

Код библиотеки

package intervalLib

import (

"math"

"sort"

)

type Interval struct {

L, R float64

}

// Операция округления до заданного знака

func RoundToDecimal(num Interval, decimals int) Interval {

scale := math.Pow(10, float64(decimals))

return Interval{L: math.Round(num.L\*scale) / scale, R: math.Round(num.R\*scale) / scale}

}

// Возвращает наибольшее значениее из 2-ух интервалов

func Comparison(X, Y Interval) Interval {

var res Interval

if X.R < Y.L {

return Y

} else if X.L > Y.R {

return X

}

return res

}

// Операция умножения интервалов

func Multiplication(X, Y Interval) Interval {

tmp := make([]float64, 4)

tmp[0] = X.L \* Y.L

tmp[1] = X.L \* Y.R

tmp[2] = X.R \* Y.L

tmp[3] = X.R \* Y.R

sort.Float64s(tmp)

res := Interval{L: tmp[0], R: tmp[len(tmp)-1]}

return RoundToDecimal(res, 15)

}

// Операция деления интервалов

func Division(X, Y Interval) Interval {

tmp := make([]float64, 4)

tmp[0] = X.L / Y.L

tmp[1] = X.L / Y.R

tmp[2] = X.R / Y.L

tmp[3] = X.R / Y.R

sort.Float64s(tmp)

res := Interval{L: tmp[0], R: tmp[len(tmp)-1]}

return RoundToDecimal(res, 15)

}

// Операция сложения интервалов

func Addition(X, Y Interval) Interval {

tmp := make([]float64, 2)

tmp[0] = X.L + Y.L

tmp[1] = X.R + Y.R

sort.Float64s(tmp)

res := Interval{L: tmp[0], R: tmp[len(tmp)-1]}

return RoundToDecimal(res, 15)

}

// Операция вычитания интервалов

func Substraction(X, Y Interval) Interval {

tmp := make([]float64, 2)

tmp[0] = X.L - Y.R

tmp[1] = X.R - Y.L

sort.Float64s(tmp)

res := Interval{L: tmp[0], R: tmp[len(tmp)-1]}

return RoundToDecimal(res, 15)

}

// Операция нестандартного вычитания интервалов

func NonStandardSubtraction(X, Y Interval) Interval {

tmp := make([]float64, 2)

tmp[0] = X.L - Y.L

tmp[1] = X.R - Y.R

sort.Float64s(tmp)

res := Interval{L: tmp[0], R: tmp[len(tmp)-1]}

return res

}

// Операция нестандартного деления интервалов

func NonStandardDivision(X, Y Interval) Interval {

if Comparison(Multiplication(X, Y), Interval{L: 0, R: 0}) == Multiplication(X, Y) {

tmp := make([]float64, 2)

tmp[0] = X.L / Y.L

tmp[1] = X.R / Y.R

sort.Float64s(tmp)

res := Interval{L: tmp[0], R: tmp[1]}

return res

} else if Comparison(Multiplication(X, Y), Interval{L: 0, R: 0}) != Multiplication(X, Y) {

tmp := make([]float64, 2)

tmp[0] = X.L / Y.R

tmp[1] = X.R / Y.L

sort.Float64s(tmp)

res := Interval{L: tmp[0], R: tmp[1]}

return res

} else if X.L <= 0 && X.R >= 0 && Y.L > 0 && Y.R > 0 {

tmp := 1 / Y.L

return Interval{L: tmp \* X.L, R: tmp \* X.R}

} else if X.L <= 0 && X.R >= 0 && Y.L < 0 && Y.R < 0 {

tmp := 1 / Y.R

return Interval{L: tmp \* X.L, R: tmp \* X.R}

} else {

panic("Данные вычисления не предусмотренны библиотекой")

}

}

func Getinstruction() string {

str := "Инструкция для работы с библиотекой\nComparison(X, Y Interval) Interval - Возвращает набольшее значение из 2-ух интервалов\nMultiplication(X, Y Interval) Interval - Возвращает интервал, являющийся произведением 2-ух интервалов\nDivision(X, Y Interval) Interval - Возвращает интервал, являющийся результатом деления 2-ух интервалов\nAddition(X, Y Interval) Interval - Возвращает интервал, являющийся суммой 2-ух интервалов\nSubstraction(X, Y Interval) Interval - Возвращает интервал, являющийся разницей 2-ух интервалов\nNonStandardSubtraction(X, Y Interval) Interval - Выполняет нестандартное делени 2-ух интервалов\nNonStandardDivision(X, Y Interval) Interval - Выполняет нестандартное вычитание 2-ух интервалов\n}"

return str

}

Код программы

package main

import (

"fmt"

"intervalLib"

)

func main() {

fmt.Println("Если хотите прочитать инструкцию нажмите 1")

var choice int

fmt.Scan(&choice)

if choice == 1 {

str := intervalLib.Getinstruction()

fmt.Print(str)

} else {

var a, b intervalLib.Interval

fmt.Print("Введите границы первого интервала: ")

fmt.Scan(&a.L, &a.R)

fmt.Print("Введите границы второго интервала: ")

fmt.Scan(&b.L, &b.R)

fmt.Println()

result := intervalLib.Multiplication(a, b)

fmt.Printf("Умножение: [%.15f, %.15f]\n\n", result.L, result.R)

result = intervalLib.Division(a, b)

fmt.Printf("Деление: [%.15f, %.15f]\n\n", result.L, result.R)

result = intervalLib.Addition(a, b)

fmt.Printf("Сложение: [%.15f, %.15f]\n\n", result.L, result.R)

result = intervalLib.Substraction(a, b)

fmt.Printf("Вычитание: [%.15f, %.15f]\n\n", result.L, result.R)

result = intervalLib.NonstandardDivision(a, b)

fmt.Printf("Нестандартное деление: [%.15f, %.15f]\n\n", result.L, result.R)

result = intervalLib.NonstandardSubtraction(a, b)

fmt.Printf("Нестандартное вычитание: [%.15f, %.15f]\n\n", result.L, result.R)

}

}

