

**LAPORAN PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL II**

**REVIEW STRUKTUR KONTROL**



**Disusun Oleh :**

**Haposan Felix Marcel Siregar / 2311102210**

**IF\_11\_06**

**Dosen Pengampu :**

**Abednego Dwi Septiadi, S.Kom., M.Kom**

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**

**FAKULTAS INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**

**2024**

## **I. DASAR TEORI**

Dasar Teori

### **Struktur Kontrol Golang**

Struktur kontrol merupakan komponen penting dalam bahasa pemrograman karena berfungsi untuk mengatur alur eksekusi program. Dalam bahasa Go (Golang), struktur kontrol yang tersedia mencakup pengulangan (loops), seleksi (conditional statements), dan berbagai bentuk pengendalian alur lainnya, seperti goroutine untuk konkuren.

#### **2.1 Pengulangan (Loops)**

Dalam Golang, pengulangan diimplementasikan melalui perintah `for`. Menariknya, Go tidak memiliki perintah `while` atau `do-while` seperti pada beberapa bahasa lainnya. Sebaliknya, semua bentuk pengulangan diimplementasikan menggunakan variasi dari pernyataan `for`. Bentuk dasar pengulangan menggunakan sintaks `for init; condition; post { }` yang serupa dengan C dan Java. Golang juga mendukung pengulangan tanpa kondisi yang akan terus berjalan sampai ada perintah `break` untuk menghentikannya.

#### **2.2 Seleksi (Conditional Statements)**

Instruksi kondisional `if` digunakan dalam Golang untuk menjalankan kode berdasarkan kondisi tertentu. Golang juga menyediakan `else if` dan `else` untuk menangani berbagai kondisi. Uniknya, Golang memperbolehkan penggunaan statement pendek sebelum kondisi `if`, yang memungkinkan deklarasi variabel yang hanya relevan di dalam blok kondisional. Selain itu, Golang memiliki `switch` statement yang digunakan untuk memilih salah satu dari beberapa blok kode untuk dijalankan berdasarkan nilai dari ekspresi.

#### **2.3 Goroutine dan Pengendalian Konkuren**

Salah satu fitur kontrol alur yang membedakan Golang adalah dukungannya terhadap concurrency melalui goroutine. Goroutine memungkinkan program menjalankan fungsi secara asinkron, tanpa perlu menunggu fungsi lain selesai. Dikombinasikan dengan kanal (channels), goroutine memberikan mekanisme sinkronisasi yang efisien. Penggunaan kanal untuk berkomunikasi antara goroutine menghilangkan banyak masalah yang biasanya terkait dengan penguncian data (locking) pada program yang menjalankan paralelisme.

Dengan fitur-fitur ini, Golang memberikan fleksibilitas dan efisiensi dalam pengelolaan alur program, menjadikannya pilihan populer untuk pengembangan aplikasi yang memerlukan kinerja tinggi dan skalabilitas, seperti layanan web, alat otomatisasi, dan sistem terdistribusi.

## II. GUIDED

### 1. Soal Studi Case

Buat program Go yang menerima tiga input string berbeda dari pengguna dan kemudian mengeluarkan string dalam urutan aslinya, diikuti oleh nilainya

#### Sourcecode

```
package main

import "fmt"

func main() {
    var (
        satu, dua, tiga string
        temp string
    )
    fmt.Print("Masukan input string : ")
    fmt.Scanln(&satu)

    fmt.Print("Masukan input string : ")
    fmt.Scanln(&dua)

    fmt.Print("Masukan input string : ")
    fmt.Scanln(&tiga)
    fmt.Println("Output awal = " + satu + " " + dua + " " + tiga)
    temp = satu
    satu = dua
    dua = tiga
    tiga = temp

    fmt.Println("Output akhir = " + satu + " " + dua + " " + tiga)
}
```

#### Screenshoot Output



```
PS C:\Golang> go run "c:\Golang\tempCodeRunnerFile.go"
Masukan input string : 2
Masukan input string : 6
Masukan input string : 12
Output awal = 2 6 12
Output akhir = 6 12 2
PS C:\Golang>
```

## Deskripsi Program

Program ini meminta pengguna untuk memasukkan tiga string secara berurutan, yang disimpan dalam variabel **satu**, **dua**, dan **tiga**. Setelah itu, program mencetak keluaran awal yang merupakan gabungan dari string ketiga tersebut.

Kemudian, program melakukan pertukaran nilai antara string ketiga tersebut menggunakan variabel sementara **temp**. Pertukaran nilai

## 2. Soal Studi Case

Kode yang diberikan adalah program Go sederhana yang menentukan apakah suatu tahun merupakan tahun kabisat. Program tersebut meminta pengguna untuk memasukkan tahun, memproses masukan tersebut untuk memeriksa apakah tahun tersebut memenuhi kriteria sebagai tahun kabisat, lalu mengeluarkan hasilnya. Penentuan tahun kabisat ditangani oleh fungsi `isLeapYear` yang menerapkan aturan untuk tahun kabisat: suatu tahun merupakan tahun kabisat jika habis dibagi 400 atau jika habis dibagi 4 tetapi tidak habis dibagi 100.

## Sourcecode

```
package main

import (
    "fmt"
)

func main() {
    var tahun int

    fmt.Print("Tahun: ")
    fmt.Scanln(&tahun)

    kabisat := isLeapYear(tahun)

    fmt.Printf("Kabisat: %t\n", kabisat)
}

func isLeapYear(year int) bool {
    if year%400 == 0 {
        return true
    }
}
```

```
if year%4 == 0 && year%100 != 0 {  
    return true  
}  
  
return false  
}
```

## Screenshoot Output

```
PS C:\Golang> go run "c:\Golang\tempCodeRunnerFile.go"  
Tahun: 2024  
Kabisat: true  
PS C:\Golang> go run "c:\Golang\tempCodeRunnerFile.go"  
Tahun: 2025  
Kabisat: false  
PS C:\Golang>
```

## Deskripsi Program

- **Deklarasi Paket** : Kode tersebut milik mainpaket, yang merupakan titik masuk untuk aplikasi Go.
- **Impor** : fmtPaket diimpor untuk operasi I/O yang diformat.
- **Fungsi Utama** :
  - Mendeklarasikan variabel integer tahun untuk menyimpan masukan pengguna.
  - Meminta pengguna untuk memasukkan tahun dan membaca input menggunakan fmt.Scanln.
  - Memanggil isLeapYear fungsi untuk menentukan apakah tahun yang dimasukkan adalah tahun kabisat.
  - Mengeluarkan hasil menggunakan fmt.Printf.
- **Fungsi isLeapYear** :
  - Mengambil parameter integer year.
  - Menerapkan logika untuk memeriksa apakah tahun tersebut merupakan tahun kabisat berdasarkan kondisi yang ditentukan.
  - Mengembalikan nilai boolean yang menunjukkan apakah tahun tersebut merupakan tahun kabisat.

### 3. Soal Studi Case

Kode Go yang diberikan mendefinisikan program yang menghitung volume dan luas permukaan bola berdasarkan radius yang diberikan pengguna. Program ini terdiri dari fungsi `CalculateSphereProperties` yang melakukan perhitungan dan `main` fungsi yang berfungsi sebagai titik masuk program. Program meminta pengguna untuk memasukkan radius, memvalidasinya, menghitung properti yang diperlukan, lalu menampilkan hasilnya. Perhitungan didasarkan pada rumus matematika standar untuk volume dan luas permukaan bola.

#### Sourcecode

```
package main

import (
    "fmt"
    "math"
)

func main() {
    var jariJari float64

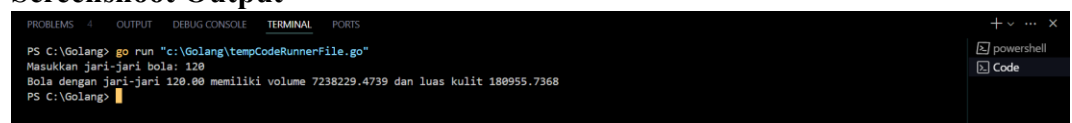
    fmt.Print("Masukkan jari-jari bola: ")
    fmt.Scanln(&jariJari)

    // Menghitung volume bola
    volumeBola := (4.0 / 3.0) * math.Pi * math.Pow(jariJari, 3)

    // Menghitung luas permukaan bola
    luasKulit := 4 * math.Pi * math.Pow(jariJari, 2)

    // Menampilkan hasil
    fmt.Printf("Bola dengan jari-jari %.2f memiliki volume %.4f dan\nluas kulit %.4f\n", jariJari, volumeBola, luasKulit)
}
```

#### Screenshoot Output

A screenshot of a terminal window showing the execution of a Go program. The terminal has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is active, showing the command `go run "c:\Golang\tempCodeRunnerFile.go"` and the output: `Masukkan jari-jari bola: 120`, `Bola dengan jari-jari 120.00 memiliki volume 7238229.4739 dan luas kulit 180955.7368`, and the prompt `PS C:\Golang>`. On the right side of the terminal, there are buttons for 'powershell' and 'Code'.

## Deskripsi Program

Deklarasi paket dalam kode menunjukkan bahwa program ini merupakan bagian dari paket utama, yang merupakan standar untuk program Go yang dapat dieksekusi. Kode ini mengimpor paket `'fmt'` untuk input/output yang diformat, serta paket `'math'` untuk konstanta dan fungsi matematika.

Dalam fungsi `'CalculateSphereProperties'`, terdapat satu parameter yaitu `'radius'` yang bertipe `'float64'`. Fungsi ini mengembalikan volume dan luas permukaan bola, serta error jika radius yang diberikan bukan nilai positif. Logika dalam fungsi ini meliputi validasi terhadap jari-jari, perhitungan volume dan luas permukaan menggunakan rumus matematika yang sesuai, dan pengembalian hasil perhitungan tersebut. Di dalam fungsi `'main'`, program meminta masukan dari pengguna, memanggil fungsi `'CalculateSphereProperties'`, memeriksa adanya kesalahan, dan mencetak hasil yang diperoleh.



### III. UNGUIDED

#### 1. Soal Studi Case

Kode yang diberikan adalah program Go (Golang) yang mendefinisikan fungsi untuk mengonversi suhu dari Celsius ke tiga skala suhu lainnya: Fahrenheit, Reamur, dan Kelvin. ConvertTemperatureFungsi tersebut mengambil suhu dalam Celsius sebagai input dan mengembalikan suhu yang sesuai dalam tiga skala lainnya. Fungsi tersebut mainberfungsi sebagai titik masuk program, yang meminta pengguna untuk memasukkan suhu dalam Celsius, melakukan konversi, lalu menampilkan hasilnya.

#### Sourcecode

```
package main

import "fmt"

func main() {
    var celsius float64

    fmt.Print("Masukkan suhu dalam derajat Celsius: ")
    fmt.Scanln(&celsius)

    // Menghitung konversi suhu
    fahrenheit := (celsius * 9 / 5) + 32
    reamur := celsius * 4 / 5
    kelvin := celsius + 273.15

    // Menampilkan hasil
    fmt.Printf("Suhu dalam Fahrenheit: %.2f\n", fahrenheit)
    fmt.Printf("Suhu dalam Reamur: %.2f\n", reamur)
    fmt.Printf("Suhu dalam Kelvin: %.2f\n", kelvin)
}
```

#### Screenshoot Output



```
PS C:\Golang> go run "c:\Golang\latihan.go"
Masukkan suhu dalam derajat Celsius: 50
Suhu dalam Fahrenheit: 122.00
Suhu dalam Reamur: 40.00
Suhu dalam Kelvin: 323.15
PS C:\Golang>
```

#### Deskripsi Program

Proses dimulai dengan mengimpor paket `fmt`, yang penting untuk menangani operasi input dan output. Variabel bernama `celsius` kemudian dideklarasikan dengan tipe data `float64` untuk menyimpan nilai suhu yang diberikan oleh pengguna. Program ini menggunakan `fmt.Scanln()` untuk menangkap input suhu dalam Celsius.

Selanjutnya, program melakukan kalkulasi untuk mengubah suhu Celsius menjadi Fahrenheit, Reamur, dan Kelvin. Konversi ke Fahrenheit dilakukan menggunakan rumus  $F = (9/5)C + 32$ , sedangkan konversi Reamur menggunakan rumus  $R = (4/5)C$ . Untuk Kelvin, kalkulasinya mudah, diwakili oleh rumus  $K = C + 273,15$ . Terakhir, program mengeluarkan nilai suhu yang dikonversi dalam Fahrenheit, Reamur, dan Kelvin, yang diformat sesuai dengan persyaratan yang ditentukan.

## 2. Soal Studi Case

Membuat program dalam bahasa Go yang akan membaca 5 buah data integer dan mencetaknya dalam format karakter. Kemudian membaca 3 buah data karakter dan mencetak 3 buah karakter setelah karakter tersebut (menurut tabel ASCII).

Masukan terdiri dari dua baris. Baris pertama berisi 5 buah data integer. Data integer mempunyai nilai antara 32 s.d. 127. Baris kedua berisi 3 buah karakter yang berdampingan satu dengan yang lain (tanpa dipisahkan spasi).

Keluaran juga terdiri dari dua baris. Baris pertama berisi 5 buah representasi karakter dari data yang diberikan, yang berdampingan satu dengan lain, tanpa dipisahkan spasi. Baris kedua berisi 3 buah karakter (juga tidak dipisahkan Oleh spasi).

### Source code

```
package main

import (
    "fmt"
)

func main() {
    var data [5]int
    var karakter [3]rune

    fmt.Println("===== Program ASCII =====")

    // Membaca 5 angka dari input
```

```
fmt.Println("Masukkan 5 angka (dari 32 hingga 127), dipisahkan  
dengan spasi:")
```

```
for i := 0; i < 5; i++ {  
    fmt.Scan(&data[i])  
    if data[i] < 32 || data[i] > 127 {  
        fmt.Println("Angka harus antara 32 dan 127.")  
        return  
    }  
}
```

```
fmt.Println("\n===== Keluaran =====")
```

```
// Mencetak karakter untuk 5 angka yang dimasukkan
```

```
fmt.Print("Karakter yang sesuai: ")  
for i := 0; i < 5; i++ {  
    fmt.Print(string(rune(data[i])))  
}  
fmt.Println()
```

```
// Membaca 3 karakter dari input
```

```
fmt.Println("Masukkan 3 karakter yang berdampingan (tanpa  
spasi):")
```

```
var input string  
fmt.Scan(&input)
```

```
if len(input) != 3 {  
    fmt.Println("Harap masukkan tepat 3 karakter.")  
    return  
}
```

```
// Mengonversi karakter ke rune
```

```
for i, char := range input {  
    karakter[i] = char  
}
```

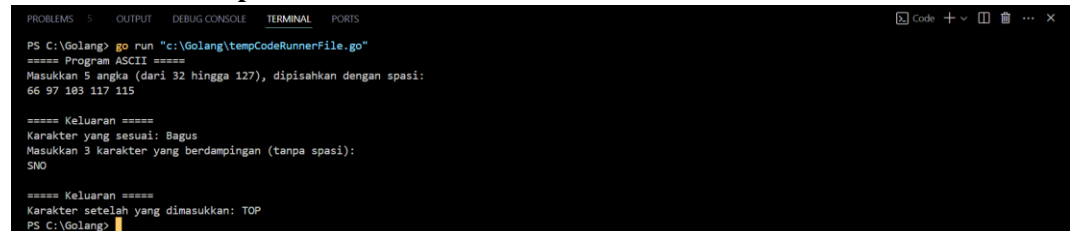
```
fmt.Println("\n===== Keluaran =====")
```

```
// Mencetak 3 karakter setelah karakter yang dimasukkan
```

```
fmt.Print("Karakter setelah yang dimasukkan: ")  
for _, char := range karakter {  
    fmt.Print(string(char + 1)) // Menampilkan karakter berikutnya  
}  
fmt.Println()
```

```
}
```

## Screenshoot Output



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Golang> go run "c:\Golang\tempCodeRunnerFile.go"
==== Program ASCII ====
Masukkan 5 angka (dari 32 hingga 127), dipisahkan dengan spasi:
66 97 103 117 115

==== Keluaran ====
Karakter yang sesuai: Bagus
Masukkan 3 karakter yang berdampingan (tanpa spasi):
SNO

==== Keluaran ====
Karakter setelah yang dimasukkan: TOP
PS C:\Golang>
```

## Deskripsi Program

Kode tersebut merupakan program Go yang meminta pengguna memasukkan 5 angka (32-127) dan 3 karakter berdampingan, lalu menampilkan karakter ASCII yang sesuai dengan angka dan 3 karakter berikutnya.

### Bagian 1:

- Meminta pengguna memasukkan 5 angka (32-127) dan memvalidasi inputnya.
- Mencetak karakter ASCII yang sesuai dengan angka-angka yang dimasukkan.

### Bagian 2:

- Meminta pengguna memasukkan 3 karakter berdampingan.
- Mengonversi karakter ke tipe data rune.
- Mencetak 3 karakter berikutnya dari karakter yang dimasukkan.

Kode tersebut mendemonstrasikan penggunaan array, konversi tipe data rune dan string, serta operasi aritmatika pada karakter.

## 3. Soal Studi Case

Kode Go yang diberikan mendefinisikan aplikasi konsol sederhana yang meminta pengguna untuk memasukkan urutan empat warna. Program tersebut memeriksa apakah warna yang dimasukkan cocok dengan urutan yang telah ditetapkan: "merah", "kuning", "hijau", dan "ungu". Pengguna memiliki lima kali kesempatan untuk memasukkan urutan yang benar. Jika pengguna berhasil mencocokkan urutan tersebut, pesan berhasil akan dicetak; jika tidak, pesan gagal akan ditampilkan.

## Sourcecode

```
package main

import (
    "fmt"
)

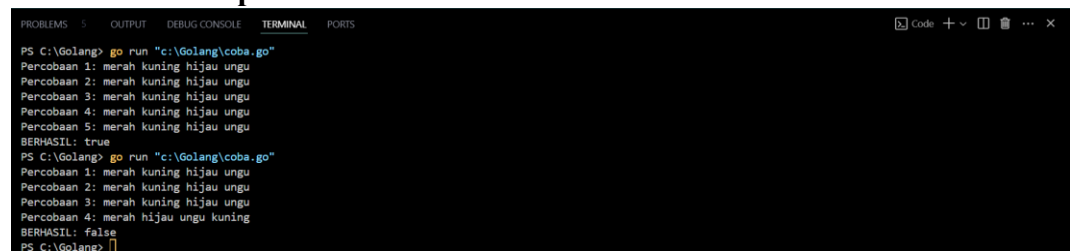
func main() {
    var warna1, warna2, warna3, warna4 string
    var berhasil bool

    // Loop untuk 5 kali percobaan
    for i := 1; i <= 5; i++ {
        fmt.Printf("Percobaan %d: ", i)
        fmt.Scanln(&warna1, &warna2, &warna3, &warna4)

        // Periksa apakah urutan warna sesuai
        if warna1 == "merah" && warna2 == "kuning" && warna3 ==
            "hijau" && warna4 == "ungu" {
            berhasil = true
        } else {
            berhasil = false
            break // Hentikan perulangan jika urutan salah
        }
    }

    // Tampilkan hasil percobaan
    if berhasil {
        fmt.Println("BERHASIL: true")
    } else {
        fmt.Println("BERHASIL: false")
    }
}
```

## Screenshoot Output



```
PS C:\Golang> go run "c:\Golang\coba.go"
Percobaan 1: merah kuning hijau ungu
Percobaan 2: merah kuning hijau ungu
Percobaan 3: merah kuning hijau ungu
Percobaan 4: merah kuning hijau ungu
Percobaan 5: merah kuning hijau ungu
BERHASIL: true
PS C:\Golang> go run "c:\Golang\coba.go"
Percobaan 1: merah kuning hijau ungu
Percobaan 2: merah kuning hijau ungu
Percobaan 3: merah kuning hijau ungu
Percobaan 4: merah hijau ungu kuning
BERHASIL: false
PS C:\Golang>
```

## Deskripsi Program

Fungsi Utama :

- Variabel :
  - warna1, warna2, warna3, warna4: String untuk menampung masukan warna dari pengguna.
  - berhasil: Boolean untuk menunjukkan apakah pengguna berhasil mencocokkan urutan warna.
- Loop : forLoop berjalan lima kali, yang memungkinkan pengguna memasukkan warna.
- Penanganan Input : Program ini digunakan `fmt.Scanln` untuk membaca input pengguna.
- Pemeriksaan Kondisi : if Pernyataan memeriksa apakah masukan cocok dengan urutan yang telah ditentukan sebelumnya.

Output : Program mencetak apakah pengguna berhasil atau gagal dalam mencocokkan urutan.

## 4. Soal Studi Case

Kode Go yang disediakan adalah aplikasi konsol sederhana yang memungkinkan pengguna memasukkan nama-nama bunga. Program tersebut meminta pengguna untuk memasukkan jumlah bunga yang ingin mereka masukkan, mengumpulkan nama-nama bunga, dan memformatnya menjadi satu string yang dipisahkan oleh " - ". Program tersebut juga menghitung jumlah bunga yang dimasukkan dan menampilkan jumlah ini. Program tersebut menghentikan proses input jika pengguna mengetik "SELESAI".

## Sourcecode

```
package main

import (
    "bufio"
    "fmt"
    "os"
)

func main() {
    var N int
    var bunga string
    var pita string
    var jumlahBunga int
```

```

    fmt.Print("Masukkan banyak bunga (N): ")
    fmt.Scanln(&N)

    if N <= 0 {
        fmt.Println("Input N harus bilangan bulat positif.")
        return
    }

    scanner := bufio.NewScanner(os.Stdin)
    for i := 1; i <= N; i++ {
        fmt.Printf("Bunga %d: ", i)
        scanner.Scan()
        bunga = scanner.Text()

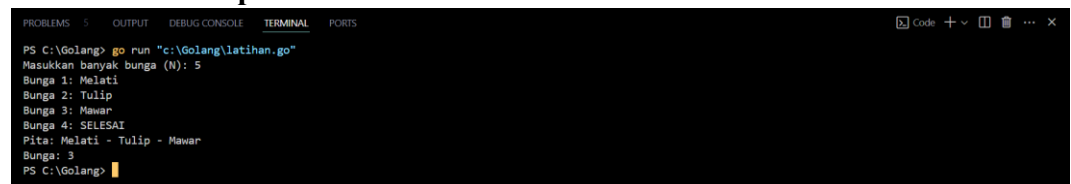
        if bunga == "SELESAI" {
            break
        }

        if i > 1 {
            pita += " - "
        }
        pita += bunga
        jumlahBunga++
    }

    fmt.Println("Pita:", pita)
    fmt.Println("Bunga:", jumlahBunga)
}

```

## Screenshoot Output



```

PS C:\Golang> go run "c:\Golang\latihan.go"
Masukkan banyak bunga (N): 5
Bunga 1: Melati
Bunga 2: Tulip
Bunga 3: Mawar
Bunga 4: SELESAI
Pita: Melati - Tulip - Mawar
Bunga: 3
PS C:\Golang>

```

## Deskripsi Program

### Impor : Mengimpor paket-paket yang diperlukan:

- bufio: Untuk operasi I/O yang di-buffer.
- fmt: Untuk operasi I/O yang diformat.
- os: Untuk fungsionalitas sistem operasi, khususnya untuk membaca dari input standar.
- Fungsi Utama :

- Variabel :
  - N: Bilangan bulat untuk menyimpan jumlah bunga.
  - bunga: Tali untuk menahan nama setiap bunga.
  - pita: Sebuah string untuk mengumpulkan nama bunga yang diformat.
  - jumlahBunga: Penghitung bilangan bulat untuk jumlah bunga yang dimasukkan.
- Penanganan Input : Meminta pengguna untuk memasukkan jumlah bunga dan memvalidasi bahwa itu adalah bilangan bulat positif.
- Loop : Mengumpulkan nama-nama bunga hingga jumlah yang ditentukan tercapai atau pengguna memasukkan "SELESAI".
- Keluaran : Mencetak rangkaian nama bunga yang diformat dan jumlah totalnya.

## 5. Soal Studi Case

### Sebelum modifikasi

Setiap hari Pak Andi membawa banyak barang belanjaan dari pasar dengan mengendarai sepeda motor. Barang belanjaan tersebut dibawa dalam kantong terpal di kiri-kanan motor. Sepeda motor tidak akan oleng jika selisih berat barang di kedua kantong sisi tidak lebih dari 9 kg.

Buatlah program Pak Andi yang menerima input dua buah bilangan real positif yang menyatakan berat total masing-masing isi kantong terpal. Program akan terus meminta Input bilangan tersebut hingga salah satu kantong terpal berisi 9 kg atau lebih.

### Sourcecode

```
package main

import (
    "fmt"
)

func main() {
    var kantong1, kantong2 float64
```



```

for {

    // Input the weight of the two bags

    fmt.Print("Masukan berat belanjaan di kedua kantong: ")

    fmt.Scan(&kantong1, &kantong2)

    // Check if either of the bags contains 9 kg or more

    if kantong1 >= 9 || kantong2 >= 9 {

        fmt.Println("Proses selesai.")

        break

    }

}

}

```

## Screenshot program

```

PROBLEMS 11 OUTPUT DEBUG CONSOLE TERMINAL PORTS
Masukan berat belanjaan di kedua kantong: 5 10
Proses selesai.
PS C:\Golang> go run "c:\Golang\coba.go"
Masukan berat belanjaan di kedua kantong: 2 6
Masukan berat belanjaan di kedua kantong: 5.5 1.0
Masukan berat belanjaan di kedua kantong: 7.1 8.5
Masukan berat belanjaan di kedua kantong: 9 5.8
Proses selesai.
PS C:\Golang>

```

## Deskripsi program

Program ini berulang kali meminta pengguna memasukkan berat belanjaan di dua kantong (kantong1 dan kantong2). Perulangan ini terus berjalan hingga dihentikan.

Program ini akan selesai ketika salah satu kantong memiliki berat 9 kg atau lebih.

Secara ringkas, program ini:

1. Terus meminta berat belanjaan di dua kantong sampai salah satu beratnya 9 kg atau lebih.

2. Tidak menghitung total berat belanjaan.
3. Menampilkan pesan "Proses selesai" ketika berat salah satu kantong mencapai 9 kg atau lebih.

### Sesudah modifikasi

Pada program modifikasi, program akan menampilkan **true** jika selisih kedua isi kantong lebih dari atau sama dengan 9 kg. Program akan berhenti memproses apabila total berat isi kedua kantong melebihi 150 kg atau salah satu kantong beratnya negatif.

### Sourcecode

```
package main

import (
    "fmt"
    "math"
)

func main() {
    var kantong1, kantong2 float64

    for {
        // Meminta input dari pengguna
        fmt.Print("Masukan berat belanjaan di kedua kantong: ")
        fmt.Scanf("%f %f", &kantong1, &kantong2)

        // Memeriksa apakah salah satu kantong beratnya negatif atau
        // total berat melebihi 150 kg
        if kantong1 < 0 || kantong2 < 0 || kantong1+kantong2 > 150 {
            fmt.Println("Proses selesai.")
            break
        }

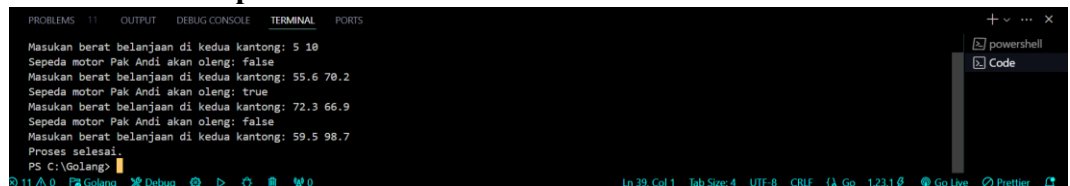
        // Menampilkan true jika selisih berat kedua kantong >= 9 kg
        if math.Abs(kantong1-kantong2) >= 9 {
            fmt.Println("Sepeda motor pak Andi akan oleng: true")
        } else {
            fmt.Println("Sepeda motor pak Andi akan oleng: false")
        }
    }
}
```

```

    }
}

```

## Screenshoot Output



```

PROBLEMS 11 OUTPUT DEBUG CONSOLE TERMINAL PORTS
Masukan berat belanjaan di kedua kantong: 5 10
Sepeda motor Pak Andi akan oleng: false
Masukan berat belanjaan di kedua kantong: 55.6 70.2
Sepeda motor Pak Andi akan oleng: true
Masukan berat belanjaan di kedua kantong: 72.3 66.9
Sepeda motor Pak Andi akan oleng: false
Masukan berat belanjaan di kedua kantong: 59.5 98.7
Proses selesai.
PS C:\Golang>

```

## Deskripsi Program

- Variabel : Dua variabel float64, kantong1 dan kantong2, dideklarasikan untuk menampung berat tas.
- Loop Tak Terbatas : forLoop digunakan untuk meminta masukan dari pengguna secara terus-menerus.
- Penanganan Input : Program ini digunakan fmt.Scan untuk membaca dua nilai float dari pengguna.
- Validasi : Memeriksa bobot negatif dan memastikan berat total tidak melebihi 150 kg.
- Pemeriksaan Perbedaan Berat : Menghitung perbedaan absolut antara dua bobot dan mencetak apakah sepeda motor akan tidak stabil berdasarkan perbedaan tersebut.

## 6. Soal Studi Case

### Sebelum modifikasi

Membuat program yang menerima input sebuah bilangan sebagai K, kemudian menghitung dan menampilkan nilai  $f(K)$  sesuai dengan rumus berikut!

$$f(k) = \frac{(4k + 2)^2}{(4k + 1)(4k + 3)}$$

### Sebelum Modifikasi

```
package main

import (
    "fmt"
)

func main() {
    // Declare the variable to hold the input
    var K int

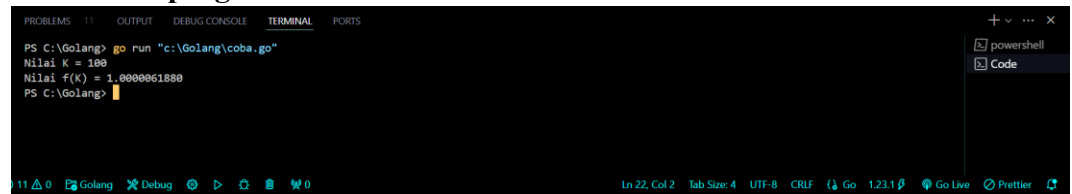
    // Ask for user input
    fmt.Print("Nilai K = ")
    fmt.Scanf("%d", &K)

    // Initialize the value of v2 (approximation of sqrt(2))
    v2 := 1.0

    // Loop through the range from 0 to K to compute the product
    for k := 0; k <= K; k++ {
        numerator := (4*float64(k) + 2) * (4*float64(k) + 2)
        denominator := (4*float64(k) + 1) * (4*float64(k) + 3)
        v2 *= numerator / denominator
    }

    // Display the result as the approximation of sqrt(2)
    fmt.Printf("Nilai akar 2 = %.10f\n", v2)
}
```

## Screenshot program



```
PS C:\Golang> go run "c:\Golang\coba.go"
Nilai K = 100
Nilai f(K) = 1.0000061880
PS C:\Golang>
```

## Deskripsi program

Fungsi Utama :

- Fungsi ini main berfungsi sebagai titik masuk program.
- Variabel  $K$  bertipe `float64` dideklarasikan untuk menyimpan masukan pengguna.
- Program meminta input kepada pengguna, membaca nilai ke dalam  $K$ , dan melakukan perhitungan untuk mendapatkan pembilang dan penyebut fungsi tersebut.
- Terakhir, ia menghitung hasilnya dan mencetaknya dengan presisi tertentu.

## Sesudah modifikasi

Memodifikasi program yang menerima input sebuah bilangan sebagai  $K$ , kemudian menghitung  $\sqrt{2}$  untuk  $k$  tersebut. Hampiran  $\sqrt{2}$  dituliskan dalam ketelitian 10 angka dibelakang koma dengan menggunakan rumus berikut ini!

$$\sqrt{2} = \prod_{k=0}^{\infty} \frac{(4k+2)^2}{(4k+1)(4k+3)}$$

## Sourcecode

```
package main

import (
    "fmt"
)

func main() {
    // Declare the variable to hold the input
    var K int
```

```

// Ask for user input
fmt.Print("Nilai K = ")
fmt.Scanf("%d", &K)

// Initialize the value of v2 (approximation of sqrt(2))
v2 := 1.0

// Loop through the range from 0 to K to compute the product
for k := 0; k <= K; k++ {
    numerator := (4*float64(k) + 2) * (4*float64(k) + 2)
    denominator := (4*float64(k) + 1) * (4*float64(k) + 3)
    v2 *= numerator / denominator
}

// Display the result as the approximation of sqrt(2)
fmt.Printf("Nilai akar 2 = %.10f\n", v2)
}

```

## Screenshoot Output

```

PROBLEMS 11 OUTPUT DEBUG CONSOLE TERMINAL PORTS
Nilai K = 10
Nilai akar 2 = 1.4062058441
PS C:\Golang> go run "c:\Golang\tempCodeRunnerFile.go"
Nilai K = 100
Nilai akar 2 = 1.4133387072
PS C:\Golang> go run "c:\Golang\tempCodeRunnerFile.go"
Nilai K = 1000
Nilai akar 2 = 1.4141252651
PS C:\Golang>

```

## Deskripsi Program

- **Deklarasi Variabel:**
- Deklarasi variabel K yang bertipe int untuk menyimpan input dari pengguna.
- **Meminta Input Pengguna:**
- Menggunakan fmt.Print untuk meminta pengguna memasukkan nilai K.
- fmt.Scanf digunakan untuk membaca input dari pengguna dan menyimpannya ke dalam variabel K.
- **Inisialisasi Variabel v2:**
- Variabel v2 diinisialisasi dengan nilai 1.0, yang akan digunakan untuk menyimpan hasil perhitungan.
- **Perulangan (for):**
- Menggunakan loop dari 0 hingga K untuk menghitung nilai berdasarkan rumus yang diberikan.

- Di dalam loop:
  - Menghitung numerator (pembilang) menggunakan rumus  $(4k+2)2(4k+2)^2(4k+2)2$ .
  - Menghitung denominator (penyebut) menggunakan rumus  $(4k+1)(4k+3)(4k+1)(4k+3)(4k+1)(4k+3)$ .
  - Mengalikan  $v2$  dengan hasil bagi dari numerator dan denominator.
- **Menampilkan Hasil:**
- Menggunakan `fmt.Printf` untuk mencetak hasil akhir  $v2$ , yang merupakan perkiraan nilai  $2\sqrt{2}$ , dengan format 10 angka desimal.