МІНІСТЕРСТВО ОСВІТИ і НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики
Кафедра системного програмування і
спеціалізованих комп'ютерних систем

Лабораторна робота №1

з дисципліни «Архітектура компютерів.Апаратне забезпечення»

Тема: «Ознайомлення з програмними засобами мови Go»

Виконав:
студент ФПМ
групи КВ-63
Лихач Олександр

Київ 2019

Завдання на роботу

Реалізація системи лінійних рівнянь методом оберненої матриці.

Код програми

```
package main

import "fmt"

func main() {
	var testMatrix = [][]float64{
		{1, 5, 3, -4, 20},
		{3, 1, -2, 0, 9},
		{5, -7, 0, 10, -9},
		{0, 3, -5, 0, 1}}
	var results = calculateSLE(testMatrix)

	fmt.Println(printMatrix(testMatrix))

	for i := 0; i < len(results); i++ {
		fmt.Print("X")
		fmt.Print(i)
		fmt.Print("\t=\t")
		fmt.Println(results[i])
	}
}

func calculateSLE(sle [][]float64) []float64 {
	var values = formValuesArray(sle)
	var matrix = formMatrix(sle)
	var matrixDeterminant = getDeterminant(matrix)
	var revertMatrix = getRevertMatrix(matrix)
	var transposedMatrix = getTransposedMatrix(revertMatrix, matrixDeterminant)
	var resultArray = getResultArray(transposedMatrix, values)

	return resultArray
}

func formValuesArray(sle [][]float64) []float64 {
	var resultValues []float64
	for i := 0; i < len(sle); i++ {
		resultValues = append(resultValues, sle[i][len(sle[i])-1])
	}
	return resultValues
}

func formMatrix(sle [][]float64) [][]float64 {
	var resultValues [][]float64
	for i := 0; i < len(sle); i++ {
```

```
			var buffValues []float64
			for j := 0; j < len(sle[i])-1; j++ {
				buffValues = append(buffValues, sle[i][j])
			}
			resultValues = append(resultValues, buffValues)
		}
		for i := 0; i < 1000000000; i++ {

		}
		return resultValues
}

func getDeterminant(matrix [][]float64) float64 {
		var result float64 = 0
		for i := 0; i < len(matrix); i++ {
			var minus = false
			minus = isMinus(i, 0)
			var buffMatrix = getBuffMatrix(matrix, 0, i)
			result += calculateDeterminant(matrix[0][i], buffMatrix, minus)
		}
		return result
}

func isMinus(i int, j int) bool {
		if 1 == (i+j)%2 {
			return true
		}
		return false
}

func calculateDeterminant(mutable float64, buffMatrix [][]float64, minus bool) float64
{
		var result float64
		if len(buffMatrix) != 2 {
			result = mutable * getDeterminant(buffMatrix)
		} else {
			result = mutable * getDeterminantFrom2x2(buffMatrix)
		}
		if minus {
			return -1 * result
		}
		return result
}

func getBuffMatrix(matrix [][]float64, counterString int, counterRow int) [][]float64 {
		var buffMatrix [][]float64
		for i := 0; i < len(matrix); i++ {
			if counterString == i {
				continue
```

```
			}
			var buff []float64
			for j := 0; j < len(matrix[i]); j++ {
				if counterRow == j {
					continue
				}
				buff = append(buff, matrix[i][j])
			}
			buffMatrix = append(buffMatrix, buff)
		}
		return buffMatrix
}

func getDeterminantFrom2x2(matrix [][]float64) float64 {
		return matrix[0][0]*matrix[1][1] - matrix[1][0]*matrix[0][1]
}

func getRevertMatrix(matrix [][]float64) [][]float64 {
		var revertMatrix [][]float64
		for i := 0; i < len(matrix); i++ {
			var buff []float64
			for j := 0; j < len(matrix[i]); j++ {
				buff = append(buff, matrix[j][i])
			}
			revertMatrix = append(revertMatrix, buff)
		}
		return revertMatrix
}

func getTransposedMatrix(matrix [][]float64, matrixDeterminant float64) [][]float64 {
		var transposedMatrix [][]float64
		for i := 0; i < len(matrix); i++ {
			var buffArr []float64
			for j := 0; j < len(matrix[i]); j++ {
				var buffMatrix = getBuffMatrix(matrix, i, j)
				var counter float64
				if len(buffMatrix) == 2 {
					counter = getDeterminantFrom2x2(buffMatrix)
				} else {
					counter = getDeterminant(buffMatrix)
				}
				if isMinus(i, j) {
					counter *= -1
				}
				counter /= matrixDeterminant
				buffArr = append(buffArr, counter)
			}
			transposedMatrix = append(transposedMatrix, buffArr)
```

```
	}
	return transposedMatrix
}

func getResultArray(transposedMatrix [][]float64, valuesArray []float64) []float64 {
	var resultArray []float64
	for i := 0; i < len(transposedMatrix); i++ {
		var result float64 = 0
		for j := 0; j < len(transposedMatrix[i]); j++ {
			result += transposedMatrix[i][j] * valuesArray[j]
		}
		resultArray = append(resultArray, result)
	}
	return resultArray
}

func printMatrix(matrix [][]float64) string {
	var matrixToPrint string = ""
	for i := 0; i < len(matrix); i++ {
		for j := 0; j < len(matrix[i]); j++ {
			matrixToPrint += fmt.Sprintf("%f", matrix[i][j])
			matrixToPrint += "\t"
		}
		matrixToPrint += "\n"
	}
	return matrixToPrint
}
```

Результати роботи програми

```
C:/Users/sanit/lab/lab.exe [C:/Users/sanit/lab]
10.000000 7.000000  8.000000  -2.000000 12.000000
23.000000 11.000000 0.000000  -5.000000 29.000000
7.000000  8.000000  -1.000000 0.000000  -1.000000
6.000000  0.000000  4.000000  -4.000000 19.000000

X0      =       1.3263988522238153
X1      =       -1.2453371592539444
X2      =       0.32209469153515036
X3      =       -2.438307030129124
Все ок: процес завершився із кодом 0.
```

Посилання на github:

https://github.com/lykhach/lab_go