

**LAPORAN PRAKTIKUM**  
**ALGORITMA DAN PEMROGRAMAN 1**  
**MODUL 10**  
**“PERCABANGAN”**



**DISUSUN OLEH:**  
**RAIHAN ADI ARBA**

**103112400071**

**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

Percabangan dalam pemrograman adalah konsep fundamental yang memungkinkan program untuk membuat keputusan dan menjalankan alur yang berbeda berdasarkan kondisi tertentu. Konsep ini sangat penting karena membuat program menjadi lebih dinamis dan interaktif dalam merespons berbagai situasi.

Terdapat beberapa jenis percabangan yang umum digunakan dalam pemrograman. Pertama, pernyataan ``if`` yang merupakan bentuk percabangan paling sederhana dan paling sering digunakan. Pernyataan ini bekerja dengan cara mengevaluasi suatu kondisi boolean (true/false), dan jika kondisi tersebut bernilai true, maka blok kode di dalamnya akan dieksekusi. Misalnya, dalam kasus pengecekan nilai: "Jika nilai lebih dari 70, maka siswa dinyatakan lulus."

Kedua, pernyataan ``if-else`` yang merupakan pengembangan dari pernyataan `if` sederhana. Struktur ini memungkinkan program untuk menjalankan blok kode alternatif ketika kondisi `if` tidak terpenuhi. Ini sangat berguna ketika kita perlu menangani dua kemungkinan yang berbeda. Contohnya: "Jika hujan turun maka bawa payung, jika tidak maka tidak perlu bawa payung."

Ketiga, pernyataan ``else-if`` yang berguna untuk menguji multiple kondisi secara berurutan. Struktur ini sangat efektif ketika ada lebih dari dua kemungkinan kondisi yang perlu diperiksa. Program akan memeriksa setiap kondisi secara berurutan hingga menemukan kondisi yang terpenuhi. Ini sering digunakan dalam kasus seperti pemberian grade nilai: "Jika nilai  $\geq 90$  maka A, jika  $\geq 80$  maka B, jika  $\geq 70$  maka C, dan seterusnya."

Terakhir, pernyataan ``switch`` yang berfungsi seperti saklar untuk mengevaluasi sebuah nilai terhadap beberapa kemungkinan kasus. Berbeda dengan `if` yang fokus pada kondisi logika, `switch` lebih efisien untuk mencocokkan nilai secara langsung, termasuk data numerik dan karakter. `Switch case` sangat berguna ketika kita perlu membandingkan satu variabel dengan beberapa nilai yang berbeda. Misalnya dalam pembuatan menu: "Switch(pilihan): case 1: menu utama, case 2: pengaturan, case 3: keluar."

Semua jenis percabangan ini memiliki peran dan kegunaan masing-masing dalam pemrograman. Penggunaan yang tepat dari struktur-struktur ini akan menghasilkan kode yang lebih efisien, mudah dibaca, dan mudah dipelihara. Percabangan memungkinkan programmer untuk membuat program yang lebih cerdas dan responsif, mampu mengambil keputusan berdasarkan berbagai kondisi yang mungkin terjadi selama program berjalan. Dengan memahami dan menggunakan percabangan dengan baik, programmer dapat menciptakan solusi yang lebih kompleks dan dapat menangani berbagai skenario yang mungkin terjadi dalam program mereka.

## A. UNGUIDED

### 1. Latihan 1

PT POS membutuhkan aplikasi perhitungan biaya kirim berdasarkan berat parcel. Maka, buatlah program BiayaPos untuk menghitung biaya pengiriman tersebut dengan ketentuan sebagai berikut!

Dari berat parcel (dalam gram), harus dihitung total berat dalam kg dan sisanya (dalam

gram). Biaya jasa pengiriman adalah Rp. 10.000,- per kg. Jika sisa berat tidak kurang dari 500 gram, maka tambahan biaya kirim hanya Rp. 5,- per gram saja. Tetapi jika kurang dari 500 gram, maka tambahan biaya akan dibebankan sebesar Rp. 15,- per gram. Sisa berat (yang kurang dari 1kg) digratiskan biayanya apabila total berat ternyata lebih dari 10kg.

Perhatikan contoh sesi interaksi program seperti di bawah ini (teks bergaris bawah adalah input/read):

No.	Contoh masukan dan keluaran
1	Berat parcel (gram): <b>8500</b> Detail berat: 8 kg + 500 gr Detail biaya: Rp. 80000 + Rp. 2500 Total biaya: Rp. 82500
2	Berat parcel (gram): <b>9250</b> Detail berat: 9 kg + 250 gr Detail biaya: Rp. 90000 + Rp. 3750 Total biaya: Rp. 93750
3	Berat parcel (gram): <b>11750</b> Detail berat: 11 kg + 750 gr Detail biaya: Rp. 110000 + Rp. 3750 Total biaya: Rp. 110000

Source Code:

```
package main  
  
import "fmt"
```

```

func main() {
    var berat, beratSisa, beratKg, biayaPerKg, totalBiaya int

    fmt.Print("Masukkan berat parsel (dalam gram): ")
    fmt.Scan(&berat)

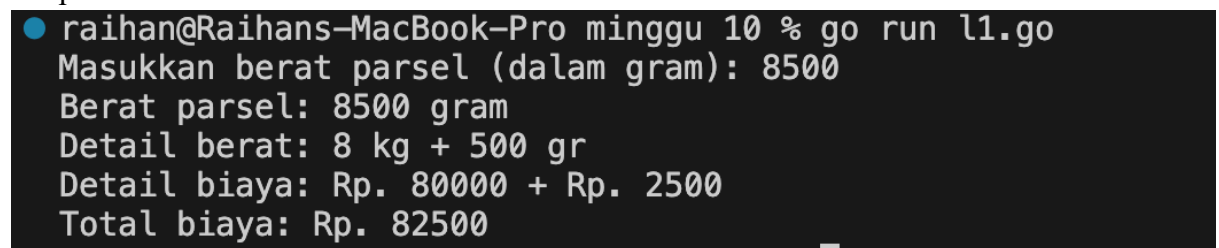
    beratKg = berat / 1000
    beratSisa = berat % 1000
    biayaPerKg = 10000
    totalBiaya = biayaPerKg * beratKg

    if beratKg > 10 {
        beratSisa = 0
    } else if beratSisa >= 500 {
        totalBiaya += beratSisa * 5
    } else {
        totalBiaya += beratSisa * 15
    }

    fmt.Printf("Berat parsel: %d gram\n", berat)
    fmt.Printf("Detail berat: %d kg + %d gr\n", beratKg, beratSisa)
    fmt.Printf("Detail biaya: Rp. %d + Rp. %d\n", biayaPerKg*beratKg,
beratSisa*5)
    fmt.Printf("Total biaya: Rp. %d\n", totalBiaya)
}

```

Output :



```

● raihan@Raihans-MacBook-Pro minggu 10 % go run l1.go
Masukkan berat parsel (dalam gram): 8500
Berat parsel: 8500 gram
Detail berat: 8 kg + 500 gr
Detail biaya: Rp. 80000 + Rp. 2500
Total biaya: Rp. 82500

```

#### Penjelasan Program:

Program meminta pengguna memasukkan berat parsel dalam satuan gram, kemudian melakukan konversi ke kilogram dan menghitung biaya pengirimannya. Sistem perhitungan menggunakan tarif dasar Rp. 10.000 per kilogram, dengan ketentuan khusus untuk sisa berat dalam gram. Jika berat total melebihi 10 kg, sisa berat diabaikan. Untuk berat di bawah 10 kg, sisa berat 500 gram atau lebih dikenakan biaya Rp. 5 per gram, sedangkan sisa berat di bawah 500 gram dikenakan Rp. 15 per gram. Program kemudian menampilkan informasi lengkap meliputi berat total, detail berat dalam kilogram dan gram, rincian biaya, serta total biaya yang harus dibayar.

## 2. Latihan 2

Diberikan sebuah nilai akhir mata kuliah (NAM) [0..100] dan standar penilaian nilai mata kuliah (NMK) sebagai berikut:

NAM	NMK
NAM > 80	A
72.5 < NAM <= 80	AB
65 < NAM <= 72.5	B
57.5 < NAM <= 65	BC
50 < NAM <= 57.5	C
40 < NAM <= 50	D
NAM <= 40	E

Program berikut menerima input sebuah bilangan riil yang menyatakan NAM.

Program menghitung NMK dan menampilkannya.

Source Code :

```
package main

import "fmt"

func main() {
    var nam float64
    var nmk string
    fmt.Print("Nilai akhir mata kuliah: ")
    fmt.Scan(&nam)
    if nam > 80 {
        nmk = "A"
    }
    if nam > 72.5 {
        nmk = "AB"
    }
    if nam > 65 {
        nmk = "B"
    }
    if nam > 57.5 {
        nmk = "BC"
    }
}
```

```

    }
    if nam > 50 {
        nam = "C"
    }
    if nam > 40 {
        nam = "D"
    } else if nam <= 40 {
        nam = "E"
    }
    fmt.Println("Nilai mata kuliah: ", nmk)
}

```

Output :

```

❗ raihan@Raihans-MacBook-Pro minggu 10 % go run lx2.go
# command-line-arguments
./lx2.go:11:9: cannot use "A" (untyped string constant) as flo
at64 value in assignment
./lx2.go:14:9: cannot use "AB" (untyped string constant) as fl
oat64 value in assignment
./lx2.go:17:9: cannot use "B" (untyped string constant) as flo
at64 value in assignment
./lx2.go:20:9: cannot use "BC" (untyped string constant) as fl
oat64 value in assignment
./lx2.go:23:9: cannot use "C" (untyped string constant) as flo
at64 value in assignment
./lx2.go:26:9: cannot use "D" (untyped string constant) as flo
at64 value in assignment
./lx2.go:28:9: cannot use "E" (untyped string constant) as flo
at64 value in assignment

```

Pembahasan :

a. Output untuk nam = 80.1 (Program Awal)

Program awal akan menghasilkan error. Meskipun variabel nam diinisialisasi sebagai float64, program mencoba memberikan nilai string ("A", "AB", dst.) ke variabel nam. Go adalah bahasa yang diketik secara statis, sehingga operasi ini tidak diizinkan dan akan menyebabkan error kompilasi. Selain itu, variabel nmk yang seharusnya menyimpan nilai huruf tidak pernah diubah, sehingga outputnya akan kosong.

b. Kesalahan Program Awal dan Alur yang Benar

Alur program yang salah :

- i. Pencampuran Tipe Data: Program salah mencoba menetapkan nilai string ke variabel float64.
- ii. Logika if yang Salah: Karena semua pernyataan if dievaluasi secara independen, nilai nam akan ditimpa berulang kali. Logika yang benar harus menggunakan else if untuk membuat rantai kondisi yang saling eksklusif.
- iii. Variabel Output yang Salah: Program mencetak variabel nmk yang tidak pernah diubah nilainya.

Alur program yang seharusnya:

- i. Menerima input nilai nam.
- ii. Memeriksa nam terhadap serangkaian rentang nilai menggunakan if-else if.
- iii. Menetapkan nilai huruf yang sesuai ke variabel nmk.
- iv. Mencetak nilai nmk.

c. Program yang Diperbaiki dan Hasil Uji

Source Code true:

```
package main

import "fmt"

func main() {
    var nmk string
    var nam float64
    fmt.Print("Nilai akhir mata kuliah: ")
    fmt.Scan(&nam)
    if nam > 80 {
        nmk = "A"
    } else if nam > 72.5 {
        nmk = "AB"
    } else if nam > 65 {
        nmk = "B"
    } else if nam > 57.5 {
        nmk = "BC"
    } else if nam > 50 {
        nmk = "C"
    } else if nam > 40 {
        nmk = "D"
    } else if nam <= 40 {
        nmk = "E"
    }
    fmt.Println("Nilai mata kuliah: ", nmk)
}
```

Output:

```

● raihan@Raihans-MacBook-Pro minggu 10 % go run l2.go
  Nilai akhir mata kuliah: 80
  Nilai mata kuliah: AB
● raihan@Raihans-MacBook-Pro minggu 10 % go run l2.go
  Nilai akhir mata kuliah: 90
  Nilai mata kuliah: A
○ raihan@Raihans-MacBook-Pro minggu 10 % █

```

#### Deskripsi Program:

Program meminta pengguna memasukkan nilai akhir mata kuliah dalam bentuk angka (float64), kemudian menggunakan struktur kondisional if-else untuk menentukan nilai huruf yang sesuai. Sistem penilaian menggunakan beberapa rentang nilai: nilai di atas 80 mendapat grade A, nilai di atas 72.5 mendapat AB, nilai di atas 65 mendapat B, nilai di atas 57.5 mendapat BC, nilai di atas 50 mendapat C, nilai di atas 40 mendapat D, dan nilai 40 ke bawah mendapat E. Setelah melakukan pengecekan, program akan menampilkan hasil konversi berupa nilai huruf (grade) yang sesuai dengan nilai angka yang dimasukkan.

### 3. Latihan 3

Sebuah bilangan bulat  $b$  memiliki faktor bilangan  $f > 0$  jika  $f$  habis membagi  $b$ .

Contoh: 2 merupakan faktor dari bilangan 6 karena 6 habis dibagi 2. Buatlah program yang menerima input sebuah bilangan bulat  $b$  dan  $b > 1$ . Program harus dapat mencari dan menampilkan semua faktor dari bilangan tersebut!

Perhatikan contoh sesi interaksi program seperti di bawah ini (teks bergaris bawah adalah input/read):

Bilangan: <u>12</u> Faktor: 1 2 3 4 6 12	Bilangan: <u>7</u> Faktor: 1 7
---------------------------------------------	-----------------------------------

Bilangan bulat  $b > 0$  merupakan bilangan prima  $p$  jika dan hanya jika memiliki persis dua faktor bilangan saja, yaitu 1 dan dirinya sendiri.

Lanjutkan program sebelumnya. Setelah menerima masukan sebuah bilangan bulat  $b > 0$ . Program tersebut mencari dan menampilkan semua faktor bilangan tersebut.

Kemudian, program menentukan apakah  $b$  merupakan bilangan prima.

Perhatikan contoh sesi interaksi program seperti di bawah ini (teks bergaris bawah adalah input/read):

Bilangan: <u>12</u> Faktor: 1 2 3 4 6 12 Prima: false	Bilangan: <u>7</u> Faktor: 1 7 Prima: true
-------------------------------------------------------------	--------------------------------------------------

#### Source Code:

```

package main

import "fmt"

```



```

func main() {
    var x, y int
    var result bool
    fmt.Scan(&x, &y)
    if y%x == 0 {
        result = true
    } else {
        result = false
    }
    fmt.Println(result)

    if x%y == 0 {
        result = true
    } else {
        result = false
    }
    fmt.Println(result)
}

```

Output:

```

● raihan@Raihans-MacBook-Pro minggu 9 % go run l3.go
10 5
false
true
● raihan@Raihans-MacBook-Pro minggu 9 % go run l3.go
3 4
false
false

```

Deskripsi Program:

Program menggunakan struktur perulangan for dengan sintaks `for i := 1; i <= b; i++` dimana `i` adalah variabel kontrol yang dimulai dari 1, akan terus bertambah (increment) selama nilainya kurang dari atau sama dengan input `b`. Dalam setiap iterasi, program memeriksa apakah bilangan tersebut merupakan faktor dengan menggunakan operator modulo (%). Setelah menampilkan semua faktor, program melakukan pengecekan bilangan prima dengan memeriksa apakah bilangan tersebut habis dibagi oleh 2, 3, 5, atau 7 (kecuali bilangan 2, 3, 5, dan 7 itu sendiri). Jika kondisi terpenuhi, program mencetak "False" yang menandakan bukan bilangan prima, dan sebaliknya akan mencetak "True" untuk bilangan prima. Program ini efektif dalam memberikan pemahaman tentang karakteristik suatu bilangan melalui faktor-faktornya dan sifat keprimaannya.

## **DAFTAR PUSTAKA**

**Prayogo, N. A. (2021). *Dasar Pemrograman Go. Ebook***