МИНИСТРЕРСТВО ОБРАЗОВАНИЯ И НАУКИ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОУ НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ИМ. Р.Е. АЛЕКСЕЕВА

ИНСТИТУТ ЭКОНОМИКИ И УПРАВЛЕНИЯ

КАФЕДРА " Прикладная математика и информатика"

Дисциплина "Структуры данных"

Отчет

по лабораторной работе № 3

Выполнил: Иванов Михаил Егорович

студент группы: 21 САИ

Проверил: Моисеев Антон Евгеньевич

Нижний Новгород

2023

**Цель работы:** реализовать решение квадратного уравнения в виде подпрограммы.

**Практическая часть.**

Квадратное уравнение имеет вид ax^2 + bx + c = 0, где a, b и c - заданные коэффициенты, а x - неизвестная переменная. Для решения квадратного уравнения необходимо найти его корни, то есть такие значения x, при которых уравнение обращается в ноль.

Для нахождения корней квадратного уравнения используется формула дискриминанта: D = b^2 - 4ac. В зависимости от знака дискриминанта, уравнение может иметь два действительных корня, один действительный корень или два мнимых корня. Формулы для нахождения корней в каждом случае следующие:

* Если D > 0, то x1 = (-b + √D) / (2a) и x2 = (-b - √D) / (2a).
* Если D = 0, то x1 = x2 = -b / (2a).
* Если D < 0, то x1 = (-b + i√|D|) / (2a) и x2 = (-b - i√|D|) / (2a), где i - мнимая единица.

**Код:**

package main  
  
import (  
 "fmt"  
 "math"  
)  
  
// Функция для решения квадратного уравнения вида ax^2 + bx + c = 0  
// Возвращает количество действительных или мнимых корней и сами корни  
// Если корней нет, возвращает ноль и нулевые значения для корней  
func solveQuadratic(a float64, b float64, c float64) (int, float64, float64, float64, float64) {  
 // Вычисляем дискриминант  
 D := discriminant(a, b, c)  
 // В зависимости от знака дискриминанта выбираем соответствующий случай  
 if D > 0 {  
 // Два действительных корня  
 return twoRealRoots(a, b, D)  
 }  
 if D == 0 {  
 // Один действительный корень  
 return oneRealRoot(a, b)  
 }  
 if D < 0 {  
 // Два мнимых корня  
 return twoComplexRoots(a, b, D)  
 }  
 return 0, 0, 0, 0, 0  
}  
  
// Функция для вычисления дискриминанта  
func discriminant(a float64, b float64, c float64) float64 {  
 return b\*b - 4\*a\*c  
}  
  
// Функция для нахождения одного действительного корня  
func oneRealRoot(a float64, b float64) (int, float64, float64, float64, float64) {  
 return 1,  
 (-b) / (2 \* a), // Единственный корень  
 0, 0, 0  
}  
  
// Функция для нахождения двух действительных корней  
func twoRealRoots(a float64, b float64, D float64) (int, float64, float64, float64, float64) {  
 return 2,  
 (-b + math.Sqrt(D)) / (2 \* a), // Первый корень  
 (-b - math.Sqrt(D)) / (2 \* a), // Второй корень  
 0, 0  
}  
  
// Функция для нахождения двух мнимых корней  
func twoComplexRoots(a float64, b float64, D float64) (int, float64, float64, float64, float64) {  
 return 4,  
 -b / (2 \* a), // Действительная часть первого корня  
 math.Sqrt(math.Abs(D)) / (2 \* a), // Мнимая часть первого корня  
 -b / (2 \* a), // Действительная часть второго корня  
 -math.Sqrt(math.Abs(D)) / (2 \* a) // Мнимая часть второго корня  
}  
  
// Функция для печати решения в зависимости от количества корней  
func printSolution(i int, x1 float64, x2 float64, x3 float64, x4 float64) {  
 if i == 1 {  
 fmt.Println("x1 =", x1)  
 }  
 if i == 2 {  
 fmt.Println("x1 =", x1, "; x2 =", x2)  
 }  
 if i == 4 {  
 fmt.Println("x1 =", x1, "+", x2, "\* i", "; x2 =", x3, "+", x4, "\* i")  
 }  
}  
  
func main() {  
  
 printSolution(solveQuadratic(1, 3, -34))  
 printSolution(solveQuadratic(3, 6, 21))  
  
}

### Вывод: