

**LAPORAN PRAKTIKUM**  
**ALGORITMA DAN PEMROGRAMAN 1**  
**MODUL 12**  
**“WHILE-LOOP”**



**DISUSUN OLEH:**  
**DAFFA TSAQIFNA FAUZTSANY**  
**103112400032**  
**S1 IF-12-01**

**DOSEN:**  
**Yohani Setiya Rafika Nur, M. Kom.**

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**  
**FAKULTAS INFORMATIKA**  
**TELKOM UNIVERSITY PURWOKERTO**  
**2024/2025**

## DASAR TEORI:

### 1. Pengertian Perulangan:

- Perulangan adalah struktur kontrol yang memungkinkan eksekusi instruksi secara berulang-ulang berdasarkan kondisi tertentu.
- Perulangan dengan kondisi memungkinkan pengulangan dilakukan selama kondisi tertentu terpenuhi (bernilai true).

### 2. Karakteristik While-Loop:

- While-loop memerlukan dua elemen utama:
  - **Kondisi:** Operasi yang menghasilkan nilai boolean. Selama kondisi bernilai true, perulangan akan terus dilakukan.
  - **Aksi:** Instruksi yang dieksekusi berulang-ulang. Salah satu aksi harus mengubah kondisi menjadi false agar perulangan berhenti.
- Berbeda dengan for-loop, while-loop cocok digunakan ketika jumlah iterasi tidak diketahui sebelumnya.

### 3. Implementasi Dalam Bahasa Go:

- Bahasa Go menggunakan kata kunci `for` untuk mewakili while-loop. Struktur sintaksnya adalah:

```
for kondisi {  
    // aksi  
}
```

- Misalnya, program penjumlahan dua bilangan yang terus berjalan selama hasilnya genap:

```
for (a + b) % 2 == 0 {  
    fmt.Println("Hasil penjumlahan", a+b)  
}
```

### 4. Contoh Penerapan:

- Menampilkan hasil faktorial.
- Memvalidasi login dengan token.
- Menghasilkan deret Fibonacci.

### 5. Prinsip Penting:

- Perulangan harus memiliki kondisi terminasi untuk menghindari "infinite loop".

## CONTOH SOAL

Conso 1:

```
package main

import "fmt"

func main() {
    var n, j int
    fmt.Scan(&n)
    j = n
    for j > 1 {
        fmt.Print(j, " x ")
        j = j - 1
    }
    fmt.Println(1)
}
```

Output:

```
PS D:\test bs> go run 'd:\test bs\lab shit\9th meet\conso9\conso9-1.go'
0
1
PS D:\test bs> go run 'd:\test bs\lab shit\9th meet\conso9\conso9-1.go'
5
5 x 4 x 3 x 2 x 1
PS D:\test bs> go run 'd:\test bs\lab shit\9th meet\conso9\conso9-1.go'
10
10 x 9 x 8 x 7 x 6 x 5 x 4 x 3 x 2 x 1
PS D:\test bs> go run 'd:\test bs\lab shit\9th meet\conso9\conso9-1.go'
1
1
```

Deskripsi Program:

Program ini menerima masukan berupa bilangan bulat positif (**n**) dari pengguna dan mencetak faktorisasi bilangan tersebut dalam format penurunan bertahap. Program tidak menghitung nilai faktorial tetapi hanya menampilkan proses penguraian.

### 1. Deklarasi variabel:

```
var n, j int
```

- **n**: Input bilangan bulat positif dari pengguna.
- **j**: Variabel penunjuk yang digunakan untuk iterasi dalam proses pencetakan.

### 2. Input dari Pengguna:

```
fmt.Scan(&n)
```

- Membaca nilai **n** dari pengguna.

3. Inisialisasi dan Perulangan:

```
j = n
for j > 1 {
    fmt.Print(j, " x ")
    j = j - 1
}
fmt.Println(1)
```

- Variabel **j** diinisialisasi dengan nilai **n**.
- Selama **j > 1**, program mencetak nilai **j** diikuti dengan tanda "x".
- Setelah mencetak, **j** dikurangi 1 pada setiap iterasi.
- Ketika **j** bernilai 1, program keluar dari perulangan dan mencetak angka 1 sebagai akhir faktorisasi.

Contoh: Jika pengguna memasukkan

0 (input)

5 x 4 x 3 x 2 x 1 (output)

## 2.cosol2

```
package main
import "fmt"
func main() {
    var token string
    fmt.Scan(&token)
    for token != "12345abcde" {
        fmt.Scan(&token)
    }
    fmt.Println("Selamat Anda berhasil login")
}
```

Output:

```
PS D:\test bs> go run 'd:\test bs\lab shit\9th meet\conso9\conso9-2.go'
qwe12312
231234
13213
123lijwe
12345abcde
Selamat Anda berhasil login
PS D:\test bs> go run 'd:\test bs\lab shit\9th meet\conso9\conso9-2.go'
12345abcde
Selamat Anda berhasil login
```

Deskripsi Program:

Program ini adalah simulasi **validasi token login**. Program meminta pengguna memasukkan **token** hingga token yang benar (dalam kasus ini: 12345abcde) dimasukkan. Setelah token valid dimasukkan, program mencetak pesan sukses login.

### 1. Deklarasi variabel:

```
var token string
```

- **token:** Variabel string yang digunakan untuk menyimpan masukan token dari pengguna.

### 2. Input Token Awal:

```
fmt.Scan(&token)
```

- Program meminta pengguna untuk memasukkan token.

### 3. Perulangan Validasi Token:

```
for token != "12345abcde" {
    fmt.Scan(&token)
}
```

- Program terus meminta input token selama token yang dimasukkan **tidak sama** dengan 12345abcde.
- Jika token tidak valid, program meminta masukan token lagi.

4. Pesan Keberhasilan:

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

- Ketika token valid (12345abcde) dimasukkan, program keluar dari perulangan dan mencetak pesan sukses login.

Contoh: Jika pengguna memasukkan

```
12345abcde (input)
```

```
Selamat Anda berhasil login (output)
```

3.cosol3

```
package main
import "fmt"
func main() {
    var N, s1, s2, j, temp int
    fmt.Scan(&N)
    s1 = 0
    s2 = 1
    j = 0
    for j < N {
        fmt.Print(s1, " ")
        temp = s1 + s2
        s1 = s2
        s2 = temp
        j = j + 1
    }
}
```

Output:

```
PS D:\test bs> go run 'd:\test bs\lab shit\9th meet\conso9\conso9-3.go'
5
0 1 1 2 3
PS D:\test bs> go run 'd:\test bs\lab shit\9th meet\conso9\conso9-3.go'
2
0 1
PS D:\test bs> go run 'd:\test bs\lab shit\9th meet\conso9\conso9-3.go'
10
0 1 1 2 3 5 8 13 21 34
```

Deskripsi Program:

Program ini menghasilkan **deret Fibonacci** sebanyak **N bilangan pertama**. Deret Fibonacci adalah deret angka di mana setiap bilangan adalah hasil penjumlahan dari dua bilangan sebelumnya, dimulai dari 0 dan 1.

#### 1. Deklarasi variabel:

```
var N, s1, s2, j, temp int
```

- **N**: Jumlah bilangan Fibonacci yang ingin dicetak.
- **s1**: Bilangan pertama dari deret Fibonacci, dimulai dari 0.
- **s2**: Bilangan kedua dari deret Fibonacci, dimulai dari 1.

- **j**: Variabel penghitung untuk perulangan.
- **temp**: Variabel sementara untuk menyimpan nilai penjumlahan dua bilangan sebelumnya.

2. Input Jumlah Bilangan Fibonacci:

```
fmt.Scan(&N)
```

- Program meminta pengguna memasukkan jumlah bilangan Fibonacci yang ingin dicetak.

3. Inisialisasi Variabel Awal:

```
s1 = 0
s2 = 1
j = 0
```

- **s1** dimulai dari 0 dan **s2** dari 1, sesuai dengan aturan deret Fibonacci.

4. Perulangan untuk Menghasilkan Deret Fibonacci:

```
for j < N {
    fmt.Print(s1, " ")
    temp = s1 + s2
    s1 = s2
    s2 = temp
    j = j + 1
}
```

- Program mencetak **s1** (bilangan saat ini) pada setiap iterasi.
- Bilangan Fibonacci berikutnya dihitung sebagai penjumlahan dari **s1** dan **s2**, disimpan di **temp**.
- Nilai **s1** dan **s2** diperbarui untuk iterasi berikutnya.

Contoh: Jika pengguna memasukkan

```
5 (input)
```

```
0 1 1 2 3 (output)
```



## SOAL LATIHAN

### 1. Latihan1

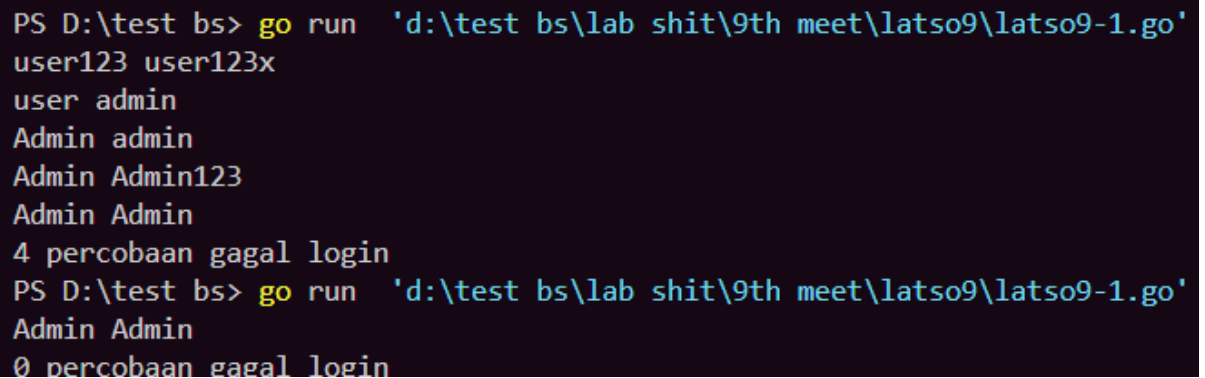
Source Code:

```
package main

import "fmt"

func main() {
    var use, pass string
    var x int
    var A bool
    for A == false {
        fmt.Scan(&use, &pass)
        if use == "Admin" && pass == "Admin" {
            A = true
        } else {
            x += 1
        }
    }
    fmt.Print(x, " percobaan gagal login")
}
```

Output:



```
PS D:\test bs> go run 'd:\test bs\lab shit\9th meet\latso9\latso9-1.go'
user123 user123x
user admin
Admin admin
Admin Admin123
Admin Admin
4 percobaan gagal login
PS D:\test bs> go run 'd:\test bs\lab shit\9th meet\latso9\latso9-1.go'
Admin Admin
0 percobaan gagal login
```

Deskripsi Program:

Program ini adalah **simulasi sistem login** sederhana yang menghitung jumlah percobaan login yang gagal sebelum pengguna berhasil login dengan username dan password yang benar.

#### 1. Deklarasi variabel:

```
var use, pass string
var x int
var A bool
```

- **use:** Untuk menyimpan input username.
- **pass:** Untuk menyimpan input password.
- **x:** Variabel penghitung jumlah percobaan login yang gagal.
- **A:** Flag (status) untuk menentukan apakah login berhasil.

2. Inisialisasi Perulangan:

```
for A == false {  
    ...  
}
```

- Perulangan berjalan terus selama nilai **A** adalah false, artinya login belum berhasil.

3. Input Username dan Password:

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

- Program meminta pengguna untuk memasukkan username (**use**) dan password (**pass**).

4. Pengecekan Kredensial:

```
if use == "Admin" && pass == "Admin" {  
    A = true  
} else {  
    x += 1  
}
```

- Jika **use** dan **pass** sama dengan "Admin", variabel **A** diatur ke true, dan login dianggap berhasil.
- Jika tidak cocok, nilai **x** (jumlah percobaan gagal) bertambah 1

5. Input Username dan Password:

```
fmt.Print(x, " percobaan gagal login")
```

- Setelah login berhasil, program mencetak jumlah percobaan login yang gagal.

Contoh: Jika pengguna memasukkan

User1 Pass1 (input)

User2 Pass2 (input)

Admin Admin (input)

2 percobaan gagal login (output)

## 2. Latihan2

### Source Code:

```
package main

import "fmt"

func main() {
    var x int
    fmt.Scan(&x)
    for x != 0 {
        fmt.Println(x % 10)
        x = x / 10
    }
}
```

Output:

```
PS D:\test bs> go run 'd:\test bs\lab shit\9th meet\latso9\latso9-2.go'
2
2
PS D:\test bs> go run 'd:\test bs\lab shit\9th meet\latso9\latso9-2.go'
2544
4
4
5
2
PS D:\test bs> go run 'd:\test bs\lab shit\9th meet\latso9\latso9-2.go'
3423554654
4
5
6
4
5
5
3
2
4
3
```

### Deskripsi Program:

Program ini bertujuan untuk mencetak digit dari sebuah bilangan bulat secara terbalik, satu per satu, dari digit paling kanan (digit terakhir) hingga digit paling kiri.

#### 1. Deklarasi variabel:

```
var x int
```

- Variabel x digunakan untuk menyimpan input bilangan yang akan diproses.

#### 2. Input:

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

- Program menerima input bilangan bulat dari pengguna dan menyimpannya dalam variabel x.
3. Looping untuk Mencetak Digit:

```
for x != 0 {  
    fmt.Println(x % 10)  
    x = x / 10  
}
```

- **x % 10:** Mengambil digit terakhir dari angka x. Operasi modulus (%) dengan 10 akan menghasilkan sisa bagi dari pembagian x dengan 10, yang merupakan digit terakhir dari bilangan tersebut.
- **x = x / 10:** Membagi x dengan 10 untuk menghilangkan digit terakhir dan menyisakan angka lainnya. Setelah digit terakhir dicetak, bilangan x diperbarui untuk memproses digit berikutnya.
- Proses akan terus berulang selama x tidak sama dengan 0. Begitu x menjadi 0, loop berhenti dan program selesai.

Contoh: Jika pengguna memasukkan

```
12345 (input)  
5 (output)  
4 (output)  
3 (output)  
2 (output)  
1 (output)
```

### 3. Latihan3

#### Source Code:

```
package main

import "fmt"

func main() {
    var x, y, z int
    fmt.Scan(&x, &y)
    for x >= y {
        x = x - y
        z++
    }
    fmt.Print(z)
}
```

Output:

```
PS D:\test bs> go run 'd:\test bs\lab shit\9th meet\latso9\latso9-3.go'
5 2
2
PS D:\test bs> go run 'd:\test bs\lab shit\9th meet\latso9\latso9-3.go'
10 7
1
PS D:\test bs> go run 'd:\test bs\lab shit\9th meet\latso9\latso9-3.go'
120 4
30
```

#### Deskripsi Program:

Program ini berfungsi untuk menghitung berapa kali bilangan y dapat dibagi ke dalam bilangan x tanpa melebihi nilai x. Ini sama seperti menghitung hasil pembagian bulat dari x oleh y (dengan cara pengurangan berulang).

##### 1. Deklarasi variabel:

```
var x, y, z int
```

- Variabel x dan y menyimpan input dari pengguna.
- Variabel z digunakan untuk menghitung berapa banyak y dapat dikurangkan dari x.

##### 2. Input:

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

- Program menerima dua input berupa bilangan bulat, x dan y, dari pengguna.

##### 3. Looping untuk Menghitung Pembagian:

```
for x >= y {
    x = x - y
    z++
}
```

- **for x >= y:** Loop ini akan terus berjalan selama x lebih besar atau sama dengan y.
- **x = x - y:** Setiap kali loop dijalankan, nilai y dikurangkan dari x.

- **z++:** Variabel z bertambah 1 setiap kali pengurangan dilakukan. Variabel ini digunakan untuk menghitung berapa kali y bisa dikurangkan dari x.

4. Output:

```
fmt.Print(z)
```

- Setelah loop selesai, program mencetak nilai z, yang merupakan hasil pembagian bulat dari x oleh y.

Contoh: Jika pengguna memasukkan

```
17 3 (input)
```

```
5 (output)
```