

ЛАБОРАТОРНА РОБОТА № 2

Організація коду в мові Go. Пакети. Тестування

Мета: розглянути організацію коду в мові Go: головний пакет; додаткові пакети, зовнішні пакети. Область видимості ідентифікаторів пакета. Отримати практичні навички у створенні пакетів (одного, двох і більше), організувати тестування функцій пакета.

Хід роботи:

Завдання 1. Реалізувати програму в одному вхідному файлі з трьома функціями:

- знаходження мінімального значення з трьох елементів
- обчислення середнього значення трьох елементів
- рішення рівняння першого порядку;.

Листинг програми:

```
package main

import "fmt"

func Min3(a, b, c float64) float64 {
    min := a
    if b < min {
        min = b
    }
    if c < min {
        min = c
    }
    return min
}

func Average3(a, b, c float64) float64 {
    return (a + b + c) / 3.0
}

func SolveLinear(a, b float64) (float64, bool) {
    if a == 0 {
        return 0, false
    }
    return -b / a, true
}

func main() {
    fmt.Println("=== Тестування Min3 ===")
    fmt.Printf("Min3(5, 2, 8) = %.2f\n", Min3(5, 2, 8))
    fmt.Printf("Min3(-1, -5, 3) = %.2f\n", Min3(-1, -5, 3))
}
```

					ДУ «Житомирська політехніка».25.121.20.000 – Лр2				
Змн.	Арк.	№ докум.	Підпис	Дата					
Розроб.		Рижченко Я.В.			Звіт з лабораторної роботи	Лім.	Арк.	Аркушів	
Перевір.		Петросян Р.В.					1	7	
Керівник						ФІКТ Гр. ІПЗ-23-1			
Н. контр.									
Зав. каф.									

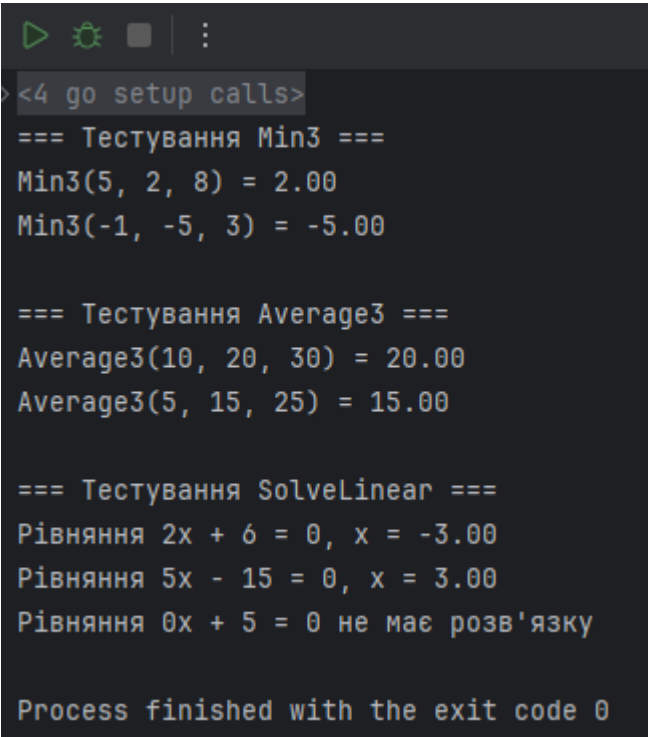
```
fmt.Println("\n=== Тестування Average3 ===")
fmt.Printf("Average3(10, 20, 30) = %.2f\n", Average3(10, 20, 30))
fmt.Printf("Average3(5, 15, 25) = %.2f\n", Average3(5, 15, 25))

fmt.Println("\n=== Тестування SolveLinear ===")
if x, ok := SolveLinear(2, 6); ok {
    fmt.Printf("Рівняння 2x + 6 = 0, x = %.2f\n", x)
} else {
    fmt.Println("Рівняння не має розв'язку")
}

if x, ok := SolveLinear(5, -15); ok {
    fmt.Printf("Рівняння 5x - 15 = 0, x = %.2f\n", x)
} else {
    fmt.Println("Рівняння не має розв'язку")
}

if x, ok := SolveLinear(0, 5); ok {
    fmt.Printf("Рівняння 0x + 5 = 0, x = %.2f\n", x)
} else {
    fmt.Println("Рівняння 0x + 5 = 0 не має розв'язку")
}
}
```

Результат виконання:



```
> <4 go setup calls>
=== Тестування Min3 ===
Min3(5, 2, 8) = 2.00
Min3(-1, -5, 3) = -5.00

=== Тестування Average3 ===
Average3(10, 20, 30) = 20.00
Average3(5, 15, 25) = 15.00

=== Тестування SolveLinear ===
Рівняння 2x + 6 = 0, x = -3.00
Рівняння 5x - 15 = 0, x = 3.00
Рівняння 0x + 5 = 0 не має розв'язку

Process finished with the exit code 0
```

Рис. 1. Завдання 1.

Завдання 2. Розбити додаток з 2 вихідними файлами (у другій файл винести реалізовані три функції)

Листинг файлу main.go:

```
package main

import "fmt"
```

```

func main() {
    fmt.Println("=== Тестування Min3 ===")
    fmt.Printf("Min3(5, 2, 8) = %.2f\n", Min3(5, 2, 8))
    fmt.Printf("Min3(-1, -5, 3) = %.2f\n", Min3(-1, -5, 3))
    fmt.Println("\n=== Тестування Average3 ===")
    fmt.Printf("Average3(10, 20, 30) = %.2f\n", Average3(10, 20, 30))
    fmt.Printf("Average3(5, 15, 25) = %.2f\n", Average3(5, 15, 25))
    fmt.Println("\n=== Тестування SolveLinear ===")
    if x, ok := SolveLinear(2, 6); ok {
        fmt.Printf("Рівняння  $2x + 6 = 0$ ,  $x = %.2f$ \n", x)
    } else {
        fmt.Println("Рівняння не має розв'язку")
    }

    if x, ok := SolveLinear(5, -15); ok {
        fmt.Printf("Рівняння  $5x - 15 = 0$ ,  $x = %.2f$ \n", x)
    } else {
        fmt.Println("Рівняння не має розв'язку")
    }

    if x, ok := SolveLinear(0, 5); ok {
        fmt.Printf("Рівняння  $0x + 5 = 0$ ,  $x = %.2f$ \n", x)
    } else {
        fmt.Println("Рівняння  $0x + 5 = 0$  не має розв'язку")
    }
}

```

Листинг файлу mathFunc.go:

```

package main

func Min3(a, b, c float64) float64 {
    min := a
    if b < min {
        min = b
    }
    if c < min {
        min = c
    }
    return min
}

func Average3(a, b, c float64) float64 {
    return (a + b + c) / 3.0
}

func SolveLinear(a, b float64) (float64, bool) {
    if a == 0 {
        return 0, false
    }
    return -b / a, true
}

```

Результат виконання:

```

PS D:\Work\60\lab2\ex2> go run .
=== Тестування Min3 ===
Min3(5, 2, 8) = 2.00
Min3(-1, -5, 3) = -5.00

=== Тестування Average3 ===
Average3(10, 20, 30) = 20.00
Average3(5, 15, 25) = 15.00

=== Тестування SolveLinear ===
Рівняння  $2x + 6 = 0$ ,  $x = -3.00$ 
Рівняння  $5x - 15 = 0$ ,  $x = 3.00$ 
Рівняння  $0x + 5 = 0$  не має розв'язку
PS D:\Work\60\lab2\ex2>

```

Рис. 2. Завдання 2.

		Рижченко Я.В.			ДУ «Житомирська політехніка».25.121.20.000 – Пр2	Арк.
		Петросян Р.В.				3
Змн.	Арк.	№ докум.	Підпис	Дата		

Завдання 3. Реалізувати пакет (в пакет винести реалізовані три функції), додайте декілька тестів для вашого пакета.

Листинг файлу main.go:

```
package main

import (
    "ex3/math"
    "fmt"
)

func main() {
    fmt.Println("=== Тестування Min3 ===")
    fmt.Printf("Min3(5, 2, 8) = %.2f\n", math.Min3(5, 2, 8))
    fmt.Printf("Min3(-1, -5, 3) = %.2f\n", math.Min3(-1, -5, 3))

    fmt.Println("\n=== Тестування Average3 ===")
    fmt.Printf("Average3(10, 20, 30) = %.2f\n", math.Average3(10, 20, 30))
    fmt.Printf("Average3(5, 15, 25) = %.2f\n", math.Average3(5, 15, 25))

    fmt.Println("\n=== Тестування SolveLinear ===")
    if x, ok := math.SolveLinear(2, 6); ok {
        fmt.Printf("Рівняння 2x + 6 = 0, x = %.2f\n", x)
    } else {
        fmt.Println("Рівняння не має розв'язку")
    }

    if x, ok := math.SolveLinear(5, -15); ok {
        fmt.Printf("Рівняння 5x - 15 = 0, x = %.2f\n", x)
    } else {
        fmt.Println("Рівняння не має розв'язку")
    }

    if x, ok := math.SolveLinear(0, 5); ok {
        fmt.Printf("Рівняння 0x + 5 = 0, x = %.2f\n", x)
    } else {
        fmt.Println("Рівняння 0x + 5 = 0 не має розв'язку")
    }
}
```

Листинг файлу math.go, що належить пакету math:

```
package math

func Min3(a, b, c float64) float64 {
    min := a
    if b < min {
        min = b
    }
    if c < min {
        min = c
    }
    return min
}

func Average3(a, b, c float64) float64 {
    return (a + b + c) / 3.0
}

func SolveLinear(a, b float64) (float64, bool) {
    if a == 0 {
        return 0, false
    }
}
```

		Рижченко Я.В.			ДУ «Житомирська політехніка».25.121.20.000 – Лр2	Арк.
		Петросян Р.В.				4
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }
    return -b / a, true
}

```

Листинг файлу math_test.go, що належить пакету math:

```

package math

import (
    "math"
    "testing"
)

func TestMin3(t *testing.T) {
    tests := []struct {
        name      string
        a, b, c   float64
        expected  float64
    }{
        {"Positive numbers", 5, 2, 8, 2},
        {"Negative numbers", -1, -5, 3, -5},
        {"All equal", 7, 7, 7, 7},
        {"First is min", 1, 5, 10, 1},
        {"Last is min", 10, 5, 1, 1},
        {"Mixed signs", -2, 0, 5, -2},
        {"Zero values", 0, 0, 0, 0},
        {"Decimals", 3.5, 2.1, 8.9, 2.1},
    }
    for _, tt := range tests {
        t.Run(tt.name, func(t *testing.T) {
            result := Min3(tt.a, tt.b, tt.c)
            if result != tt.expected {
                t.Errorf("Min3(%.2f, %.2f, %.2f) = %.2f; want %.2f",
                    tt.a, tt.b, tt.c, result, tt.expected)
            }
        })
    }
}

func TestAverage3(t *testing.T) {
    tests := []struct {
        name      string
        a, b, c   float64
        expected  float64
    }{
        {"Positive numbers", 10, 20, 30, 20},
        {"Same numbers", 5, 5, 5, 5},
        {"Negative numbers", -3, -6, -9, -6},
        {"Mixed signs", -5, 0, 5, 0},
        {"Zero values", 0, 0, 0, 0},
        {"Decimals", 1.5, 2.5, 3.0, 2.3333333333333333},
        {"Large numbers", 100, 200, 300, 200},
    }
    for _, tt := range tests {
        t.Run(tt.name, func(t *testing.T) {
            result := Average3(tt.a, tt.b, tt.c)
            if math.Abs(result-tt.expected) > 1e-9 {
                t.Errorf("Average3(%.2f, %.2f, %.2f) = %.10f; want %.10f",
                    tt.a, tt.b, tt.c, result, tt.expected)
            }
        })
    }
}

func TestSolveLinear(t *testing.T) {
    tests := []struct {

```

		Рижченко Я.В.			ДУ «Житомирська політехніка».25.121.20.000 – Пр2	Арк.
		Петросян Р.В.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    name      string
    a, b      float64
    expected  float64
    hasRoot   bool
}

{
    {"Simple equation 2x + 6 = 0", 2, 6, -3, true},
    {"Simple equation 5x - 15 = 0", 5, -15, 3, true},
    {"Equation with a=0", 0, 5, 0, false},
    {"Negative coefficient", -3, 9, 3, true},
    {"Root is zero", 4, 0, 0, true},
    {"Decimal coefficients", 2.5, 5, -2, true},
    {"Large numbers", 100, 200, -2, true},
}

for _, tt := range tests {
    t.Run(tt.name, func(t *testing.T) {
        result, ok := SolveLinear(tt.a, tt.b)
        if ok != tt.hasRoot {
            t.Errorf("SolveLinear(%.2f, %.2f) returned ok=%v; want ok=%v",
                tt.a, tt.b, ok, tt.hasRoot)
        }
        if ok && math.Abs(result-tt.expected) > 1e-9 {
            t.Errorf("SolveLinear(%.2f, %.2f) = %.10f; want %.10f",
                tt.a, tt.b, result, tt.expected)
        }
    })
}

}

func BenchmarkMin3(b *testing.B) {
    for i := 0; i < b.N; i++ {
        Min3(5.5, 2.3, 8.7)
    }
}

func BenchmarkAverage3(b *testing.B) {
    for i := 0; i < b.N; i++ {
        Average3(10, 20, 30)
    }
}

func BenchmarkSolveLinear(b *testing.B) {
    for i := 0; i < b.N; i++ {
        SolveLinear(2, 6)
    }
}
}

```

Результат виконання програми:

```

PS D:\Work\60\lab2\ex3> go run .
=== Тестування Min3 ===
Min3(5, 2, 8) = 2.00
Min3(-1, -5, 3) = -5.00

=== Тестування Average3 ===
Average3(10, 20, 30) = 20.00
Average3(5, 15, 25) = 15.00

=== Тестування SolveLinear ===
Рівняння 2x + 6 = 0, x = -3.00
Рівняння 5x - 15 = 0, x = 3.00
Рівняння 0x + 5 = 0 не має розв'язку
PS D:\Work\60\lab2\ex3>

```

Рис. 3. Завдання 3.

		Рижченко Я.В.			ДУ «Житомирська політехніка».25.121.20.000 – Лр2	Арк.
		Петросян Р.В.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

Результати тестів:

```
PS D:\Work\G0\lab2\ex3> go test -cover ./math
ok      ex3/math      0.563s coverage: 100.0% of statements
PS D:\Work\G0\lab2\ex3> go test -v -cover -bench=. ./math
ex3
coverage: 0.0% of statements

=== RUN    TestMin3
=== RUN    TestMin3/Positive_numbers
=== RUN    TestMin3/Negative_numbers
=== RUN    TestMin3/All_equal
=== RUN    TestMin3/First_is_min
=== RUN    TestMin3/Last_is_min
=== RUN    TestMin3/Mixed_signs
=== RUN    TestMin3/Zero_values
=== RUN    TestMin3/Decimals
--- PASS: TestMin3 (0.00s)
    --- PASS: TestMin3/Positive_numbers (0.00s)
    --- PASS: TestMin3/Negative_numbers (0.00s)
    --- PASS: TestMin3/All_equal (0.00s)
    --- PASS: TestMin3/First_is_min (0.00s)
    --- PASS: TestMin3/Last_is_min (0.00s)
    --- PASS: TestMin3/Mixed_signs (0.00s)
    --- PASS: TestMin3/Zero_values (0.00s)
    --- PASS: TestMin3/Decimals (0.00s)
=== RUN    TestAverage3
=== RUN    TestAverage3/Positive_numbers
=== RUN    TestAverage3/Same_numbers
=== RUN    TestAverage3/Negative_numbers
=== RUN    TestAverage3/Mixed_signs
=== RUN    TestAverage3/Zero_values
=== RUN    TestAverage3/Decimals
=== RUN    TestAverage3/Large_numbers
--- PASS: TestAverage3 (0.00s)
    --- PASS: TestAverage3/Positive_numbers (0.00s)
    --- PASS: TestAverage3/Same_numbers (0.00s)
    --- PASS: TestAverage3/Negative_numbers (0.00s)
    --- PASS: TestAverage3/Mixed_signs (0.00s)
    --- PASS: TestAverage3/Zero_values (0.00s)
    --- PASS: TestAverage3/Decimals (0.00s)
    --- PASS: TestAverage3/Large_numbers (0.00s)
```

Рис. 4. Результати тестів.

Посилання на репозиторій - <https://github.com/JanRizhenko/GoLang-sigma-practice>

Висновок: У ході виконання лабораторної роботи було розглянуто принципи організації коду в мові Go та практичне застосування пакетів для структурування програми. Робота дала можливість на практиці створити власні пакети, розділити функціональність між кількома файлами та модулями, а також зрозуміти особливості видимості ідентифікаторів у межах пакета. Було опрацьовано механізм імпорту зовнішніх пакетів і використання стандартної бібліотеки, що дозволило побачити, як Go забезпечує модульність та повторне використання коду. Окрему увагу приділено написанню тестів, завдяки чому стало зрозуміло, як перевіряти коректність роботи функцій і підтримувати надійність програмних рішень. Загалом лабораторна робота сформувала цілісне уявлення про те, як правильно організовувати проєкт у Go, структурувати код, створювати пакети та супроводжувати їх тестуванням.

		Рижченко Я.В.			ДУ «Житомирська політехніка».25.121.20.000 – Лр2	Арк.
		Петросян Р.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		7