Одномерный интервальный массив A состоит из N элементов , значения который определяется формулами:

Где ;

Базовый действительный тип - double

* Вычислить сумму элементов массива при N=10
* Вывести исходный массив (мантисса 5 знаков) и сумму (мантисса 15 знаков)
* Упорядочить массив по возрастанию, используя любую сортировку
* Вычислить сумму элементов упорядоченного массива
* Вывести упорядоченный массив (Мантисса 5 знаков) и полученную сумму (мантисса 15 знаков)

package main

import (

"fmt"

"math"

"sort"

)

const V = 3

const rad = 0.001

type Interval struct {

l, r float64

}

func (i Interval) Add(other Interval) Interval {

return Interval{i.l + other.l, i.r + other.r}

}

func makeArr(N int) []Interval {

arr := make([]Interval, N)

for i := 0; i < N; i++ {

arr[i].l = math.Sin(float64(V+i)) - rad

arr[i].r = math.Sin(float64(V+i)) + rad

}

return arr

}

func displayIntervals(arr []Interval) {

fmt.Println("\nИнтервалы:")

for \_, interval := range arr {

fmt.Printf("[%0.5f, %0.5f]\n", interval.l, interval.r)

}

fmt.Println()

}

func calculateSum(arr []Interval) Interval {

var sum Interval

for \_, interval := range arr {

sum = sum.Add(interval)

}

return sum

}

func main() {

var N int

fmt.Print("Введите N: ")

fmt.Scan(&N)

arr := makeArr(N)

sumBeforeSorting := calculateSum(arr)

fmt.Println("Исходный массив")

displayIntervals(arr)

sort.Slice(arr, func(i, j int) bool {

return arr[i].l < arr[j].l

})

fmt.Println("Сортированные интервалы по полю l (по левому краю каждого интервала):")

displayIntervals(arr)

sumAfterSorting := calculateSum(arr)

fmt.Println("Сумма интервалов до сортировки: \t", sumBeforeSorting)

fmt.Println("Сумма интервалов после сортировки: \t", sumAfterSorting)

}

