

**LAPORAN PRAKTIKUM ALGORITMA
DAN PEMROGRAMAN 1**

**MODUL 12
WHILE-LOOP**



Disusun oleh:

ELSA DWI RIZQIYANTI

109082500090

S1IF-13-02

Asisten Praktikum

Adithana dharma putra

Alfin Ilham Berlianto

PROGRAM STUDI S1 INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

LATIHAN KELAS – GUIDED

1. Guided 1

Source Code

```
package main

import "fmt"

func main() {

    var n int

    fmt.Print("Masukkan bilangan: ")

    fmt.Scan(&n)

    if n == 0 || n == 1 {

        fmt.Println("1")

        return

    }

    i := n

    for i >= 1 {

        if i == 1 {

            fmt.Print("1")

        } else {

            fmt.Print(i, " x ")

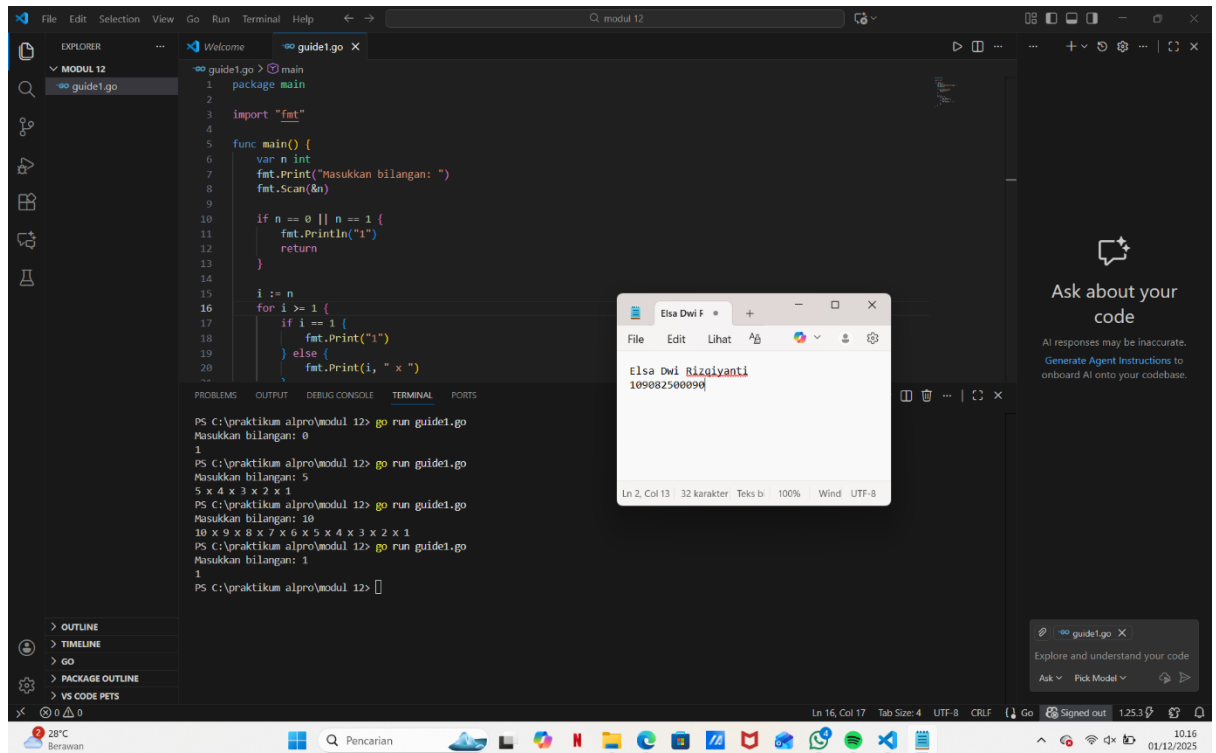
        }

        i--

    }
```

```
}
```

Screenshoot program



Deskripsi program

```
if n == 0 || n == 1 {  
    fmt.Println("1")  
    return  
}
```

Jika pengguna memasukkan 0 atau 1, program langsung mencetak 1.

Kemudian program menghentikan eksekusi dengan return.

Ini dibuat karena perkalian mundur dari 1 sampai 1 atau 0 akan menghasilkan nilai 1 (mirip konsep faktorial).

```
i := n
```

Membuat variabel i dan mengisinya dengan nilai n.

Variabel i digunakan sebagai counter dalam perulangan.

```
for i >= 1 {
```

Ini adalah while-loop gaya Go. Loop akan berjalan selama i >= 1.

Bagian dalam loop:

```
if i == 1 {  
    fmt.Print("1")
```

```

    } else {
        fmt.Print(i, " x ")
    }
    i--

```

Penjelasannya:

1. Jika `i == 1`
Cetak hanya angka 1 tanpa tambahan " x " di belakangnya
(supaya output tidak berakhir dengan simbol " x ")
2. Jika `i > 1`
i x

Contoh:

5 x 4 x 3 x 2 x 1

i --

Mengurangi nilai i satu per satu hingga mencapai 1.

2. Guided 2

Source Code

```

package main

import "fmt"

func main() {

    const tokenBenar = "12345abcde"

    var input string

    fmt.Print("Masukkan token: ")

    fmt.Scan(&input)

    for input != tokenBenar {

        fmt.Print("Masukkan token: ")

        fmt.Scan(&input)
    }
}

```

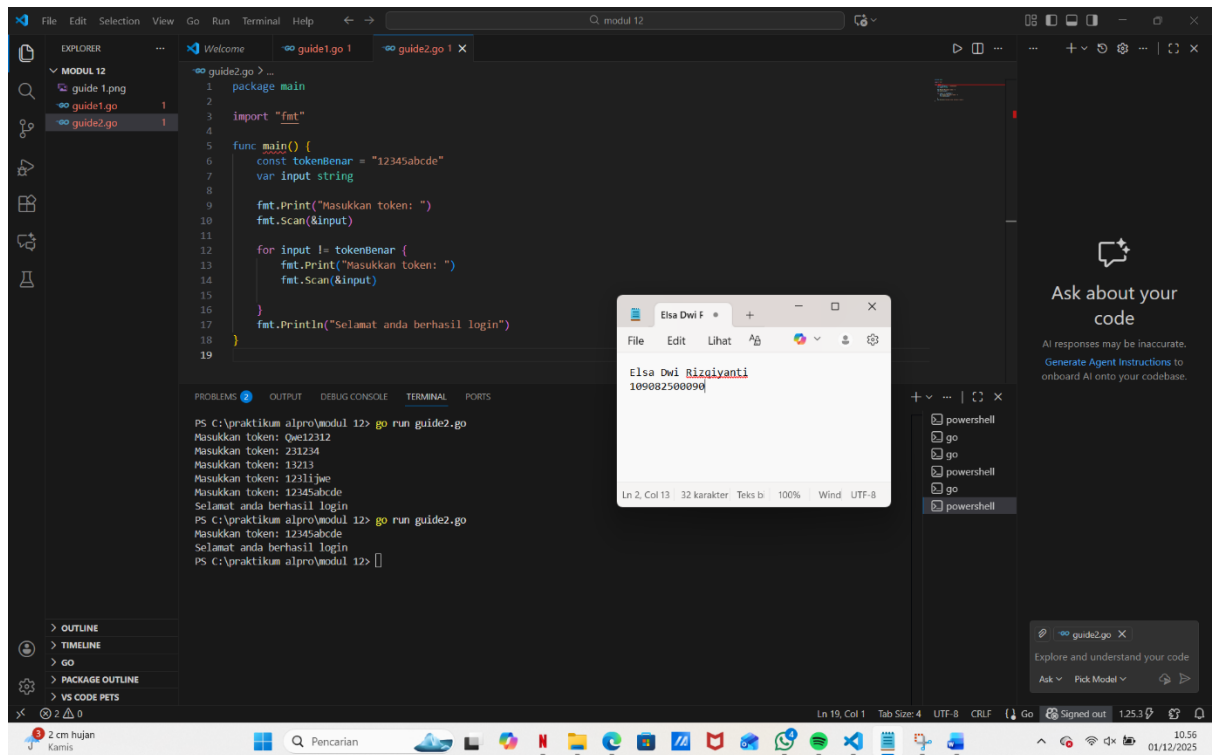
```

    }

    fmt.Println("Selamat anda berhasil login")
}

```

Screenshoot program



Deskripsi program

`const tokenBenar = "12345abcde"`

- Membuat konstanta bernama tokenBenar.
- Konstanta adalah nilai yang *tidak bisa diubah* selama program berjalan.
- Token ini adalah nilai yang harus dimasukkan oleh pengguna agar login berhasil.

Perulangan (While-loop versi Go)

```

for input != tokenBenar {
    fmt.Println("Masukkan token: ")
    fmt.Scan(&input)
}

```

Penjelasan:

- for tanpa kondisi khusus seperti for {} bisa berfungsi sebagai while-loop.

- Kondisi loop: `input != tokenBenar`
artinya: *ulang terus selama input tidak sama dengan token yang benar*
- Jika pengguna salah memasukkan token, loop akan:
 - meminta input ulang
 - membaca input baru
 - memeriksa lagi

Loop berhenti hanya ketika `input == tokenBenar`.

```
fmt.Println("Selamat anda berhasil login")
```

Jika keluar dari loop, artinya token sudah benar.
Program kemudian mencetak pesan sukses login.

3. Guided 3

Source Code

```
package main

import "fmt"

func main() {
    var n int

    fmt.Print("Masukkan N: ")

    fmt.Scan(&n)

    a := 0
    b := 1

    fmt.Print(a, " ", b)
```

```

i := 2

for i < n {

    c := a + b

    fmt.Print(" ", c)

    a = b

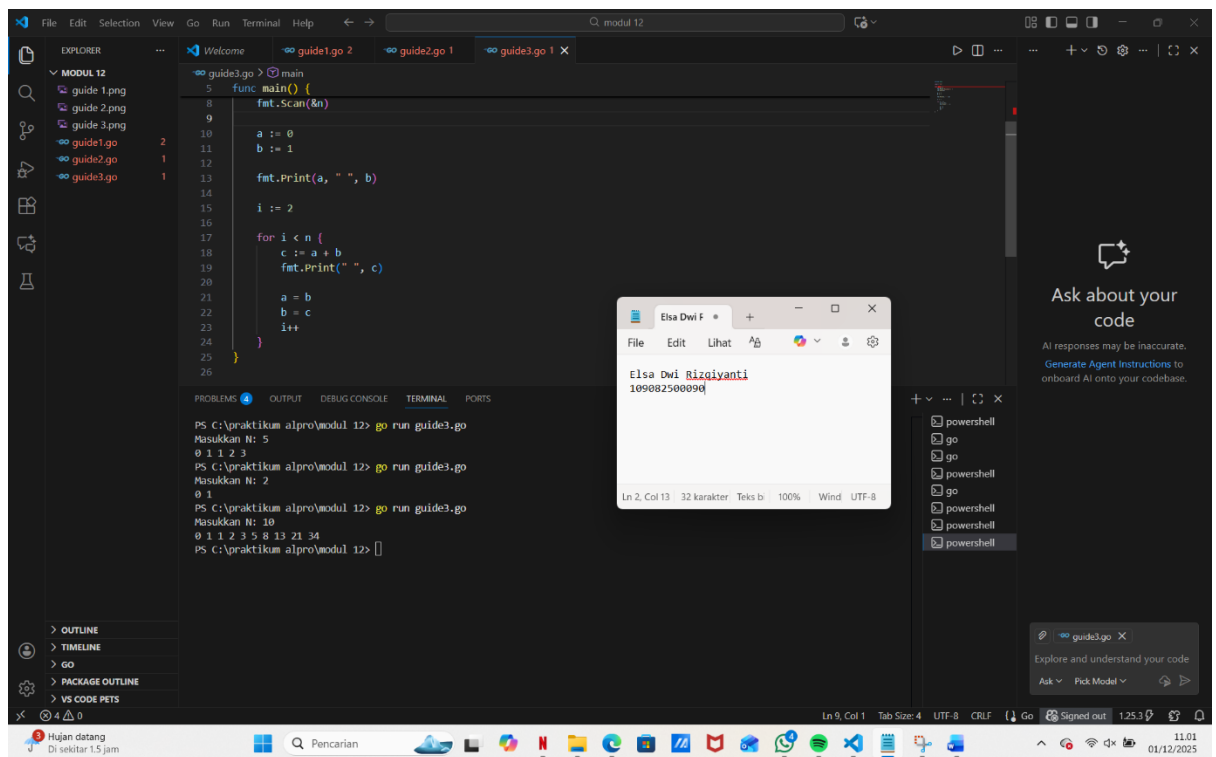
    b = c

    i++

}

```

Screenshoot program



Deskripsi program

Inisialisasi dua angka pertama Fibonacci

`a := 0`

`b := 1`

Deret Fibonacci selalu dimulai dari:

angka pertama = 0

angka kedua = 1

Variabel:

a menyimpan nilai Fibonacci sebelumnya

b menyimpan nilai Fibonacci saat ini

Menampilkan dua bilangan pertama

`fmt.Print(a, " ", b)`

Akan menghasilkan output seperti:

0 1

Ini adalah dua angka pertama deret Fibonacci.

Inisialisasi counter (i)

`i := 2`

Kenapa 2?

Karena deret Fibonacci sudah mencetak dua angka pertama ("0 1"), maka perhitungan berikutnya dimulai dari indeks ke-2.

Perulangan While-loop Fibonacci

`for i < n {`

`c := a + b`

`fmt.Print(" ", c)`

`a = b`

`b = c`

`i++`

`}`

Ini adalah while-loop versi Go.

Penjelasan langkah demi langkah:

Kondisi loop:

`i < n`

Artinya, loop akan terus berjalan sampai jumlah bilangan yang dicetak mencapai N.

Menghitung bilangan Fibonacci berikutnya

`c := a + b`

- c adalah angka Fibonacci baru.
- Rumus Fibonacci:
bilangan berikutnya = dua bilangan sebelumnya dijumlahkan

Menampilkan bilangan ke-i
fmt.Print(" ", c)
Mencetak angka Fibonacci baru.

Geser nilai (update variabel)

a = b

b = c

Pemindahan nilai ini membuat:

- nilai b menjadi nilai Fibonacci sebelumnya pada iterasi selanjutnya
- nilai c menjadi nilai Fibonacci saat ini

Ini penting agar perhitungan Fibonacci terus berlanjut.

Increment counter

i++

Menambah nilai i agar loop mendekati kondisi berhenti.

Hasil Output Contoh

Jika input N = 7

Maka output:

0 1 1 2 3 5 8

Karena itu adalah 7 bilangan pertama Fibonacci.

TUGAS

1. Tugas 1

Source code

```
package main

import "fmt"

func main() {
    var user, pass string
    gagal := 0

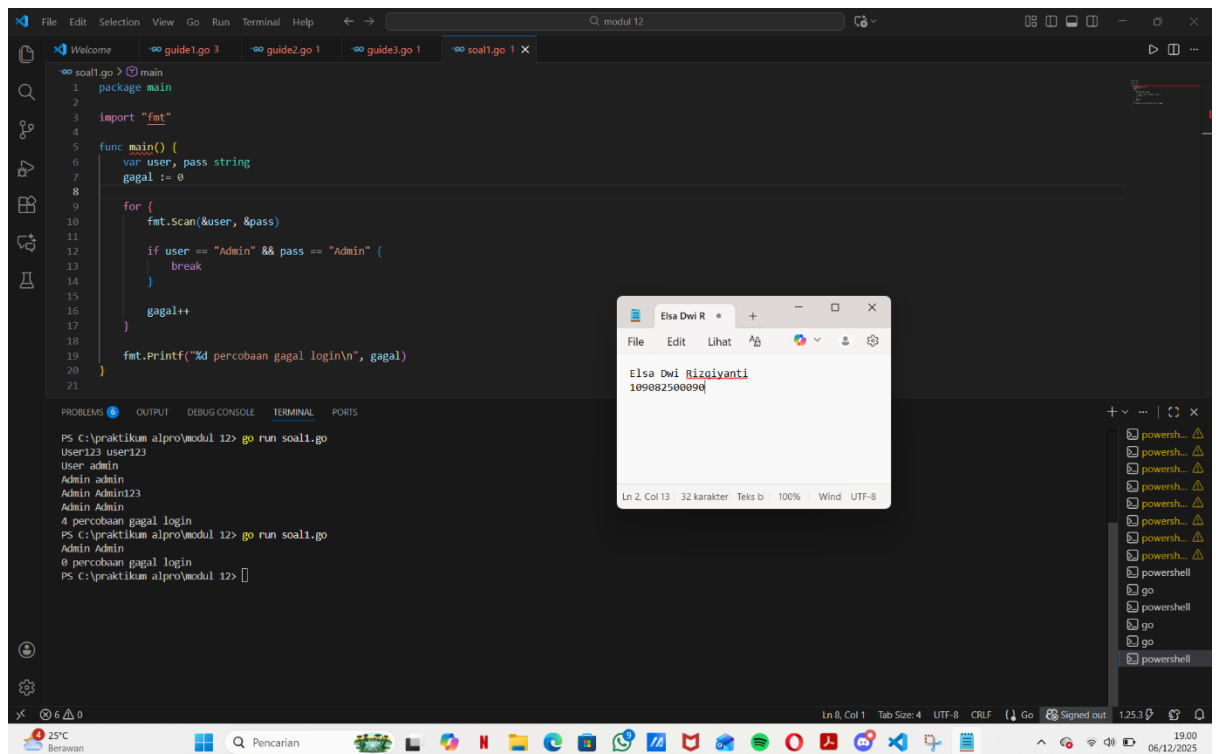
    for {
        fmt.Scan(&user, &pass)

        if user == "Admin" && pass == "Admin" {
            break
        }

        gagal++
    }

    fmt.Printf("%d percobaan gagal login\n", gagal)
}
```

Screenshoot program



Deskripsi program

Input username dan password

`fmt.Scan(&user, &pass)`

Fungsi ini membaca dua input sekaligus dari user:

1. username
2. password

Contoh input:

Admin Admin

Mengecek apakah login benar

```
if user == "Admin" && pass == "Admin" {  
    break  
}
```

Program melakukan pengecekan:

- username harus "Admin"
- password harus "Admin"

Jika kedua kondisi benar, maka:

- break menghentikan loop
- proses login selesai

Jika login salah: tambah hitungan gagal
`gagal++`

Setiap kali user memasukkan kombinasi yang salah, variabel gagal bertambah 1.

Setelah keluar dari loop: tampilkan jumlah kesalahan

```
fmt.Printf("%d percobaan gagal login\n", gagal)
```

Menampilkan jumlah berapa kali login salah dilakukan sebelum akhirnya berhasil.

Contoh output:

4 percobaan gagal login

2. Tugas 2

Source code

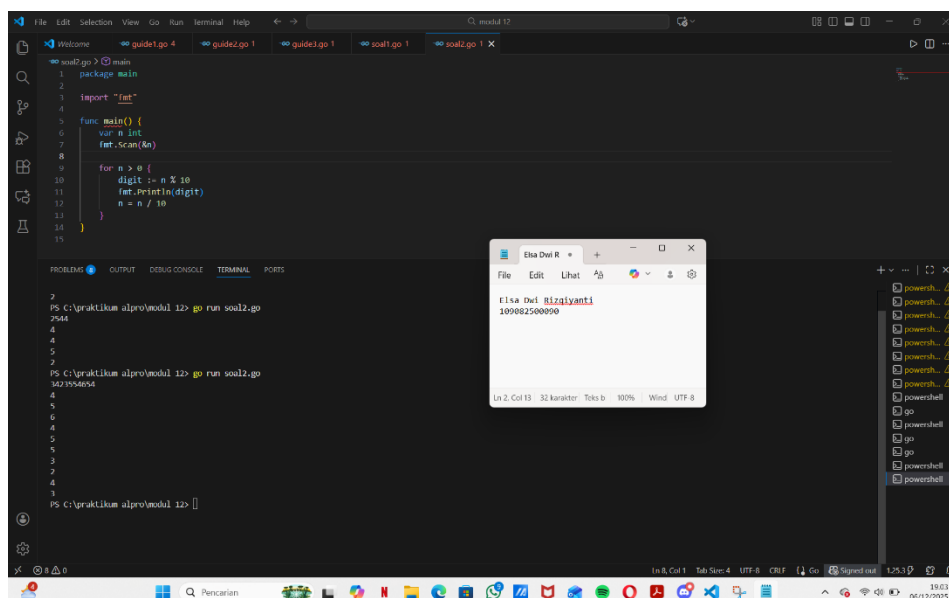
```
package main

import "fmt"

func main() {
    var n int
    fmt.Scan(&n)

    for n > 0 {
        digit := n % 10
        fmt.Println(digit)
        n = n / 10
    }
}
```

Screenshoot program



Deskripsi program

Perulangan while-loop

for $n > 0$ {

Loop ini berjalan selama nilai n masih lebih besar dari 0.

Loop akan berhenti ketika n telah habis dibagi-bagi sehingga menjadi 0.

Mengambil digit terakhir

$\text{digit} := n \% 10$

Operator modulus $\% 10$ digunakan untuk mengambil digit paling belakang dari angka.

Contoh:

- $n = 2544$ digit = 4
- $n = 254$ digit = 4
- $n = 25$ digit = 5

Menampilkan digit

`fmt.Println(digit)`

Digit yang baru diambil ditampilkan satu per baris.

Menghapus digit terakhir

$n = n / 10$

Pembagian integer ($/10$) membuang digit paling belakang.

Contoh:

- $n = 2544$ 254
- $n = 254$ 25
- $n = 25$ 2
- $n = 2$ 0

Ketika n menjadi 0, loop berhenti.

3. Tugas 3

Source code

```
package main

import "fmt"

func main() {
    var x, y int
    var hasil int

    fmt.Scan(&x, &y)
```

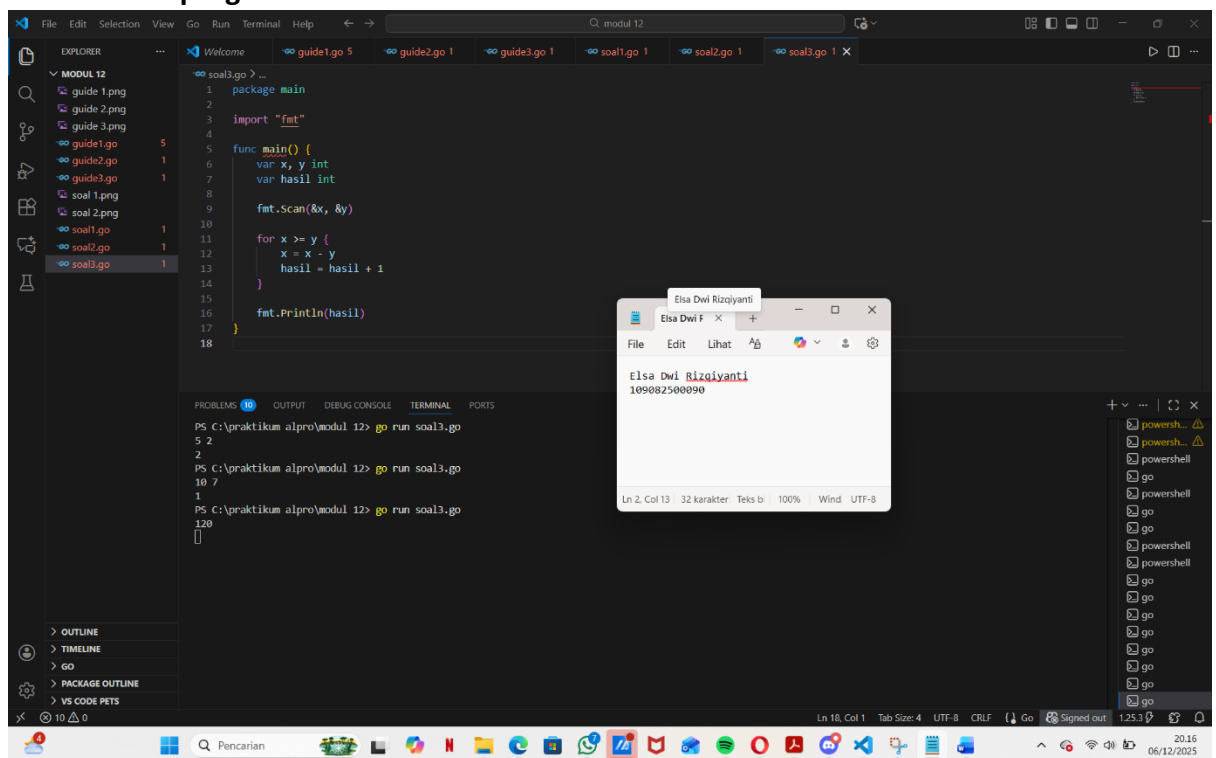
```

    for x >= y {
        x = x - y
        hasil = hasil + 1
    }

    fmt.Println(hasil)
}

```

Screenshoot program



Deskripsi program

Perulangan while-loop (pembagian manual)

```
for x >= y {
```

```
    x = x - y
```

```
    hasil = hasil + 1
```

```
}
```

Kondisi loop

$x \geq y$

Loop terus berjalan selama x masih bisa dikurangi dengan y.

Ini meniru pembagian tanpa operator.

$x = x - y$

Setiap iterasi, program mengurangi x dengan y.

Ini seperti menghitung berapa kali y muat di dalam x.

hasil = hasil + 1

Setiap kali pengurangan dilakukan, itu berarti pembagi (y) masuk sekali ke dalam x.

Maka hasil bertambah 1.

Menampilkan hasil

fmt.Println(hasil)

Setelah loop berhenti, hasil berisi hasil pembagian $x \div y$.

Input:

10 3

Proses:

- $10 \geq 3$: $10 - 3 = 7$ hasil = 1
- $7 \geq 3$: $7 - 3 = 4$ hasil = 2
- $4 \geq 3$: $4 - 3 = 1$ hasil = 3
- $1 < 3$: stop