

**LAPORAN PRAKTIKUM ALGORITMA**

**DAN PEMROGRAMAN 1**

**MODUL 12 – WHILE-LOOP**

**ALGORITMA DAN PEMOGRAMAN 1**



**Disusun oleh:**

**NAMA : PRIMATAMA SIGALINGGING**

**NIM : 109082500076**

**S1IF-13-07**

**Asisten Praktikum**

**Adithana dharma putra**

**Apri pandu wicaksono**

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

#### Screenshot program

```
-go soal1.go
1 package main
2 import "fmt"
3 func main() {
4     var n, j int
5     fmt.Scan(&n)
6     j = n
7     for j > 1 {
8         fmt.Print(j, " x ")
9         j = j - 1
10    }
11    fmt.Println(1)
12 }
```

The screenshot shows a terminal window with the following output:

```
PS C:\alproo\go\minggu 12> go run guided1.go
masukan biangan : 0
1
PS C:\alproo\go\minggu 12> go run guided1.go
masukan biangan : 5
5 x 4 x 3 x 2 x 1
PS C:\alproo\go\minggu 12> go run guided1.go
masukan biangan : 10
10 x 9 x 8 x 7 x 6 x 5 x 4 x 3 x 2 x 1
PS C:\alproo\go\minggu 12>
```

## **Deskripsi program**

### ➤ **Tujuan**

Program ini dibuat untuk menghasilkan sebuah rangkaian angka menurun dimulai dari nilai yang dimasukkan oleh pengguna. Setiap angka ditampilkan dengan format khusus, yaitu angka tersebut diikuti tanda “x”. Melalui program ini, mahasiswa dapat memahami cara kerja perulangan dengan kondisi penurunan nilai secara bertahap, serta bagaimana sebuah pola output dapat dibentuk menggunakan loop.

### ➤ **Proses**

1. Program mendeklarasikan variabel n dan j sebagai bilangan bulat.
2. Pengguna diminta memasukkan sebuah angka. Angka tersebut disimpan ke dalam variabel n.
3. Nilai n kemudian disalin ke variabel j, yang akan digunakan sebagai penghitung dalam perulangan.
4. Program menjalankan sebuah for loop dengan kondisi  $j > 1$ . Selama kondisi terpenuhi:
  - Angka j dicetak, diikuti dengan spasi dan karakter “x”.
  - Nilai j dikurangi satu setiap iterasi untuk menghasilkan deret menurun.
5. Ketika perulangan selesai (saat j bernilai 1), angka terakhir yaitu 1 dicetak sebagai penutup pola.

### ➤ **Kesimpulan**

Program ini menampilkan pola deret menurun dengan format yang konsisten menggunakan logika perulangan sederhana. Melalui mekanisme decrement pada variabel penghitung, pengguna dapat memahami konsep iterasi menurun dan bagaimana menata format output menggunakan print statement.

## **2. Guided 2**

### **Source Code**

```
package main

import "fmt"

func main() {
    var token string
    fmt.Scan(&token)

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

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

## Screenshot program

The screenshot shows a Go development environment. On the left, the code editor displays `guided2.go` with the following content:

```
1 package main
2 import "fmt"
3 func main() {
4     var token string
5     fmt.Scan(&token)
6     for token != "12345abcde" {
7         fmt.Scan(&token)
8     }
9     fmt.Println("Selamat Anda berhasil login")
10 }
```

On the right, there are two windows. The top window is a terminal window showing the output of running the program:

```
PS C:\alproo\go\minggu 12> go run guided2.go
12345abcde
Selamat Anda berhasil login
PS C:\alproo\go\minggu 12>
```

The bottom window is another terminal window showing the user's input:

```
Nama : Primatama Sigalingging
NIM: 109082500076
```

At the bottom of the interface, there are tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS.

## Deskripsi program

### ➤ Tujuan

Program ini bertujuan melakukan pengecekan token login yang dimasukkan pengguna. Validasi dilakukan dengan cara membandingkan input pengguna terhadap token tetap yang telah ditentukan. Program ini menjadi contoh dasar dari sistem autentikasi yang berulang hingga pengguna memasukkan data yang benar.

### ➤ Proses

1. Program mendeklarasikan sebuah variabel string bernama token.
2. Pengguna diminta memasukkan token pertama kali menggunakan `fmt.Scan`.
3. Program memasuki perulangan dengan kondisi `token != "12345abcde"`. Selama token tidak cocok:
  - Program meminta pengguna memasukkan input ulang.
  - Program terus mengulang proses ini hingga input sesuai dengan token yang ditentukan.
4. Jika pengguna berhasil mengetik token yang benar ("12345abcde"), perulangan berhenti.
5. Terakhir, program mencetak pesan bahwa pengguna berhasil login.

### ➤ Kesimpulan

Program ini menunjukkan prinsip dasar validasi input dengan perulangan. Dengan meminta pengguna mengulang input hingga benar, program ini merepresentasikan mekanisme login sederhana yang mendemonstrasikan kombinasi antara pengecekan kondisi dan loop.

### 3. Guided 3

#### Source Code

```
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.Println(s1, " ")
        temp = s1 + s2
        s1 = s2
        s2 = temp
        j = j + 1
    }
}
```

#### Screenshot program

The screenshot shows a Go development environment with two main panes. The left pane displays the source code for `guided3.go`, which contains a simple implementation of the Fibonacci sequence. The right pane shows a terminal window with the output of the program, demonstrating its functionality.

`guided3.go` content:

```
1 package main
2
3 func main() {
4     s1 = 0
5     s2 = 1
6     j = 0
7
8     for j < N {
9         fmt.Println(s1, " ")
10        temp = s1 + s2
11        s1 = s2
12        s2 = temp
13        j = j + 1
14    }
15 }
```

Terminal output:

```
PS C:\alproo\go\minggu 12> go run guided3.go
5
0 1 1 2 3
PS C:\alproo\go\minggu 12> go run guided3.go
2
0 1
PS C:\alproo\go\minggu 12> go run guided3.go
10
0 1 1 2 3 5 8 13 21 34
PS C:\alproo\go\minggu 12>
```

## **Deskripsi program**

### ➤ **Tujuan**

Program ini dibuat untuk menghasilkan deret Fibonacci sebanyak N angka sesuai input pengguna. Program ini berguna untuk memahami konsep perulangan, pembaruan nilai variabel secara berurutan, dan metode perhitungan pola matematis yang umum digunakan dalam algoritma dasar.

### ➤ **Proses**

1. Program mendeklarasikan variabel N, s1, s2, j, dan temp.
2. Pengguna memasukkan nilai N yang menentukan berapa banyak angka Fibonacci yang ingin ditampilkan.
3. Dua nilai pertama Fibonacci diinisialisasi: s1 = 0 dan s2 = 1.
4. Program memulai perulangan dengan kondisi  $j < N$ . Pada setiap iterasi:
  - Program menampilkan nilai s1.
  - Nilai Fibonacci berikutnya dihitung dengan  $\text{temp} = s1 + s2$ .
  - Variabel s1 digeser menjadi s2, dan s2 digeser menjadi temp. Hal ini membuat dua variabel tersebut selalu menyimpan angka terakhir dan sebelum terakhir dalam deret.
  - Nilai penghitung j dinaikkan satu per satu.
5. Proses ini berlangsung hingga jumlah angka yang ditampilkan sesuai dengan nilai N.

### ➤ **Kesimpulan**

Program ini menggambarkan cara memproduksi deret Fibonacci dengan memanfaatkan dua variabel yang diperbarui secara berulang. Pendekatan ini memperlihatkan konsep iteratif yang lebih efisien dibanding metode rekursif untuk skala dasar, sekaligus membantu memahami pola matematis yang umum dipelajari dalam struktur data dan algoritma.

## TUGAS

### 1. Tugas 1

#### Source code

```
package main

import (
    "fmt"
)

func main() {
    var username, password string
    var gagal int
    gagal = 0

    for {
        fmt.Scan(&username, &password)

        if username == "Admin" && password == "Admin" {
            break
        } else {
            gagal++
        }
    }

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

## Screenshot program

The screenshot shows a Go development environment. On the left, the code editor displays a file named 'soal1.go' with the following content:

```
1go soal1.go
2 import (
3 )
4 func main() {
5     var username, password string
6     var gagal int
7     gagal = 0
8     for {
9         fmt.Scan(&username, &password)
10        if username == "Admin" && password == "Admin" {
11            break
12        } else {
13            gagal++
14        }
15    }
16    fmt.Printf("%d percobaan gagal login\n", gagal)
17 }
18 }
```

Below the code editor are tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is active, showing the command 'go run soal1.go' and its output:

```
PS C:\alproo\go\minggu 12> go run soal1.go
Admin Admin
0 percobaan gagal login
PS C:\alproo\go\minggu 12>
```

To the right of the terminal is a browser window displaying a simple form with two input fields and a submit button. The browser's status bar shows 'Ln 1, Col 30 | 48 character| Plain t | 100% | Wind | UTF-8'.

## Deskripsi program

### ➤ Tujuan

Program ini dirancang untuk mensimulasikan proses login yang tidak hanya memeriksa kecocokan username dan password, tetapi juga menghitung berapa kali pengguna salah memasukkan data. Program ini membantu memahami bagaimana menggabungkan perulangan tak terbatas, percabangan, dan variabel penghitung dalam satu alur sistem login.

### ➤ Proses

1. Program mendefinisikan dua variabel string (username dan password) serta satu variabel penghitung gagal yang diset awal ke 0.
2. Program memasuki perulangan tak terbatas menggunakan for {}.
3. Di dalam perulangan:
  - Pengguna diminta memasukkan username dan password melalui layar.
  - Program mengecek apakah input tersebut cocok dengan username dan password yang ditentukan, yaitu "Admin" untuk keduanya.
4. Jika kedua input benar:
  - Program keluar dari perulangan menggunakan break.
5. Jika input salah:
  - Variabel gagal ditambah 1 sebagai penanda satu kali percobaan gagal.
  - Perulangan dilanjutkan kembali hingga pengguna memasukkan data yang benar.

6. Setelah berhasil login, program menampilkan jumlah percobaan gagal yang sudah dilakukan pengguna sebelum berhasil masuk.

➤ **Kesimpulan**

Program ini memberikan ilustrasi nyata bagaimana menghitung jumlah kesalahan login dan bagaimana perulangan dapat digunakan untuk mengatur interaksi berulang. Dengan mengombinasikan logika if-else dan penghitung iterasi, program menghasilkan sistem login sederhana namun fungsional yang dapat memperlihatkan performa input pengguna.

## 2. Tugas 2

### Source code

```
package main

import "fmt"

func main() {
    var n int
    fmt.Print("Masukkan bilangan: ")
    fmt.Scan(&n)

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

## Screenshot program

```
C:\alproo\go\minggu 12\soal3.go
1 package main
2 import "fmt"
3 func main() {
4     var n int
5     fmt.Print("Masukkan bilangan: ")
6     fmt.Scan(&n)
7
8     for n > 0 {
9         digit := n % 10
10        fmt.Println(digit)
11        n = n / 10
12    }
13}
14
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\alproo\go\minggu 12> go run soal2.go
Masukkan bilangan: 2
2
PS C:\alproo\go\minggu 12> go run soal2.go
Masukkan bilangan: 2544
4
4
5
2
PS C:\alproo\go\minggu 12> go run soal2.go
Masukkan bilangan: 3423554654
```

```
Nama : Primatama Sigalingging
NIM: 109082500076
```

Ln 1, Col 30 | 48 caracte Plain t | 100% | Wind | UTF-8

## Deskripsi program

### ➤ Tujuan

Program ini dibuat untuk memecah sebuah bilangan menjadi digit-digit pembentuknya, dimulai dari digit paling kanan (digit satuan). Program ini melatih penggunaan operator modulo dan pembagian bilangan bulat, yang merupakan teknik penting untuk pengolahan angka.

### ➤ Proses

1. Program mendeklarasikan variabel `n` yang digunakan untuk menyimpan input bilangan.
2. Pengguna memasukkan sebuah bilangan bulat.
3. Program menjalankan for loop dengan kondisi `n > 0`. Selama kondisi ini terpenuhi:
  - Digit terakhir dihitung melalui `n % 10`.
  - Digit tersebut ditampilkan ke layar.
  - Nilai `n` dikurangi digit terakhirnya dengan cara `n = n / 10` (bilangan dibagi tanpa desimal).
4. Perulangan berhenti ketika `n` habis terbagi hingga mencapai nilai nol.

### ➤ Kesimpulan

Program ini menunjukkan bagaimana sebuah bilangan dapat diolah per digit menggunakan operasi dasar matematika. Teknik ini sering digunakan dalam banyak problem seperti pembalikan angka, perhitungan checksum, validasi digit, dan manipulasi numerik lainnya.

### 3. Tugas 3

#### Source code

```
package main
import "fmt"
func main() {
    var x, y int
    fmt.Print("Masukkan dua bilangan (x y): ")
    fmt.Scan(&x, &y)
    h := 0
    for x >= y {
        x = x - y
        h++
    }
    fmt.Println("Hasil:", h)
}
```

#### Screenshot program

The screenshot shows a terminal window with the following content:

```
-go soal3.go
1 package main
2 import "fmt"
3 func main() {
4     var x, y int
5     fmt.Print("Masukkan dua bilangan (x y): ")
6     fmt.Scan(&x, &y)
7     h := 0
8     for x >= y {
9         x = x - y
10        h++
11    }
12    fmt.Println("Hasil:", h)
13 }
14

Output (Ctrl+Shift+U)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
```

PS C:\alproo\go\minggu 12> go run soal3.go  
Masukkan dua bilangan (x y): 5 2  
Hasil: 2  
PS C:\alproo\go\minggu 12> go run soal3.go  
Masukkan dua bilangan (x y): 10 7  
Hasil: 1

In the terminal output, the user inputs '5 2' and '10 7' respectively, and the program outputs 'Hasil: 2' and 'Hasil: 1'.

On the right side of the terminal window, there is a text input field containing:

Nama : Primatama Sigalingging  
NIM: 109082500076

Below the terminal, status bar text includes: Ln 1, Col 30 | 48 character | Plain text | 100% | Wind | UTF-8

## **Deskripsi program**

### ➤ **Tujuan**

Program ini dibuat untuk menghitung hasil pembagian dua bilangan bulat tanpa menggunakan operator pembagian (/). Sebagai gantinya, program menggunakan teknik pengurangan berulang (repeated subtraction) untuk menentukan berapa kali penyebut dapat dikurangkan dari pembilang. Program ini melatih penggunaan perulangan (loop), operasi aritmatika dasar, serta logika kontrol alur dalam bahasa Go.

### ➤ **Proses**

1. Program mendeklarasikan variabel x dan y sebagai input dari pengguna, serta variabel h yang digunakan untuk menyimpan hasil pembagian dalam bentuk bilangan bulat.
2. Pengguna memasukkan dua bilangan bulat, yaitu nilai x (pembilang) dan y (penyebut)
3. Program menjalankan sebuah for loop dengan kondisi  $x \geq y$ . Selama kondisi ini terpenuhi:
  - Nilai x dikurangi dengan y menggunakan operasi  $x = x - y$ .
  - Setiap kali pengurangan terjadi, variabel h ditambah satu ( $h++$ ), menandakan bahwa penyebut dapat dikurangkan satu kali lagi dari pembilang.
4. Perulangan berhenti ketika nilai x menjadi lebih kecil dari y, yang berarti penyebut sudah tidak bisa dikurangkan lagi dari pembilang.
5. Setelah perulangan selesai, program menampilkan nilai h sebagai hasil pembagian bilangan bulat.

### ➤ **Kesimpulan**

Program ini menunjukkan bagaimana operasi pembagian dapat disimulasikan dengan menggunakan pengurangan berulang. Teknik ini berguna untuk memahami konsep fundamental pembagian, serta sering digunakan dalam algoritma dasar, implementasi pembagian manual pada sistem sederhana, dan pembuatan fungsi matematika tanpa operator bawaan.