LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 1
MODUL V & VI
“FOR-LOOP”

DISUSUN OLEH:

FEROS PEDROSA VALENTINO

103112400055

S1 IF-12-01

DOSEN:

Yohani Setiya Rafika Nur, M. Kom.

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

PURWOKERTO

2024/2025

DASAR TEORI

Perulangan

- Tugas utama komputer adalah melakukan suatu instruksi/proses secara berulang dan terus-menerus tanpa adanya perbedaan.
- Hal ini berbeda dengan manusia yang bila melakukan hal sama secara berulang bisa melakukan kesalahan.
- Setiap baris instruksi dieksekusi satu persatu.
- Satu atau lebih instruksi bisa dieksekusi berulang kali.

Syarat Perulangan:

- Perulangan harus berhenti
- Apabila perulangan tidak pernah berhenti, maka algoritmanya salah (proses lama ≠ tidak pernah berhenti)
- Pemrogram harus mengetahui perulangan akan berhenti/tidak sebelum program dijalankan

Jenis Instruksi Perulangan:

- Berdasarkan jumlah iterasi
- Berdasarkan kondisi (kapan harus diulangi/berhenti)

Perulangan berdasarkan Iterasi:

- Untuk menyelesaikan kasus dengan jumlah iterasi diketahui di awal
- Iterasi adalah variabel bertipe integer untuk menampung iterasi perulangan
- Variabel i, init dan end adalah variabel atau nilai bertipe integer
- Aksi dieksekusi berulang sebanyak end init +1 kali (iterasi)
- Setiap aksi dieksekusi, nilai dari variabel iterasi selalu bertambah 1, dari init hingga end.

Perulangan: For-Loop

- Loop “for” dalam Go (Golang) adalah konstruksi fundamental untuk iterasi yang efisien atas koleksi, yang menawarkan sintaksis yang ringkas dan fleksibilitas. Ini mencakup komponen inisialisasi, kondisi, dan post, yang membuatnya serbaguna untuk berbagai skenario.
- Dalam Golang, for loop digunakan untuk mengulang sekumpulan data. For loop juga dapat digunakan untuk mengulang blok kode sejumlah kali tertentu atau hingga suatu kondisi terpenuhi. For loop berperilaku seperti while loop dalam bahasa pemrograman lain.

- Hampir sama seperti if dan switch, kita juga bisa menambah statement pada for loop. Ada dua statement yang bisa kita tambahkan yaitu: 1) Init statement, merupakan sebuah statement sebelum for loop dieksekusi, dan 2) Post statement, merupakan statement yang akan selalu dieksekusi di akhir pada tiap-tiap perulangan.
- Init statement hanya dieksekusi sekali sehingga bisa kita digunakan untuk menginisialisasi variabel awal yang menentukan counter perulangan. Sedangkan post statement yang selalu dieksekusi pada akhir tiap perulangan bisa kita gunakan untuk menaikkan nilai counter perulangan. Ini akan membuat kode kita menjadi lebih simpel.

Instruksi for-loop memiliki beberapa komponen, yaitu :

- Inisialisasi merupakan assignment variabel iterasi yang bertipe integer. Pada contoh di atas biasanya variabel iterasi = 0 atau 1, artinya iterasi dimulai dari 0 atau 1.
- Kondisi merupakan suatu operasi bernilai boolean yang menyatakan kapan perulangan harus dilakukan. Pada contoh di atas kondisi adalah variabel iterasi <= n (kurang dari atau sama dengan)
- Update merupakan ekspresi yang menyatakan perubahan nilai dari variabel iterasi.

CONTOH SOAL

1. Coso1

Source code:

```
package main

import "fmt"

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

Output:

```
PS C:\semester1_alpro1\alpro1_week5> go run "c:\semester1_alpro1\alpro1_week5\coso1\coso1.go"
2 5
2345
PS C:\semester1_alpro1\alpro1_week5> go run "c:\semester1_alpro1\alpro1_week5\coso1\coso1.go"
-5 7
-5-4-3-2-101234567
```

Deskripsi Program:

Program pada kode Go di atas menerima dua input bilangan bulat, yaitu a dan b. Kemudian, program mencetak semua angka dari a hingga b, termasuk batasnya. Berikut adalah penjelasannya:

1. Deklarasi Variabel:
   - var a, b int: variabel a dan b digunakan untuk menyimpan input bilangan bulat yang dimasukkan pengguna.
   - var j int: variabel j digunakan sebagai variabel pengulangan (counter) dalam loop.
2. Fungsi ‘fmt.Scan(&a, &b)’ akan membaca dua nilai input dari pengguna dan menyimpannya ke dalam variabel ‘a’ dan ‘b’.
3. Perulangan ‘for j = a; j <= b; j = j + 1’ dimulai dari nilai ‘a’ dan akan berjalan sampai nilai ‘j’ mencapai ‘b’. Setiap iterasi, nilai ‘j’ akan bertambah 1.
4. Fungsi ‘fmt.Print(j, "")’ akan mencetak nilai ‘j’ pada setiap iterasi. Fungsi Print digunakan untuk mencetak nilai ‘j’ tanpa menambahkan pemisah atau karakter tambahan antara angka-angka yang dicetak.

2. Coso2

Source code:

```
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 C:\semester1_alpro1\alpro1_week5> go run "c:\semester1_alpro1\alpro1_week5\coso2\coso2.go"
5
11 2
11
32 14
224
6 2
6
15 15
112.5
20 35
350
```

Deskripsi Program:

Program pada kode Go di atas menghitung luas beberapa segitiga berdasarkan input alas dan tinggi dari pengguna. Pengguna memberikan jumlah segitiga yang ingin dihitung, kemudian program mengulang proses perhitungan luas untuk setiap segitiga yang dimasukkan. Berikut adalah penjelasannya:

1. Deklarasi Variabel:
    - var j, alas, tinggi, n int: variabel j digunakan untuk loop, alas dan tinggi untuk menyimpan input alas dan tinggi dari segitiga, serta n untuk menyimpan jumlah segitiga yang akan dihitung.
    - var luas float64: variabel ini menyimpan hasil perhitungan luas segitiga dalam tipe data float64 (bilangan pecahan).
2. Fungsi ‘fmt.Scan(&n)’ akan membaca jumlah segitiga (n) yang akan dihitung luasnya dari input pengguna.
3. Perulangan ‘for j = 1; j <= n; j += 1’ loop ini berjalan dari 1 hingga n, sesuai dengan jumlah segitiga yang ingin dihitung. Pada setiap iterasi, program akan meminta input alas dan tinggi, lalu menghitung luas segitiga berdasarkan input tersebut.
4. Fungsi ‘fmt.Scan(&alas, &tinggi)’ pada setiap iterasi, program meminta input alas dan tinggi untuk setiap segitiga.
5. Luas segitiga dihitung menggunakan rumus luas = 0.5 * alas * tinggi, di mana alas dan tinggi dikonversi ke tipe float64 agar hasil perhitungan memiliki presisi yang lebih baik.
6. Fungsi ‘fmt.Println(luas)’ akan mencetak hasil luas segitiga setelah perhitungan untuk setiap iterasi loop.

3. Coso3

Source code:

```
package main

import "fmt"

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

Output:

```
PS C:\semester1_alpro1\alpro1_week5> go run "c:\semester1_alpro1\alpro1_week5\coso3\coso3.go"
2 100
200
PS C:\semester1_alpro1\alpro1_week5> go run "c:\semester1_alpro1\alpro1_week5\coso3\coso3.go"
7 6
42
```

Deskripsi Program:

Program pada kode Go di atas melakukan operasi pengulangan untuk menghitung hasil perkalian dua bilangan bulat tanpa menggunakan operator perkalian langsung. Proses ini dilakukan dengan menggunakan penjumlahan berulang. Berikut adalah penjelasannya:

1. Deklarasi Variabel:
   - var j, hasil, v1, v2 int:
     - j adalah variabel yang digunakan sebagai counter dalam loop.
     - hasil adalah variabel yang digunakan untuk menyimpan hasil akhir penjumlahan berulang.
     - v1 dan v2 adalah dua bilangan bulat yang dimasukkan pengguna, di mana v1 adalah bilangan yang akan dijumlahkan, dan v2 adalah jumlah iterasi (banyaknya penjumlahan).
2. Fungsi ‘fmt.Scan(&v1, &v2)’ akan membaca dua nilai input dari pengguna, yaitu v1 dan v2.
3. Perulangan ‘for j = 1; j <= v2; j++’ akan dijalankan sebanyak v2 kali, yang berarti bilangan v1 akan dijumlahkan berulang kali sejumlah v2.
4. Penjumlahan Berulang:

   hasil = hasil + v1: pada setiap iterasi, nilai v1 akan ditambahkan ke variabel hasil. Pada awalnya, hasil bernilai nol, dan akan bertambah sebesar v1 pada setiap iterasi.

5. Fungsi ‘fmt.Print(hasil)’ akan mencetak nilai akhir dari hasil, yang merupakan hasil perkalian v1 dan v2.

LATIHAN SOAL

1. Buatlah program untuk menjumlahkan sekumpulan bilangan.
   Masukan terdiri dari suatu bilangan bulat positif n.
   Keluaran berupa bilangan hasil penjumlahan dari 1 sampai dengan n.
   Source code:

```
package main

import "fmt"

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

	for i := 1; i <= n; i++ {
		total += i
	}
	fmt.Println("Hasil penjumlahan dari 1 sampai", n, "adalah", total)
}
```

   Output:

```
PS C:\semester1_alpro1\alpro1_week5> go run "c:\semester1_alpro1\alpro1_week5\kumpulanbil\kumpulanbil.go"
Masukkan bilangan bulat positif: 3
Hasil penjumlahan dari 1 sampai 3 adalah 6
PS C:\semester1_alpro1\alpro1_week5> go run "c:\semester1_alpro1\alpro1_week5\kumpulanbil\kumpulanbil.go"
Masukkan bilangan bulat positif: 5
Hasil penjumlahan dari 1 sampai 5 adalah 15
```

   Deskripsi program:
   Program pada kode Go di atas menghitung jumlah bilangan bulat dari 1 hingga n, di mana n merupakan bilangan bulat positif yang dimasukkan oleh pengguna. Berikut adalah penjelasannya:

   1. Deklarasi Variabel:
      - var n, total int:
        - n adalah variabel yang menyimpan input dari pengguna, yaitu bilangan bulat positif.
        - total adalah variabel yang digunakan untuk menyimpan hasil penjumlahan bilangan dari 1 hingga n.
   2. Input Pengguna:
      - fmt.Print("Masukkan bilangan bulat positif: "): Program menampilkan pesan kepada pengguna untuk memasukkan bilangan bulat positif.
      - fmt.Scan(&n): Bagian ini membaca input dari pengguna dan menyimpannya ke dalam variabel n.
   3. Perulangan ‘for i := 1; i <= n; i++’ perulangan ini berjalan dari i = 1 hingga i mencapai nilai n. Pada setiap iterasi, nilai i akan ditambahkan ke variabel total.
   4. Penjumlahan Bilangan:
      total += i: Pada setiap iterasi, nilai i ditambahkan ke dalam variabel total, sehingga hasil akhir dari loop ini adalah jumlah seluruh bilangan dari 1 hingga n.

5. Fungsi ‘fmt.Println("Hasil penjumlahan dari 1 sampai", n, "adalah", total)’ akan mencetak hasil penjumlahan bilangan dari 1 hingga n.

2. Buatlah program yang digunakan untuk menghitung volume sejumlah n kerucut, apabila diketahui panjang jari-jari alas kerucut dan tinggi dari kerucut.
Masukan terdiri dari beberapa baris. Baris pertama adalah bilangan bulat n, selanjutnya n baris berikutnya masing-masing merupakan panjang jari-jari alas kerucut dan tinggi dari kerucut.
Keluaran terdiri dari beberapa baris, yang masing-masingnya menyatakan volume dari n kerucut.
Source code:

```
package main

import (
	"fmt"
	"math"
)

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

	for i := 1; i <= n; i++ {
		var jariJari, tinggi float64
		fmt.Printf("Masukkan jari-jari dan tinggi kerucut ke-%d: ", i)
		fmt.Scan(&jariJari, &tinggi)
		volume := (1.0 / 3.0) * math.Pi * math.Pow(jariJari, 2) * tinggi
		fmt.Printf("Volume kerucut ke-%d adalah: %.14f\n", i,
volume)
	}
}
```

Output:

```
PS C:\semester1_alpro1\alpro1_week5> go run "c:\semester1_alpro1\alpro1_week5\kerucut\kerucut.go"
Masukkan jumlah kerucut: 1
Masukkan jari-jari dan tinggi kerucut ke-1: 3 4
Volume kerucut ke-1 adalah: 37.69911184307752
PS C:\semester1_alpro1\alpro1_week5> go run "c:\semester1_alpro1\alpro1_week5\kerucut\kerucut.go"
Masukkan jumlah kerucut: 3
Masukkan jari-jari dan tinggi kerucut ke-1: 1 1
Volume kerucut ke-1 adalah: 1.04719755119660
Masukkan jari-jari dan tinggi kerucut ke-2: 2 2
Volume kerucut ke-2 adalah: 8.37758040957278
Masukkan jari-jari dan tinggi kerucut ke-3: 3 3
Volume kerucut ke-3 adalah: 28.27433388230814
```

Deskripsi program:
Program pada kode Go di atas menghitung volume beberapa kerucut berdasarkan input jari-jari dan tinggi dari pengguna. Program ini meminta jumlah kerucut yang ingin dihitung, kemudian menghitung dan mencetak volume setiap kerucut. Berikut adalah penjelasannya:

1. Import Paket:
   - import "fmt": digunakan untuk menangani input dan output.

- import "math": digunakan untuk perhitungan matematika, seperti menghitung nilai π (Pi) dan kuadrat.

2. Deklarasi Variabel:

   var n int: variabel n digunakan untuk menyimpan jumlah kerucut yang akan dihitung volumenya.

3. Input Pengguna:
   - fmt.Print("Masukkan jumlah kerucut: "): program menampilkan pesan untuk meminta jumlah kerucut.
   - fmt.Scan(&n): pengguna memasukkan jumlah kerucut, yang disimpan di variabel n.
4. Perulangan 'for i := 1; i <= n; i++' perulangan ini berjalan sebanyak n kali, sesuai jumlah kerucut yang dimasukkan pengguna. Pada setiap iterasi, program akan meminta input jari-jari dan tinggi kerucut, serta menghitung volumenya.
5. Input Jari-jari dan Tinggi Kerucut:
   - var jariJari, tinggi float64: variabel jariJari dan tinggi digunakan untuk menyimpan input jari-jari dan tinggi kerucut dalam tipe data float64.
   - fmt.Printf("Masukkan jari-jari dan tinggi kerucut ke-%d: ", i): program menampilkan pesan untuk meminta jari-jari dan tinggi kerucut ke-i.
   - fmt.Scan(&jariJari, &tinggi): pengguna memasukkan nilai jari-jari dan tinggi yang kemudian disimpan di variabel jariJari dan tinggi.
6. Perhitungan Volume Kerucut:
   - Rumus volume kerucut adalah:

$$V = \frac{1}{3} x\ pi\ x\ r\ x\ r\ x\ t$$

   - math.Pi: digunakan untuk mendapatkan nilai π (pi).
   - math.Pow(jariJari, 2): digunakan untuk menghitung kuadrat dari jari-jari (r^2).
7. Fungsi 'fmt.Printf("Volume kerucut ke-%d adalah: %.14f\n", i, volume)' akan mencetak hasilnya dengan 14 digit desimal.

3. Buatlah program yang digunakan untuk menghitung hasil pemangkatan dari dua buah bilangan. Program dibuat dengan menggunakan operator perkalian dan struktur kontrol perulangan.
   Masukan terdiri dari dua bilangan bulat positif.
   Keluaran terdiri dari suatu bilangan yang menyatakan hasil bilangan pertama dipangkatkan dengan bilangan kedua.
   Source code:

```
package main

import "fmt"

func main() {
        var dasar, pangkat, result int
```

```
		fmt.Print("Masukkan bilangan dasar : ")
		fmt.Scan(&dasar)
		fmt.Print("Masukkan bilangan pangkat : ")
		fmt.Scan(&pangkat)

		result = 1
		for i := 1; i <= pangkat; i++ {
			result *= dasar
		}
		fmt.Printf("Hasil dari %d pangkat %d adalah: %d\n", dasar, pangkat,
	result)
	}
```

Output:

```
Masukkan bilangan pangkat (exponent): Hasil dari 2 pangkat 10 adalah: 1024
PS C:\semester1_alpro1\alpro1_week5> go run "c:\semester1_alpro1\alpro1_week5\pemangkatan\pemangkatan.go"
Masukkan bilangan dasar : 4 2
Masukkan bilangan pangkat : Hasil dari 4 pangkat 2 adalah: 16
PS C:\semester1_alpro1\alpro1_week5> go run "c:\semester1_alpro1\alpro1_week5\pemangkatan\pemangkatan.go"
Masukkan bilangan dasar : 2 10
Masukkan bilangan pangkat : Hasil dari 2 pangkat 10 adalah: 1024
PS C:\semester1_alpro1\alpro1_week5> go run "c:\semester1_alpro1\alpro1_week5\pemangkatan\pemangkatan.go"
Masukkan bilangan dasar : 10 3
Masukkan bilangan pangkat : Hasil dari 10 pangkat 3 adalah: 1000
```

Deskripsi program:
Program pada kode Go di atas melakukan perhitungan eksponensial sederhana, yaitu menghitung hasil dari suatu bilangan dasar yang dipangkatkan dengan bilangan pangkat yang diberikan oleh pengguna. Berikut adalah penjelasannya:

1. Deklarasi Variabel:
   - var dasar, pangkat, result int:
     - dasar adalah bilangan dasar (base) yang akan dipangkatkan.
     - pangkat adalah bilangan pangkat (exponent) yang menunjukkan seberapa banyak bilangan dasar akan dikalikan.
     - result adalah variabel yang digunakan untuk menyimpan hasil akhir dari perhitungan.
2. Input Pengguna:
   - fmt.Print("Masukkan bilangan dasar : "): program menampilkan pesan untuk meminta bilangan dasar.
   - fmt.Scan(&dasar): pengguna memasukkan nilai bilangan dasar yang disimpan dalam variabel dasar.
   - fmt.Print("Masukkan bilangan pangkat : "): program menampilkan pesan untuk meminta bilangan pangkat.
   - fmt.Scan(&pangkat): pengguna memasukkan nilai bilangan pangkat yang disimpan dalam variabel pangkat.
3. Inisialisasi Hasil:
   result = 1: variabel result diinisialisasi dengan nilai 1, karena bilangan apa pun yang dipangkatkan 0 hasilnya adalah 1.
4. Perhitungan Pemangkatan:
   - for i := 1; i <= pangkat; i++: loop ini berjalan sebanyak pangkat kali. Pada setiap iterasi, program akan mengalikan nilai result dengan dasar.
   - result *= dasar: pada setiap iterasi, nilai result diperbarui dengan mengalikan nilai saat ini dengan bilangan dasar. Ini adalah cara untuk menghitung eksponensial dengan perkalian berulang.

5. Fungsi ‘fmt.Printf("Hasil dari %d pangkat %d adalah: %d\n", dasar, pangkat, result)’ akan mencetak hasil dari operasi pemangkatan setelah perhitungan selesai.

4. Buatlah program yang digunakan untuk menghitung hasil faktorial dari suatu bilangan.
Masukan terdiri dari suatu bilangan bulat non negatif.
Keluaran terdiri dari hasil faktorial dari bilangan bulat n.
Source code:

```
package main

import "fmt"

func main() {
	var n, result int
	fmt.Print("Masukkan bilangan bulat non-negatif: ")
	fmt.Scan(&n)

	result = 1
	for i := 1; i <= n; i++ {
		result *= i
	}
	fmt.Printf("Faktorial dari %d adalah: %d\n", n, result)
}
```

Output:

```
PS C:\semester1_alpro1\alpro1_week5> go run "c:\semester1_alpro1\alpro1_week5\faktorial\faktorial.go"
Masukkan bilangan bulat non-negatif: 0
Faktorial dari 0 adalah: 1
PS C:\semester1_alpro1\alpro1_week5> go run "c:\semester1_alpro1\alpro1_week5\faktorial\faktorial.go"
Masukkan bilangan bulat non-negatif: 1
Faktorial dari 1 adalah: 1
PS C:\semester1_alpro1\alpro1_week5> go run "c:\semester1_alpro1\alpro1_week5\faktorial\faktorial.go"
Masukkan bilangan bulat non-negatif: 5
Faktorial dari 5 adalah: 120
```

Deskripsi program:
Program pada kode Go di atas menghitung faktorial dari sebuah bilangan bulat non-negatif yang dimasukkan oleh pengguna. Faktorial dari suatu bilangan n adalah hasil perkalian semua bilangan bulat positif dari 1 hingga n. Berikut adalah penjelasannya:

1. Deklarasi Variabel:
   - var n, result int:
     - n adalah bilangan bulat non-negatif yang dimasukkan oleh pengguna dan merupakan bilangan yang akan dihitung faktorialnya.
     - result adalah variabel yang digunakan untuk menyimpan hasil perhitungan.
2. Input Pengguna:
   - fmt.Print("Masukkan bilangan bulat non-negatif: "): program menampilkan pesan untuk meminta bilangan bulat non-negatif dari pengguna.

- fmt.Scan(&n): pengguna memasukkan nilai bilangan yang disimpan dalam variabel n.

3. Inisialisasi Hasil:
   result = 1: variabel result diinisialisasi dengan nilai 1, karena faktorial 0 didefinisikan sebagai 1 (0! = 1).
4. Perulangan untuk Menghitung Faktorial:
   - for i := 1; i <= n; i++: loop ini berjalan dari 1 hingga n. Pada setiap iterasi, nilai result akan dikalikan dengan nilai i (1, 2, 3, ..., hingga n).
   - result *= i: pada setiap iterasi, nilai result diperbarui dengan mengalikan nilai saat ini dengan i, sehingga setelah loop selesai, result akan berisi hasil faktorial dari n.
5. Fungsi ‘fmt.Printf("Faktorial dari %d adalah: %d\n", n, result)’ akan mencetak hasilnya setelah perhitungan faktorial selesai.

DAFTAR PUSTAKA

A.14. Perulangan. (n.d.). Retrieved from https://dasarpemrogramangolang.novalagung.com/A-perulangan.html

W3Schools.com. (n.d.). Retrieved from https://www.w3schools.com/go/go_loops.php

Loops in Go Language. Retrieved from https://www.geeksforgeeks.org/loops-in-go-language/