LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 1
MODUL 5&6
“FOR-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 (Looping)

Perulangan adalah mekanisme dalam pemrograman yang memungkinkan instruksi dieksekusi secara berulang selama kondisi tertentu terpenuhi. Tujuan utama dari penggunaan perulangan adalah untuk menghindari penulisan instruksi berulang secara manual, sehingga program menjadi lebih efisien dan mudah dipelihara.

2. Paradigma Perulangan dalam Pemrograman

Ada berbagai jenis struktur perulangan, namun yang ditekankan di modul ini adalah **for-loop**. For-loop merupakan salah satu jenis perulangan yang iterasinya diatur berdasarkan batas yang jelas, seperti jumlah tertentu atau kondisi tertentu. Misalnya, jika kita perlu menampilkan teks atau melakukan tugas berulang 1000 kali, for-loop digunakan agar kita tidak perlu menulis instruksi yang sama sebanyak 1000 kali secara manual.

3. Komponen For-Loop

For-loop terdiri dari tiga komponen utama:

- **Inisialisasi**: Digunakan untuk mengatur nilai awal dari variabel iterasi, yang biasanya merupakan variabel bertipe integer.
- **Kondisi**: Mengevaluasi apakah perulangan masih harus dilanjutkan. Jika kondisinya benar, instruksi akan terus dijalankan.
- **Update**: Melakukan perubahan pada variabel iterasi di setiap akhir iterasi, sehingga perulangan dapat bergerak menuju kondisi penghentian.

4. Implementasi dalam Bahasa Go

Modul ini menjelaskan bahwa for-loop dapat digunakan dengan sangat mudah dalam bahasa Go, dengan struktur sebagai berikut:

go

Copy code

```
for inisialisasi; kondisi; update {
    // instruksi yang diulang
}
```

Sebagai contoh, untuk menampilkan teks sebanyak 1000 kali dalam bahasa Go, for-loop digunakan untuk mengatur iterasi dengan inisialisasi variabel iterasi, kondisi iterasi kurang dari atau sama dengan 1000, dan update variabel di setiap langkah.

5. Contoh Implementasi

Dalam file ini juga disertakan beberapa contoh program yang menggunakan for-loop:

- Menampilkan bilangan antara dua angka (contoh: menampilkan bilangan dari 2 hingga 5).
- Menghitung luas segitiga untuk beberapa input.
- Mengalikan dua bilangan tanpa menggunakan operator kali.

6. Pentingnya Pemahaman Struktur For-Loop

Pemahaman yang baik mengenai for-loop sangat penting untuk membuat program lebih efisien dan menghindari kesalahan seperti perulangan tak terbatas, yang bisa menyebabkan program berjalan terus tanpa henti.

CONTOH SOAL

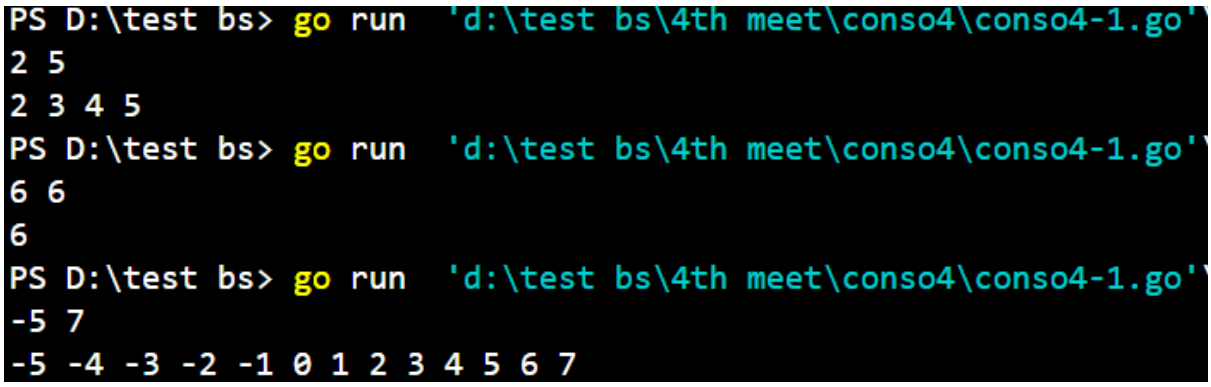
Conso 1:

```
package main

import "fmt"

func main() {
    var a, b int
    var j int
    fmt.Scan(&a, &b)
    for j = a; j <= b; j += 1 {
        fmt.Print(j, " ")
    }
}
```

Output:



Deskripsi Program:

Program Go di atas digunakan untuk mencetak bilangan bulat dari nilai awal (a) hingga nilai akhir (b) secara berurutan dengan jarak 1 antar bilangan. Berikut adalah penjelasan rinci mengenai program tersebut:

1. **Deklarasi variabel:**

```
var a, b int
var j int
```

- **a**: Menyimpan nilai awal.
- **b**: Menyimpan nilai akhir.
- **j**: Variabel untuk iterasi dalam perulangan.

2. Membaca input dari pengguna:

```
fmt.Scan(&a, &b)
```

Program membaca dua nilai yang dimasukkan oleh pengguna:

- **a**: Nilai awal, di mana perulangan akan dimulai.
- **b**: Nilai akhir, di mana perulangan akan berhenti.

3. Perulangan for:

```
for j = a; j <= b; j += 1 {
        fmt.Print(j, " ")
}
```

- **for j = a; j <= b; j += 1**: Struktur perulangan ini menginisialisasi variabel j dengan nilai a, kemudian memeriksa apakah j kurang dari atau sama dengan b. Jika benar, maka blok kode di dalam perulangan akan dijalankan, dan setiap iterasi akan menambah nilai j dengan 1.
- Setiap iterasi mencetak nilai j, diikuti dengan spasi.

4. Menampilkan hasil:

```
fmt.Print(j, " ")
```

- Program mencetak nilai variabel j pada setiap iterasi, dengan spasi di antara angka-angka yang dicetak.

Contoh: Jika pengguna memasukkan

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

2.cosol2

```
package main
import "fmt"
func main() {
   var j, alas, tinggi, n int
   var luas float64
   fmt.Scan(&n)
   for j = 1; j <= n; j += 1 {
      fmt.Scan(&alas, &tinggi)
      luas = 0.5 * float64(alas*tinggi)
      fmt.Println(luas)
   }
}
```

Output:

```
PS D:\test bs> go run  'd:\test bs\4th meet\conso4\conso4-2.go'
5
11 2
11
32 14
224
6 2
6
15 15
112.5
20 35
350
PS D:\test bs> go run  'd:\test bs\4th meet\conso4\conso4-2.go'
3
12 32
192
231 234
27027
43 34
731
```

Deskripsi Program:

Program Go di atas digunakan untuk menghitung dan mencetak luas beberapa segitiga berdasarkan input dari pengguna. Program menerima banyak pasangan nilai alas dan tinggi, lalu menghitung luas masing-masing segitiga menggunakan rumus luas segitiga. Berikut adalah penjelasan rinci mengenai program tersebut.

1. **Deklarasi variabel:**

```
var j, alas, tinggi, n int
var luas float64
```

- **j**: Variabel untuk iterasi dalam perulangan.
- **alas**: Menyimpan nilai alas dari segitiga.
- **tinggi**: Menyimpan nilai tinggi dari segitiga.
- **n**: Jumlah segitiga yang akan dihitung.
- **luas**: Menyimpan nilai luas dari segitiga (tipe float64 untuk hasil desimal).

2. Membaca input dari pengguna:

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

- Program meminta pengguna memasukkan nilai **n**, yang merupakan jumlah segitiga yang ingin dihitung luasnya.

3. Perulangan for:

```
for j = 1; j <= n; j += 1 {
        fmt.Scan(&alas, &tinggi)
        luas = 0.5 * float64(alas*tinggi)
        fmt.Println(luas)
```

```
}
```

- Program melakukan perulangan sebanyak **n** kali, sesuai jumlah segitiga yang ingin dihitung luasnya.
- Pada setiap iterasi:

  - Program meminta pengguna memasukkan dua nilai: **alas** dan **tinggi** segitiga.
  - Menghitung luas segitiga dengan rumus: luas=0.5×alas×tinggi\text{luas} = 0.5 \times \text{alas} \times \text{tinggi}luas=0.5×alas×tinggi
    - Nilai alas dan tinggi dikalikan terlebih dahulu, kemudian hasilnya dikalikan dengan 0.5 untuk menghitung luas segitiga.
    - `float64(alas*tinggi)` digunakan untuk memastikan hasil perkalian memiliki tipe `float64` sehingga operasi perhitungan desimal dapat dilakukan dengan benar.
  - Program mencetak hasil luas yang sudah dihitung.

  4. Menampilkan hasil:

```
fmt.Println(luas)
```

- Setiap kali perhitungan selesai, program mencetak luas segitiga.

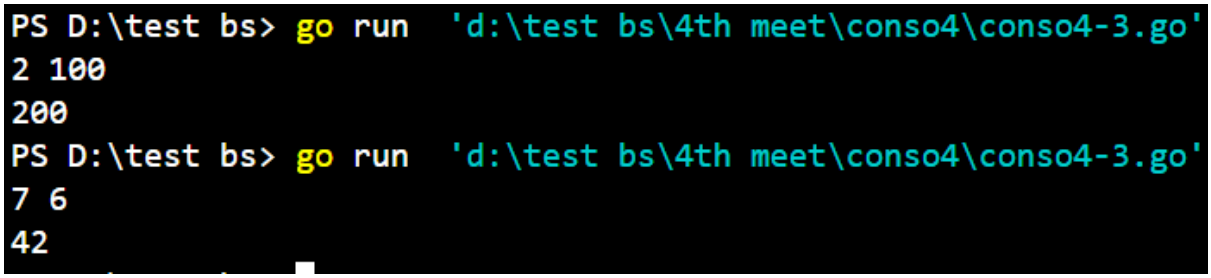
Contoh: Jika pengguna memasukkan

```
2(input)
3 4(input)
5 6(input)
6.0(output)
15.0(output)
```

3.cosol3

```
package main
import "fmt"
func main() {
   var j, v1, v2 int
   var hasil int
   fmt.Scan(&v1, &v2)
   hasil = 0
   for j = 1; j <= v2; j += 1 {
      hasil = hasil + v1
   }
   fmt.Println(hasil)
}
```

Output:



Deskripsi Program:

Program Go di atas digunakan untuk menghitung hasil perkalian antara dua bilangan bulat menggunakan perulangan for. Daripada menggunakan operator perkalian langsung (*), program ini menghitung hasil perkalian dengan cara menambah bilangan pertama (v1) sebanyak nilai bilangan kedua (v2) kali. Berikut penjelasan rinci mengenai setiap bagian dari program:

1. **Deklarasi variabel:**

```
var j, v1, v2 int
var hasil int
```

- **j**: Variabel yang digunakan untuk iterasi dalam perulangan.
- **v1**: Bilangan pertama yang akan dikalikan.
- **v2**: Bilangan kedua yang merupakan berapa kali v1 akan dijumlahkan.
- **hasil**: Menyimpan hasil dari operasi perkalian (melalui penjumlahan berulang).

2. Membaca input dari pengguna:

```
fmt.Scan(&v1, &v2)
```

Program meminta pengguna untuk memasukkan dua bilangan bulat:

- **v1**: Bilangan yang akan dikalikan.
- **v2**: Bilangan yang menentukan berapa kali `v1` akan dijumlahkan.

3. Inisialisasi hasil:

```
hasil = 0
```

- Variabel hasil diinisialisasi dengan nilai 0. Ini adalah nilai awal sebelum penjumlahan dilakukan.

4. Perulangan for:

```
for j = 1; j <= v2; j += 1 {
        hasil = hasil + v1
}
```

- Perulangan dilakukan sebanyak **v2** kali.
- Pada setiap iterasi, v1 ditambahkan ke dalam variabel hasil. Jadi, program melakukan penjumlahan v1 secara berulang hingga v2 kali.
- Dengan kata lain, ini adalah cara untuk melakukan perkalian v1 * v2 melalui penjumlahan.

5. Menampilkan hasil:

```
fmt.Println(hasil)
```

- Setelah perulangan selesai, program mencetak nilai akhir dari hasil, yang merupakan hasil dari perkalian v1 * v2.

Contoh: jika pengguna memasukkan:

3 4(input) 12(output)

SOAL LATIHAN

1. Latihan1

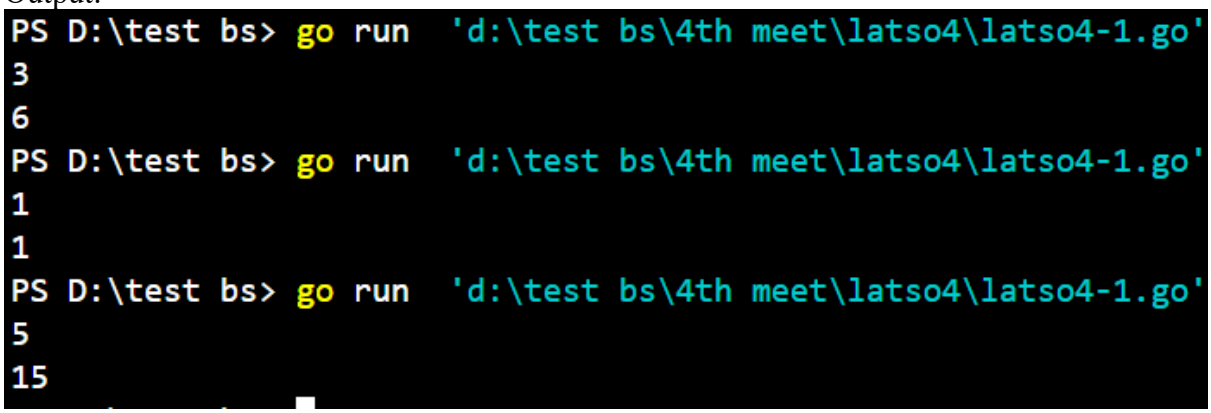
Source Code:

```
package main

import "fmt"

func main() {
    var x, i, y int
    fmt.Scan(&x)
    for i = 1; i <= x; i++ {
        y = y + i
    }
    fmt.Print(y)
}
```

Output:



Deskripsi Program:
Program Go di atas digunakan untuk menghitung jumlah total dari semua bilangan bulat dari 1 hingga x. Program ini menggunakan perulangan untuk menjumlahkan setiap bilangan dari 1 hingga x dan mencetak hasil penjumlahan tersebut. Berikut adalah penjelasan rinci mengenai program tersebut:

1. **Deklarasi variabel:**

```
var x, i, y int
```

- **x**: Menyimpan batas atas bilangan hingga mana penjumlahan akan dilakukan. Ini adalah input dari pengguna.
- **i**: Variabel untuk iterasi dalam perulangan.
- **y**: Menyimpan hasil penjumlahan. Diinisialisasi dengan nilai 0 secara implisit karena tipe datanya int.

2. Membaca input dari pengguna:

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

- Program meminta pengguna memasukkan nilai **x**, yaitu bilangan bulat hingga mana program akan melakukan penjumlahan dari 1 sampai x.

3. Perulangan for:

```
for i = 1; i <= x; i++ {
        y = y + i
}
```

- Program melakukan perulangan mulai dari i = 1 hingga i <= x.
- Pada setiap iterasi, nilai variabel y diperbarui dengan menambahkan nilai i ke y. Jadi, y adalah akumulator yang menampung hasil penjumlahan dari semua bilangan dari 1 hingga x.

4. Menampilkan hasil:

```
fmt.Print(y)
```

- Setelah perulangan selesai, program mencetak nilai akhir dari y, yang merupakan hasil penjumlahan dari semua bilangan dari 1 hingga x.

Contoh: Jika pengguna memasukkan

```
5(input)

15(output)
```

2. .Latihan2

Source Code:

```
package main

import "fmt"

func main() {
var r, phi, t, l, i, x float64
   phi = 3.14159265359
   fmt.Scan(&x)
   for i = 0; i < x; i++ {
      fmt.Scan(&r, &t)
      l = (phi * r * r * t) / 3
      fmt.Print(l)
   }
}
```

Output:

```
PS D:\test bs> go run  'd:\test bs\4th meet\latso4\latso4-2.go'
1
3 4
37.699111843080004
PS D:\test bs> go run  'd:\test bs\4th meet\latso4\latso4-2.go'
3
1 1
1.0471975511966667
2 2
8.377580409573333
3 3
28.274333882310003
```

Deskripsi Program:
Program Go di atas digunakan untuk menghitung volume beberapa kerucut berdasarkan input dari pengguna. Setiap kerucut memiliki jari-jari (r) dan tinggi (t), dan program menghitung volumenya menggunakan rumus volume kerucut. Berikut adalah penjelasan rinci program ini:

1. **Deklarasi variabel:**

```
var r, phi, t, l, i, x float64
```

- **r**: Jari-jari alas kerucut.
- **phi**: Nilai konstanta π\piπ (3.14159265359).
- **t**: Tinggi kerucut.
- **l**: Volume kerucut yang akan dihitung.
- **i**: Variabel iterasi untuk perulangan.
- **x**: Jumlah kerucut yang volumenya akan dihitung.

2. Inisialisasi nilai phi:

```
phi = 3.14159265359
```

- Konstanta nilai π\piπ digunakan dalam perhitungan volume kerucut

3. Membaca input dari pengguna:

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

- Program meminta pengguna memasukkan nilai **x**, yang mewakili jumlah kerucut yang akan dihitung volumenya.

4. Perulangan for:

```
for i = 0; i < x; i++ {
        fmt.Scan(&r, &t)
        l = (phi * r * r * t) / 3
        fmt.Print(l)
}
```

- Program melakukan perulangan sebanyak **x** kali (jumlah kerucut).
- Pada setiap iterasi:

- Program meminta pengguna memasukkan nilai **r** (jari-jari) dan **t** (tinggi) untuk kerucut tersebut.
- Volume kerucut dihitung dengan rumus:l=(π*r^2*t)/3
    - r^2 adalah hasil kuadrat dari jari-jari.
    - Tinggi **t** dikalikan dengan hasil kuadrat jari-jari, lalu dikalikan dengan π\piπ, dan hasilnya dibagi 3.
- Hasil volume kerucut dicetak oleh program.

5. Menampilkan hasil:

```
fmt.Print(l)
```

- Program mencetak hasil volume kerucut pada setiap iterasi.

Contoh: Jika pengguna memasukkan:

```
2(input)
3 5(input)
4 6(input)
47.12389(output)
100.53096(output)
```

3. Latihan3

Source Code:

```
package main

import "fmt"

func main() {
   var x, y, z, i int
   z = 1
   fmt.Scan(&x, &y)
   for i = 0; i < y; i++ {
      z = z * x
   }
   fmt.Print(z)
}
```

Output:

```
PS D:\test bs> go run  'd:\test bs\4th meet\latso4\latso4-3.go'
4 2
16
PS D:\test bs> go run  'd:\test bs\4th meet\latso4\latso4-3.go'
2 10
1024
PS D:\test bs> go run  'd:\test bs\4th meet\latso4\latso4-3.go'
10 3
1000
```

Deskripsi Program:

Program Go di atas digunakan untuk menghitung **pangkat** dari suatu bilangan x yang dipangkatkan dengan y. Dengan kata lain, program ini menghitung xyx^yxy (x pangkat y). Berikut adalah penjelasan rinci mengenai program tersebut:

1. **Deklarasi variabel:**

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

- **x**: Bilangan dasar yang akan dipangkatkan (input dari pengguna).
- **y**: Eksponen atau pangkat (input dari pengguna).
- **z**: Hasil dari perhitungan pangkat xyx^yxy. Diinisialisasi dengan nilai 1 karena setiap bilangan yang dipangkatkan ke 0 sama dengan 1.
- **i**: Variabel iterasi untuk perulangan.

2. **Inisialisasi nilai z**:

```
z = 1
```

- **z** diinisialisasi dengan 1 karena setiap perhitungan pangkat dimulai dengan 1.

3. Membaca input dari pengguna:

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

Program meminta pengguna memasukkan dua bilangan:

- **x**: Bilangan yang akan dipangkatkan.
- **y**: Pangkat yang akan digunakan.

4. Perulangan for:

```
for i = 0; i < x; i++ {
        fmt.Scan(&r, &t)
        l = (phi * r * r * t) / 3
        fmt.Print(l)
}
```

- Perulangan ini berjalan sebanyak **y** kali, yang berarti bilangan **x** akan dikalikan dengan dirinya sendiri sebanyak **y** kali.
- Setiap kali perulangan berjalan, nilai **z** diperbarui menjadi hasil perkalian dari **z** dengan **x**.

- Misalnya, jika **x** adalah 3 dan **y** adalah 4, maka perulangan berjalan seperti ini:
  - Iterasi 1: Z = 1 X 3 = 3

- Iterasi 2: Z = 3 X 3 = 9
- Iterasi 3: Z = 9 X 3 = 27
- Iterasi 4: Z = 27 X 3 = 81

- 5. Menampilkan hasil:

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

- Setelah perulangan selesai, program mencetak hasil akhir **z**, yang merupakan hasil dari perhitungan xy (x pangkat y).

Contoh: Jika pengguna memasukkan:

```
3 4(input)

81(output)
```

4. Latihan4

Source Code:

```
package main

import "fmt"

func main() {
   var x, y, i int
   fmt.Scan(&x)
   y = 1
   for i = 1; i <= x; i++ {
      y = y * i
   }
   fmt.Print(y)
}
```

Output:

```
PS D:\test bs> go run  'd:\test bs\4th meet\latso4\latso4-4.go'
0
1
PS D:\test bs> go run  'd:\test bs\4th meet\latso4\latso4-4.go'
1
1
PS D:\test bs> go run  'd:\test bs\4th meet\latso4\latso4-4.go'
5
120
PS D:\test bs> go run  'd:\test bs\4th meet\latso4\latso4-4.go'
10
3628800
```

Deskripsi Program:

Program Go di atas digunakan untuk menghitung **faktorial** dari bilangan bulat positif yang dimasukkan oleh pengguna. Berikut adalah penjelasan rinci mengenai program ini:

1. **Deklarasi variabel:**

```
var x, y, i int
```

- **x**: Menyimpan nilai input dari pengguna, yaitu bilangan yang faktorialnya akan dihitung.
- **y**: Menyimpan hasil dari faktorial. Diinisialisasi dengan nilai 1, karena faktorial selalu dimulai dari 1.
- **i**: Variabel iterasi untuk perulangan.

2. Membaca input dari pengguna:

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

- Program meminta pengguna memasukkan sebuah bilangan bulat **x**, yang faktorialnya akan dihitung

3. Perulangan for:

```
for i = 1; i <= x; i++ {
        y = y * i
}
```

- Program melakukan perulangan dari 1 hingga **x**.
- Pada setiap iterasi, nilai **y** diperbarui dengan mengalikan **y** dengan nilai iterasi **i**. Ini menghitung faktorial secara bertahap.

- Misalnya, jika **x** adalah 5, maka perhitungan dalam perulangan adalah:
  - Iterasi 1: y = 1 x 1 = 1
  - Iterasi 2: y = 1 x 2 = 2
  - Iterasi 3: y = 2 x 3 = 6
  - Iterasi 4: y = 6 x 4 = 24
  - Iterasi 5: y = 24 x 5 = 120
-

4. Menampilkan hasil:

```
fmt.Print(y)
```

- Setelah perulangan selesai, program mencetak hasil faktorial **y**, yaitu hasil dari perkalian semua bilangan dari 1 hingga **x**.

Contoh: Jika pengguna memasukkan

```
5(input)

120(output)
```