        break

*#Конец цикла*