Московский государственный технический

университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Отчет по лабораторной работе №1

«Основные конструкции языка Python»

по дисциплине

«Парадигмы и конструкции языков программирования»

Выполнил:

студент группы ИУ5-33Б:

Буйдина К.А.

Подпись и дата:

Проверил:

преподаватель каф. ИУ5

Гапанюк Ю.Е.

Подпись и дата:

Москва, 2023 г.

Описание задания:

Разработать программу для решения биквадратного уравнения.

- Программа должна быть разработана в виде консольного приложения на языке Python.
- Программа осуществляет ввод с клавиатуры коэффициентов A, B, C, вычисляет дискриминант и ДЕЙСТВИТЕЛЬНЫЕ корни уравнения (в зависимости от дискриминанта).
- 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.
- 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.
- Дополнительное задание 1 (*). Разработайте две программы на языке Python - одну с применением процедурной парадигмы, а другую с применением объектно-ориентированной парадигмы.
- Дополнительное задание 2 (*). Разработайте две программы - одну на языке Python, а другую на любом другом языке программирования (кроме C++).

Текст программы:

1. **go.mod**

```
module lab1_1
```

2. **main.go**

```
package main

import (
    "bufio"   // для буферизованного ввода/вывода
    "fmt"     // для форматированного ввода/вывода
    "math"    // для математических функций
    "os"      // для взаимодействия с операционной системой
    "strconv" // для преобразования строк
    "strings" // для работы со строками
)

func main() {
    coef_list := get_roots() // Получаем коэффициенты
    all_process(coef_list)   // Обрабатываем коэффициенты
}

func all_process(coef_list []float64) {
    print_ans(calculation(coef_list)) // Вычисляем корни и выводим их
}

func print_ans(root_list []float64) {
    // Если корней нет, выводим сообщение
```

```
    if len(root_list) == 0 {
        fmt.Println("Нет корней, дискриминант меньше нуля :(")
    } else {
        fmt.Println("Корни:")
        // Выводим каждый корень
        for _, e := range root_list {
            fmt.Println(e, " ")
        }
    }
}

func calculation(coef_list []float64) []float64 {
    A := coef_list[0]
    B := coef_list[1]
    C := coef_list[2]
    D := math.Pow(B, 2) - 4*A*C // Вычисляем дискриминант
    root_list := make([]float64, 0)
    all_roots := make([]float64, 0)
    // Вычисляем корни, если дискриминант неотрицательный
    if D >= 0.0 {
        root_list = append(root_list, (-B+math.Sqrt(D))/(2.0*A))
        root_list = append(root_list, (-B-math.Sqrt(D))/(2.0*A))
    }
    // Добавляем корни в список
    for _, r := range root_list {
        if r >= 0 {
            all_roots = append(all_roots, math.Sqrt(r))
            all_roots = append(all_roots, -math.Sqrt(r))
        }
    }
    // Выводим коэффициенты и дискриминант
    for _, e := range coef_list {
        fmt.Println(e, " ", D)
    }
    return all_roots // Возвращаем список корней
}

func get_roots() []float64 {
    coef_list := make([]float64, 3) // Создаем список для коэффициентов
    for i := 0; i < 3; i++ {
        coef_list[i] = check_root(i + 1) // Проверяем каждый коэффициент
    }
    return coef_list
}

func check_root(ind int) float64 {
    reader := bufio.NewReader(os.Stdin)
    for {
        fmt.Printf("Введите коэффициент %d: ", ind)
        input, _ := reader.ReadString('\n')
        coef, err := strconv.ParseFloat(strings.TrimSpace(input), 64)
        // Если введенное значение не является числом, выводим сообщение
об ошибке
        if err != nil {
            fmt.Println("Ошибка. Попробуйте еще раз")
            continue
        }
        // Если первый коэффициент равен нулю, выводим сообщение об
ошибке
        if coef == 0.0 && ind == 1 {
            fmt.Println("Коэффициент 1 равен 0. Так не пойдет")
            continue
        }
        return coef
```

```
            }
        }
```

Экранные формы с примерами выполнения программы:

1. Пример с отсутствием корней

```
Введите коэффициент 1: 4
Введите коэффициент 2: 0
Введите коэффициент 3: 10
4   -160
0   -160
10   -160
Нет корней, дискриминант меньше нуля :(

Process finished with the exit code 0
```

2. Пример, при котором первый коэффициент вводится равным 0 (при дальнейшей работе программы это привело бы к делению на 0)

```
Введите коэффициент 1: 0
Коэффициент 1 равен 0. Так не пойдет
Введите коэффициент 1: 1
Введите коэффициент 2: 1
Введите коэффициент 3: 9
1   -35
1   -35
9   -35
Нет корней, дискриминант меньше нуля :(
```

3. Пример ввода иных символов

```
Введите коэффициент 1: 1
Введите коэффициент 2: 0
Введите коэффициент 3: -4
1   16
0   16
-4   16
Корни:
1.4142135623730951
-1.4142135623730951
```

Также я решила попробовать применить в Go ООП подход:

Текст программы:

main.go

```go
package main

import (
    "bufio"    // для буферизованного ввода/вывода
    "fmt"      // для форматированного ввода/вывода
    "math"     // для математических функций
    "os"       // для взаимодействия с операционной системой
    "strconv" // для преобразования строк
```

```
    "strings" // для работы со строками
)

type Coefficients struct {
    A, B, C float64
}

// Метод для получения коэффициентов
func (c *Coefficients) get_roots() {
    for i := 0; i < 3; i++ {
       switch i {
       case 0:
          c.A = check_coefficient(i + 1)
       case 1:
          c.B = check_coefficient(i + 1)
       case 2:
          c.C = check_coefficient(i + 1)
       }
    }
}

// Метод для вычисления корней уравнения
func (c *Coefficients) calculate() []float64 {
    D := math.Pow(c.B, 2) - 4*c.A*c.C
    root_list := make([]float64, 0)
    all_roots := make([]float64, 0)
    if D >= 0.0 {
       root_list = append(root_list, (-c.B+math.Sqrt(D))/(2.0*c.A))
       root_list = append(root_list, (-c.B-math.Sqrt(D))/(2.0*c.A))
    }
    for _, r := range root_list {
       if r >= 0 {
          all_roots = append(all_roots, math.Sqrt(r))
          all_roots = append(all_roots, -math.Sqrt(r))
       }
    }
    return all_roots
}

func main() {
    var coef Coefficients
    coef.get_roots()                // Получаем коэффициенты
    print_answer(coef.calculate()) // Выводим корни уравнения
}

// Функция для вывода корней уравнения
func print_answer(root_list []float64) {
    if len(root_list) == 0 {
       fmt.Println("Нет корней, дискриминант меньше нуля :(")
    } else {
       fmt.Println("Корни:")
       for _, e := range root_list {
          fmt.Println(e, " ")
       }
    }
}

// Функция для проверки корректности введенных коэффициентов
func check_coefficient(ind int) float64 {
    reader := bufio.NewReader(os.Stdin)
    for {
       fmt.Printf("Введите коэффициент %d: ", ind)
       input, _ := reader.ReadString('\n')
       coef, err := strconv.ParseFloat(strings.TrimSpace(input), 64)
```

```
        if err != nil {
            fmt.Println("Ошибка. Попробуйте еще раз")
            continue
        }
        if coef == 0.0 && ind == 1 {
            fmt.Println("Коэффициент 1 равен 0. Так не пойдет")
            continue
        }
        return coef
    }
}
```

Экранные формы с примерами выполнения программы: