**ЛАБОРАТОРНА РОБОТА № 7**

**Веб-програмування. Rest API**

***Мета:*** засвоїти принцип проектування веб-серверу. Отримати практичні знання по роботі з пакетом net/http та get- та post-методами.

**Хід роботи**

Написати програму рішення квадратного рівняння. Результат відобразити в браузері. Рішення надати в двох варіантах: отримання вхідних даних з використанням *get*-запиту та *post*-запиту;

type Solution string

var HttpSolution1 Solution = "Квадратне рівняння"

func QuadraticEquation(a, b, c float64) []float64 {

    var d, x1, x2 float64

    results := []float64{}

    d = b\*b - 4\*a\*c

*if* d > 0 {

        x1 = ((-b) + math.Sqrt(d)) / (2 \* a)

        x2 = ((-b) - math.Sqrt(d)) / (2 \* a)

        fmt.Println("x1 ->", x1, "\nx2 ->", x2)

        results = append(results, x1, x2)

    }

*if* d == 0 {

        x1 = -(b / (2 \* a))

        fmt.Println("x1 ->", x1)

        results = append(results, x1)

    }

*if* d < 0 {

        fmt.Println("D < 0, рівняння немає дійсних рішень")

    }

*return* results

}

func (s Solution) ServeHTTP(w http.ResponseWriter, r \*http.Request) {

    fmt.Fprint(w, pageHeader, pageBody, form) *// Формируем страницу в браузере*

*if* r.Method == "POST" {                   *// Обрабатываем входные данные*

        err := r.ParseForm() *// Парсим форму*

        post := r.PostForm

*if* err != nil {

            fmt.Fprintf(w, anError, err)

*return*

        }

        pa := post.Get("a")

        pb := post.Get("b")

        pc := post.Get("c")

        a, \_ := strconv.ParseFloat(pa, 64)

        b, \_ := strconv.ParseFloat(pb, 64)

        c, \_ := strconv.ParseFloat(pc, 64)

        fmt.Fprintf(w, "a = %v  b = %v  c = %v", a, b, c) *// Выводим сообщение в браузер*

        fmt.Fprintf(w, "\n\t<br /><br />")

        res := QuadraticEquation(a, b, c)

*if* len(res) == 2 {

            fmt.Fprintf(w, "<h3>x1 = %v, x2 = %v</h3>", res[0], res[1])

        }

*if* len(res) == 1 {

            fmt.Fprintf(w, "<h3>x = %v</h3>", res[0])

        }

*if* len(res) == 0 {

            fmt.Fprintf(w, "<h3>D<0, рівняння немає дійсних рішень</h3>")

        }

    }

*if* r.Method == "GET" {

        va := r.FormValue("a")

        vb := r.FormValue("b")

        vc := r.FormValue("c")

        a, \_ := strconv.ParseFloat(va, 64)

        b, \_ := strconv.ParseFloat(vb, 64)

        c, \_ := strconv.ParseFloat(vc, 64)

        fmt.Fprintf(w, "a = %v  b = %v  c = %v", a, b, c) *// Выводим сообщение в браузер*

        fmt.Fprintf(w, "\n\t<br /><br />")

        res := QuadraticEquation(a, b, c)

*if* len(res) == 2 {

            fmt.Fprintf(w, "<h3>x1 = %v, x2 = %v</h3>", res[0], res[1])

        }

*if* len(res) == 1 {

            fmt.Fprintf(w, "<h3>x = %v</h3>", res[0])

        }

*if* len(res) == 0 {

            fmt.Fprintf(w, "<h3>D<0, рівняння немає дійсних рішень</h3>")

        }

    }

}

func main() {

*// Запускаем локальный сервер*

    http.ListenAndServe("localhost:80", HttpSolution1)

}

Get запити



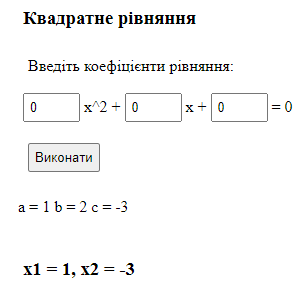


Рис.1. Результат



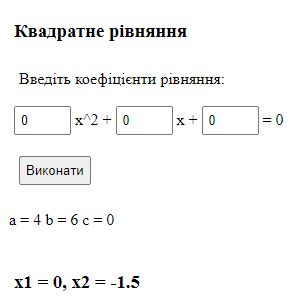


Рис.2. Результат

Post-запити

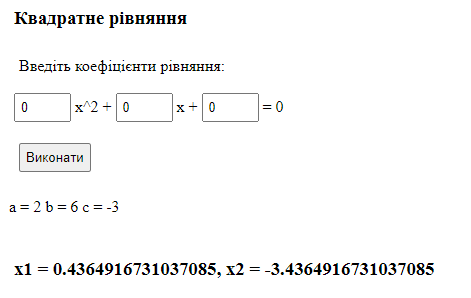
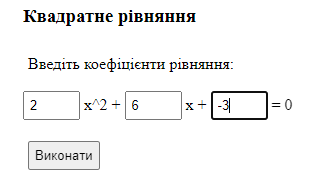


Рис.3. Результат

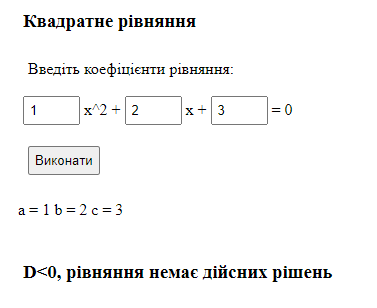


Рис.4. Результат

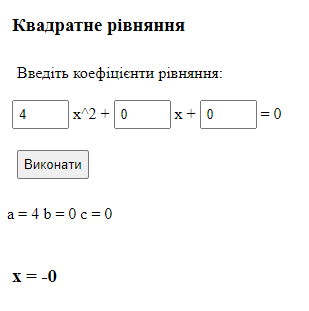
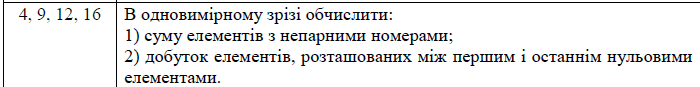


Рис.5.Результат

Написати програму, яка заповнює одновимірний зріз дійсними числами. Виконати розрахунки у відповідності з варіантом (таблиця 7.1). Рішення надати в двох варіантах: отримання вхідних даних з використанням *get*-запиту та *post*-запиту.



package main

import (

    "fmt"

    "math"

    "math/rand"

    "net/http"

    "strconv"

)

const (

    pageHeader = `<!DOCTYPE HTML>

    <html>

    <head>

    <title>Task 2</title>

    <style>

    \*{

        padding: 5px;

    }

    .error{

    color:#FF0000;

    }

    </style>

    </head>`

    pageBody = `<body>

    <h3>Завдання 2</h3>`

    form = `<form action="/" method="POST">

    <label>Введіть кількість чисел зрізу:</label><br /><br />

    <input type="text" name="num" size="2" value="0"><br /><br />

    <label>Введіть максимальне та мінімальне значення:</label><br /><br />

     min -> <input type="text" name="min" size="2" value="0">

     max -> <input type="text" name="max" size="2" value="0">

    <p><input type="submit" value="Виконати"></p>

    </form>`

    anError = `<p class="error">%s</p>`

)

type Task string

var HttpSolution Task = "Завдання 2"

func roundFloat(val float64, precision uint) float64 {

    ratio := math.Pow(10, float64(precision))

*return* math.Round(val\*ratio) / ratio

}

func SearchDobBetweenNulls(arr []float64) float64 {

    var firstNull int

    var lastNull int

*for* i, value := *range* arr {

*if* value == 0 {

            firstNull = i

*break*

        }

    }

    lenArr := len(arr)

*//reverse*

*for* i, j := 0, len(arr)-1; i < j; i, j = i+1, j-1 {

        arr[i], arr[j] = arr[j], arr[i]

    }

*for* i, value := *range* arr {

*if* value == 0 {

            lastNull = i

*break*

        }

    }

*//reverse*

*for* i, j := 0, len(arr)-1; i < j; i, j = i+1, j-1 {

        arr[i], arr[j] = arr[j], arr[i]

    }

    lastNull = lastNull + 1 - lenArr

    fmt.Println("firstNull ->", firstNull)

    fmt.Println("lastNull ->", math.Abs(float64(lastNull)))

    var dob float64 = 1

*for* i := firstNull + 1; i < int(math.Abs(float64(lastNull))); i++ {

        fmt.Println("i ->", i, "value ->", arr[i])

        dob \*= arr[i]

    }

*return* dob

}

func (s Task) ServeHTTP(w http.ResponseWriter, r \*http.Request) {

    fmt.Fprint(w, pageHeader, pageBody, form) *// Формируем страницу в браузере*

*if* r.Method == "POST" {                   *// Обрабатываем входные данные*

        err := r.ParseForm() *// Парсим форму*

        post := r.PostForm

*if* err != nil {

            fmt.Fprintf(w, anError, err)

*return*

        }

        pnum := post.Get("num")

        pmin := post.Get("min")

        pmax := post.Get("max")

        num, \_ := strconv.ParseInt(pnum, 10, 32)

        min, \_ := strconv.ParseFloat(pmin, 64)

        max, \_ := strconv.ParseFloat(pmax, 64)

        fmt.Fprintf(w, "Кількість чисел = %v,   min = %v,   max = %v", num, min, max) *// Выводим сообщение в браузер*

        fmt.Fprintf(w, "\n\t<br /><br />")

        arrNumbers := []float64{}

*for* i := 0; i < int(num); i++ {

            arrNumbers = append(arrNumbers, roundFloat((min+rand.Float64()\*(max-min)), 2))

        }

*//fmt.Println(roundFloat((min + rand.Float64()\*(max-min)), 2))*

        fmt.Println("Len ->", len(arrNumbers), "arr ->", arrNumbers)

        fmt.Fprintln(w, arrNumbers)

        var sum float64

*for* i := 0; i < len(arrNumbers); i += 2 {

            sum += arrNumbers[i]

        }

        fmt.Fprintf(w, "<h3>Сумма елементів з непарними номерами = %v</h3>", sum)

*// ----------------------------------*

        checkNulls := 0

*for* i := 0; i < len(arrNumbers); i++ {

*if* arrNumbers[i] == 0 {

                checkNulls++

            }

        }

*if* checkNulls == 0 {

            arr := []float64{1, 2, 3.2, 0, 2, -2, 1, 4.4, 0, 24, 2.2}

            fmt.Fprintln(w, "<h4>Нулів не знайдено, тому для демонстрації завдання візьмемо зріз з нулями</h4></br>", arr)

            dob := SearchDobBetweenNulls(arr)

            fmt.Println("arr ->", arr)

            fmt.Println("dob ->", dob)

            fmt.Fprintf(w, "<h3>Добуток = %v</h3>", dob)

        }

*if* checkNulls > 0 {

            dob := SearchDobBetweenNulls(arrNumbers)

            fmt.Println("dob ->", dob)

            fmt.Fprintf(w, "<h3>Добуток = %v</h3>", dob)

        }

    }

*if* r.Method == "GET" {

        vnum := r.FormValue("num")

        vmin := r.FormValue("min")

        vmax := r.FormValue("max")

        num, \_ := strconv.ParseFloat(vnum, 64)

        min, \_ := strconv.ParseFloat(vmin, 64)

        max, \_ := strconv.ParseFloat(vmax, 64)

        fmt.Fprintf(w, "Кількість чисел = %v,   min = %v,   max = %v", num, min, max) *// Выводим сообщение в браузер*

        fmt.Fprintf(w, "\n\t<br /><br />")

        arrNumbers := []float64{}

*for* i := 0; i < int(num); i++ {

            arrNumbers = append(arrNumbers, roundFloat((min+rand.Float64()\*(max-min)), 2))

        }

*//fmt.Println(roundFloat((min + rand.Float64()\*(max-min)), 2))*

        fmt.Println("Len ->", len(arrNumbers), "arr ->", arrNumbers)

        fmt.Fprintln(w, arrNumbers)

        var sum float64

*for* i := 0; i < len(arrNumbers); i += 2 {

            sum += arrNumbers[i]

        }

        fmt.Fprintf(w, "<h3>Сумма елементів з непарними номерами = %v</h3>", sum)

*// ----------------------------------*

        checkNulls := 0

*for* i := 0; i < len(arrNumbers); i++ {

*if* arrNumbers[i] == 0 {

                checkNulls++

            }

        }

*if* checkNulls == 0 {

            arr := []float64{1, 2, 3.2, 0, 2, -2, 1, 4.4, 0, 24, 2.2}

            fmt.Fprintln(w, "<h4>Нулів не знайдено, тому для демонстрації завдання візьмемо зріз з нулями</h4></br>", arr)

            dob := SearchDobBetweenNulls(arr)

            fmt.Println("arr ->", arr)

            fmt.Println("dob ->", dob)

            fmt.Fprintf(w, "<h3>Добуток = %v</h3>", dob)

        }

*if* checkNulls > 0 {

            dob := SearchDobBetweenNulls(arrNumbers)

            fmt.Println("dob ->", dob)

            fmt.Fprintf(w, "<h3>Добуток = %v</h3>", dob)

        }

    }

}

func main() {

*// Запускаем локальный сервер*

    http.ListenAndServe("localhost:80", HttpSolution)

}

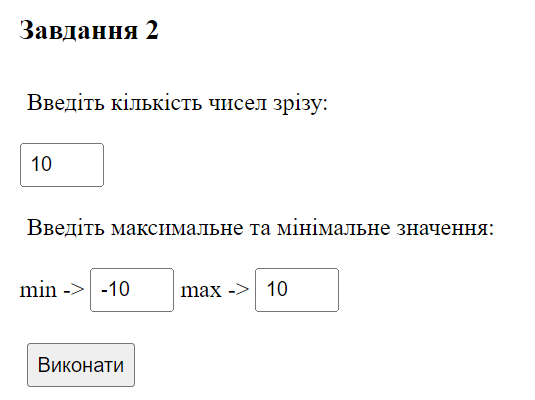


Рис.6. Результат

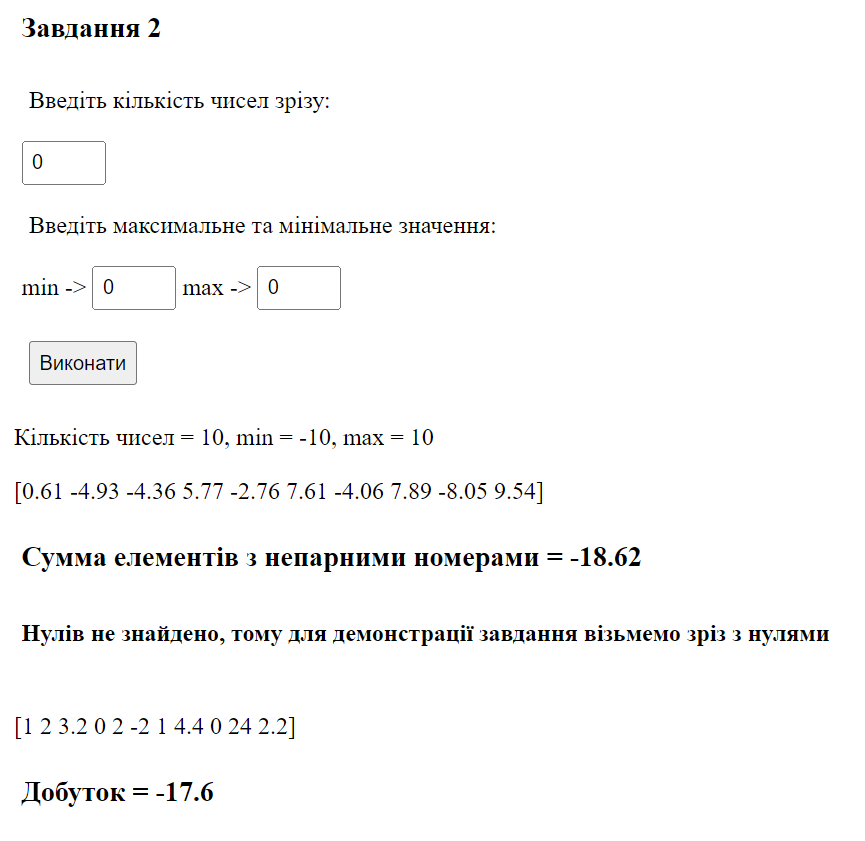


Рис.7. Результат

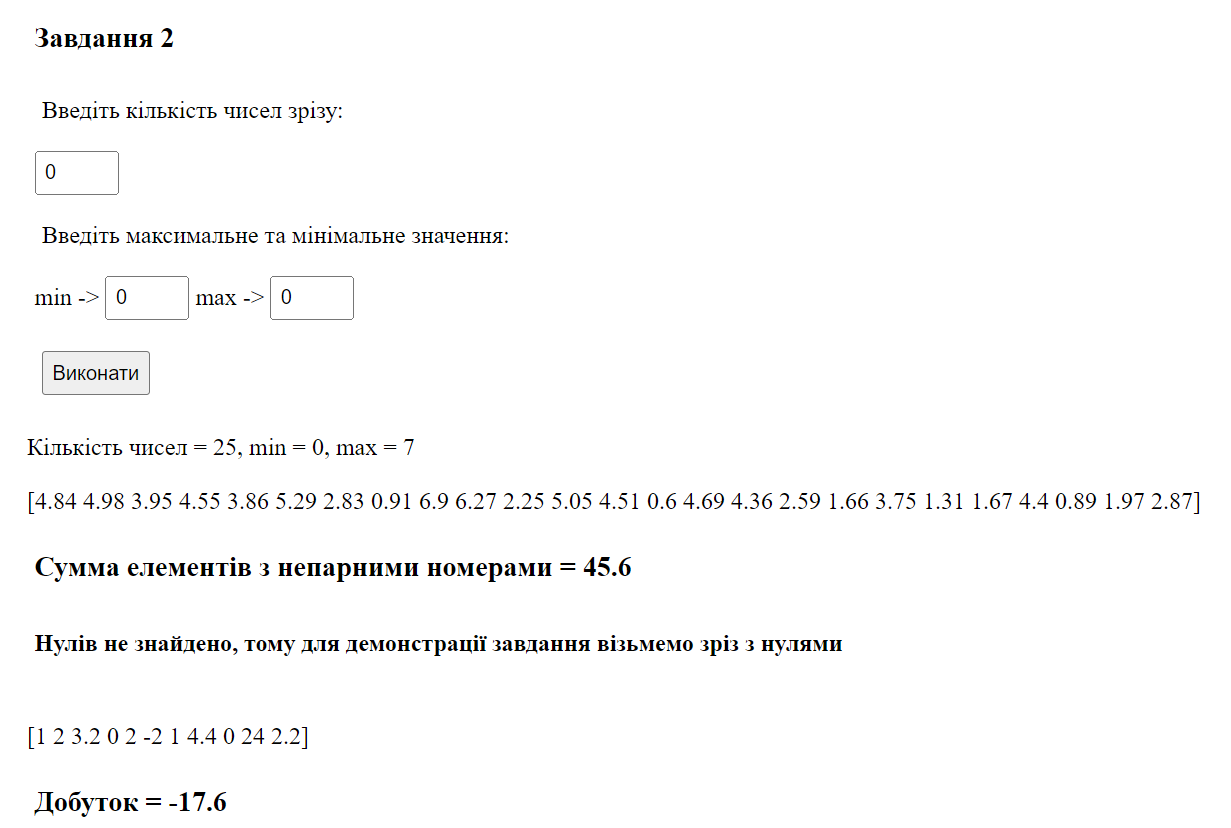


Рис.8. Результат



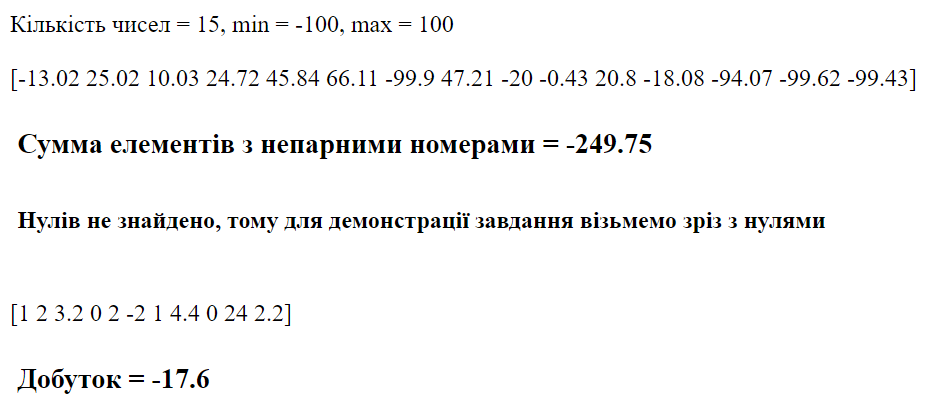


Рис.9. Результат



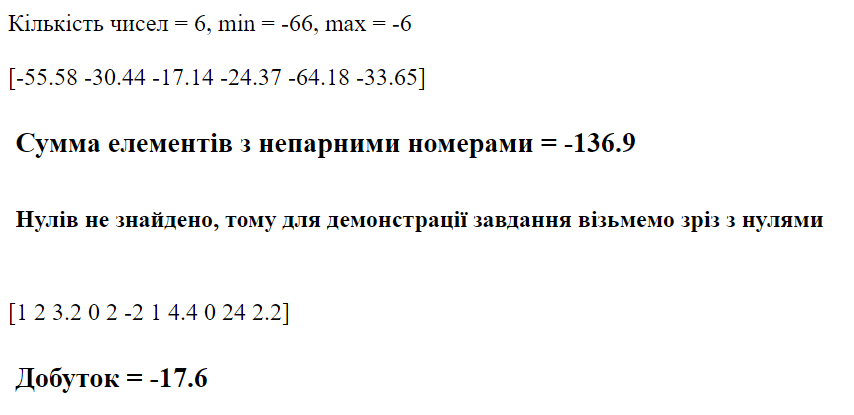


Рис.10. Результат

***Висновки:*** у ході виконання лабораторної роботи було засвоєно принцип проектування веб-серверу, було отримано практичні знання по роботі з пакетом net/http та get- та post-методами.