**Московский государственный технический университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе №1

Выполнил:
студент группы ИУ5-33Б
Нестерова Екатерина

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.

Москва, 2024 г.

**Постановка задачи:**

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 A, B, C, вычисляет дискриминант и ДЕЙСТВИТЕЛЬНЫЕ корни уравнения (в зависимости от дискриминанта).
3. Коэффициенты A, B, C могут быть заданы в виде параметров командной строки. Если они не заданы, то вводятся с клавиатуры в соответствии с пунктом 2. Описание работы с параметрами командной строки.
4. 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.
5. Дополнительное задание 1 (*). Разработайте две программы на языке Python - одну с применением процедурной парадигмы, а другую с применением объектно-ориентированной парадигмы.
6. Дополнительное задание 2 (*). Разработайте две программы - одну на языке Python, а другую на любом другом языке программирования (кроме C++).

**Код (на Python) :**

```
import sys
import math
class Solver:
    def __init__(self,A=None,B=None,C=None):
        self.A=A
        self.B=B
        self.C=C
    def get_coeff(self,str):
        while True:
            try:
                value=float(input(str))
```

```
            return value
        except ValueError:
            print("Error")
def inp_coeff(self):
    if len(sys.argv) == 4:
        try:
            self.A = float(sys.argv[1])
            self.B = float(sys.argv[2])
            self.C = float(sys.argv[3])
        except ValueError:
            print("Error: Invalid input. Please provide valid numbers.")
            self.A = self.get_coeff("Enter A: ")
            self.B = self.get_coeff("Enter B: ")
            self.C = self.get_coeff("Enter C: ")
    elif len(sys.argv) == 2:
        print("Error: Insufficient coefficients provided.")
        self.A = float(sys.argv[1])
        self.B = self.get_coeff("Enter B: ")
        self.C = self.get_coeff("Enter C: ")
    elif len(sys.argv) == 3:
        print("Error: Insufficient coefficients provided.")
        self.A = float(sys.argv[1])
        self.B = float(sys.argv[2])
        self.C = self.get_coeff("Enter C: ")
    else:
        print("Error: No coefficients provided.")
        self.A = self.get_coeff("Enter A: ")
        self.B = self.get_coeff("Enter B: ")
        self.C = self.get_coeff("Enter C: ")
def Discr(self):
   return self.B**2-4*self.A*self.C
def roots(self):
```

```
        D=self.Discr()
        print(f"D={D}")
        if D > 0:
            x1 = (-self.B + math.sqrt(D)) / (2 * self.A)
            x2 = (-self.B - math.sqrt(D)) / (2 * self.A)
            roots = {x1, x2}
            print("Roots:", roots)
        elif D == 0:
            x = -self.B / (2 * self.A)
            roots = {x}
            print("Roots:", roots)
        else:
            print("No real roots detected.")
def main():
    solver=Solver()
    solver.inp_coeff()
    solver.roots()
if __name__=="__main__":
    main()
```

**Код на golang:**

```
package main

import (
    "fmt"
    "os"
    "strconv"
)

func getCoefficient(prompt string) float64 {
    for {
  fmt.Print(prompt)
  var input string
```

```
  fmt.Scanln(&input)

  value, err := strconv.ParseFloat(input, 64)
  if err != nil {
   fmt.Println("Некорректное значение. Пожалуйста, введите действительное число.")
   continue
  }
  return value
 }
}

func main() {
    // Проверяем, переданы ли коэффициенты через командную строку
    if len(os.Args) == 4 {
        // Три коэффициента A, B, C
        A, err := strconv.ParseFloat(os.Args[1], 64)
        if err != nil {
            fmt.Println("Некорректные коэффициенты в командной строке. Будем запрашивать значения.")
            A = getCoefficient("Введите коэффициент A: ")
        }
        B, err := strconv.ParseFloat(os.Args[2], 64)
        if err != nil {
            fmt.Println("Некорректные коэффициенты в командной строке. Будем запрашивать значения.")
            B = getCoefficient("Введите коэффициент B: ")
        }
        C, err := strconv.ParseFloat(os.Args[3], 64)
        if err != nil {
            fmt.Println("Некорректные коэффициенты в командной строке. Будем запрашивать значения.")
            C = getCoefficient("Введите коэффициент C: ")
        }
```

```
    } else if len(os.Args) == 2 {
        // Одно значение, берем его за A, запрашиваем B и C
        A, err := strconv.ParseFloat(os.Args[1], 64)
        if err != nil {
            fmt.Println("Некорректное значение для A. Будем запрашивать значение снова.")
            A = getCoefficient("Введите коэффициент A: ")
        }
        B = getCoefficient("Введите коэффициент B: ")
        C = getCoefficient("Введите коэффициент C: ")
    } else {
        // Менее одного аргумента
        fmt.Println("Недостаточно аргументов в командной строке. Вводим коэффициенты вручную.")
        A = getCoefficient("Введите коэффициент A: ")
        B = getCoefficient("Введите коэффициент B: ")
        C = getCoefficient("Введите коэффициент C: ")
    }


 // Вычисление дискриминанта для биквадратного уравнения
 D := B*B - 4*A*C
 fmt.Printf("Дискриминант D = %.2fn", D)

 // Анализ дискриминанта и поиск корней
 if D > 0 {
  // Два различных вещественных корня
  x1_sq := (-B + math.Sqrt(D)) / (2 * A)
  x2_sq := (-B - math.Sqrt(D)) / (2 * A)
  roots := make(map[float64]bool)

  // Извлечение корней из x^2
  if x1_sq >= 0 {
```

```
 roots[math.Sqrt(x1_sq)] = true
 roots[-math.Sqrt(x1_sq)] = true
}
if x2_sq >= 0 {
 roots[math.Sqrt(x2_sq)] = true
 roots[-math.Sqrt(x2_sq)] = true
}

fmt.Print("Корни уравнения: ")
first := true
for root := range roots {
 if !first {
  fmt.Print(", ")
 }
 fmt.Printf("%.2f", root)
 first = false
}
fmt.Println()

} else if D == 0 {
// Один двойной корень
x_sq := -B / (2 * A)
roots := make(map[float64]bool)
if x_sq >= 0 {
 roots[math.Sqrt(x_sq)] = true
 roots[-math.Sqrt(x_sq)] = true
}

fmt.Print("Корни уравнения: ")
first := true
for root := range roots {
 if !first {
```

```
    fmt.Print(", ")
   }
   fmt.Printf("%.2f", root)
   first = false
  }
  fmt.Println()

 } else {
  // Нет действительных корней
  fmt.Println("Уравнение не имеет действительных корней.")
 }
}
```

**Примеры работы кода:**

```
C:\Users\vilen>py source\repos\lab1_vers2\lab1_vers2\lab1_vers2.py
Error: No coefficients provided.
Enter A: 1
Enter B: 2
Enter C: 5
D=-16.0
No real roots detected.
```

```
C:\Users\vilen>py source\repos\lab1_vers2\lab1_vers2\lab1_vers2.py 3 4 5
D=-44.0
No real roots detected.

C:\Users\vilen>
```

```
Error: No coefficients provided.
Enter A: 1
Enter B: 4
Enter C: -4
D=32.0
Roots: {0.8284271247461903, -4.82842712474619}
```