

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

MODUL II

JUDUL



Disusun Oleh :

Daffa Aryaputra / 2311102272

IF-11-06

Dosen Pengampu :

Abednego Dwi Septiadi

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Struktur kontrol dalam bahasa pemrograman Golang adalah komponen penting yang mengatur alur eksekusi program berdasarkan kondisi tertentu atau perulangan. Salah satu struktur kontrol yang umum adalah percabangan kondisional menggunakan `'if'`, `'else'`, dan `'else if'`. Dengan percabangan ini, program dapat menentukan apakah blok kode tertentu harus dijalankan berdasarkan hasil evaluasi kondisi yang menghasilkan nilai `'true'` atau `'false'`. Misalnya, jika suatu kondisi terpenuhi, satu set perintah dijalankan, jika tidak, set perintah yang berbeda dijalankan. Selain itu, Golang juga mendukung struktur kontrol `'switch'`, yang digunakan untuk mengevaluasi satu variabel terhadap beberapa kemungkinan nilai. `'switch'` sering lebih efisien dan mudah dibaca daripada menggunakan banyak blok `'if-else'`, terutama ketika ada banyak kondisi yang perlu diuji.

Selain percabangan, Golang juga mendukung struktur perulangan dengan menggunakan `'for'`. Dalam Golang, `'for'` adalah satu-satunya bentuk loop yang tersedia, tetapi sangat fleksibel dan bisa digunakan sebagai loop berbasis kondisi, loop dengan counter, atau bahkan loop tak terbatas. Dalam perulangan berbasis counter, `'for'` akan terus berjalan selama kondisi yang ditentukan terpenuhi. Namun, jika perulangan perlu diakhiri lebih awal, kita bisa menggunakan perintah `'break'` untuk keluar dari loop. Ada juga perintah `'continue'` yang digunakan untuk melewati satu iterasi dalam loop dan melanjutkan ke iterasi berikutnya tanpa menjalankan kode yang ada setelahnya pada iterasi tersebut.

Selain kontrol alur standar seperti `'if'` dan `'for'`, Golang juga memiliki fitur `'defer'`. `'defer'` digunakan untuk menunda eksekusi suatu perintah atau fungsi hingga fungsi di mana `'defer'` berada selesai dijalankan. Ini sangat berguna untuk memastikan bahwa sumber daya seperti file atau koneksi selalu dibebaskan setelah digunakan, karena perintah yang di-defer akan tetap dieksekusi bahkan jika terjadi error atau fungsi berakhir lebih awal. Semua struktur kontrol ini memungkinkan programmer untuk membuat alur logika yang fleksibel dan responsif dalam program, memudahkan pengelolaan kondisi dan perulangan yang kompleks.

II. GUIDED

1.

Sourcecode

```
package main

import "fmt"

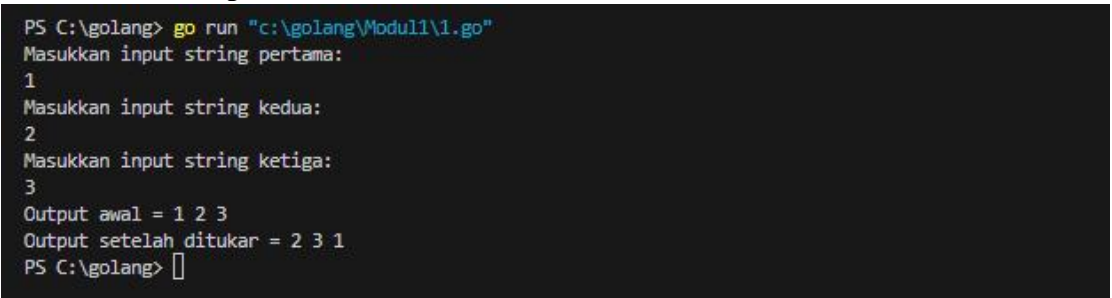
func main() {
    var (
        satu, dua, tiga string
        temp                string
    )

    fmt.Println("Masukkan input string pertama: ")
    fmt.Scanln(&satu)
    fmt.Println("Masukkan input string kedua: ")
    fmt.Scanln(&dua)
    fmt.Println("Masukkan input string ketiga: ")
    fmt.Scanln(&tiga)

    fmt.Println("Output awal = " + satu + " " + dua + " "
+ tiga)

    // Proses penukaran
    temp = satu
    satu = dua
    dua = tiga
    tiga = temp
\
    fmt.Println("Output setelah ditukar = " + satu + " " +
dua + " " + tiga)
}
```

Screenshoot Output



```
PS C:\golang> go run "c:\golang\Modul1\1.go"
Masukkan input string pertama:
1
Masukkan input string kedua:
2
Masukkan input string ketiga:
3
Output awal = 1 2 3
Output setelah ditukar = 2 3 1
PS C:\golang> 
```

Deskripsi Program

Program di atas adalah program sederhana yang berfungsi untuk menerima tiga input string dari pengguna, kemudian menukarnya dalam urutan tertentu, dan menampilkan hasilnya. Program ini ditulis menggunakan bahasa pemrograman Go. Cara kerjanya dapat dijelaskan sebagai berikut:

1. Pertama, tiga variabel string `satu`, `dua`, dan `tiga` dideklarasikan, serta satu variabel `temp` untuk membantu proses penukaran nilai.
2. Program kemudian meminta pengguna untuk memasukkan tiga string melalui prompt di terminal. Setiap input disimpan ke dalam variabel `satu`, `dua`, dan `tiga`.
3. Setelah pengguna memasukkan ketiga input, program menampilkan urutan awal dari string yang diinputkan dengan mencetak ketiga variabel tersebut secara berurutan.
4. Kemudian, dilakukan proses penukaran string, di mana:
 - Nilai dari variabel `satu` disimpan sementara ke dalam variabel `temp`.
 - Nilai dari variabel `dua` dipindahkan ke `satu`, nilai dari variabel `tiga` dipindahkan ke `dua`, dan nilai yang disimpan sementara di `temp` dipindahkan ke `tiga`.
5. Setelah proses penukaran selesai, program mencetak output baru dengan urutan yang telah diubah.

Misalnya, jika pengguna memasukkan string "apel", "jeruk", dan "mangga", program akan mencetak output awal sebagai "apel jeruk mangga". Setelah penukaran, program mencetak "jeruk mangga apel", menunjukkan urutan string telah berubah.

2.

Sourcecode

```
package main

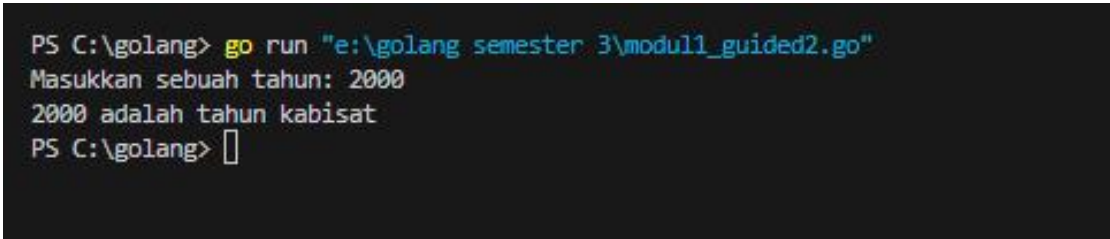
import (
    "fmt"
)

func cekTahunKabisat() {
    var tahun int
    fmt.Print("Masukkan sebuah tahun: ")
    fmt.Scanln(&tahun)

    if (tahun%400 == 0) || (tahun%4 == 0 && tahun%100 !=
```

```
0) {  
    fmt.Println(tahun, "adalah tahun kabisat")  
} else {  
    fmt.Println(tahun, "bukan tahun kabisat")  
}  
}  
  
func main() {  
    cekTahunKabisat()  
}
```

Screenshoot Output



```
PS C:\golang> go run "e:\golang semester 3\modul1_guided2.go"  
Masukkan sebuah tahun: 2000  
2000 adalah tahun kabisat  
PS C:\golang> 
```

Deskripsi Program

Program yang dibuat adalah program untuk menentukan apakah sebuah tahun merupakan tahun kabisat atau tidak. Program ini menggunakan algoritma logika kondisional untuk menentukan apakah sebuah tahun memenuhi kriteria tahun kabisat.

Cara kerja program ini adalah sebagai berikut:

1. Program meminta pengguna untuk memasukkan sebuah tahun melalui fungsi `fmt.Print` dan `fmt.Scanln`.
2. Kemudian, program memeriksa apakah tahun yang dimasukkan memenuhi kriteria tahun kabisat dengan menggunakan kondisi logika.
3. Jika tahun memenuhi salah satu kriteria tahun kabisat, maka program akan menampilkan pesan bahwa tahun tersebut adalah tahun kabisat. Jika tidak, maka program akan menampilkan pesan bahwa tahun tersebut bukan tahun kabisat.

Dengan demikian, program ini dapat membantu pengguna untuk menentukan apakah sebuah tahun merupakan tahun kabisat atau tidak dengan cara yang mudah dan akurat.

3.

Sourcecode

```
package main

import (
    "fmt"
    "math"
)

func main() {
    var radius int

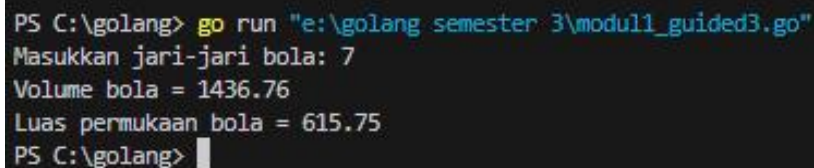
    // Input jari-jari
    fmt.Print("Masukkan jari-jari bola: ")
    fmt.Scan(&radius)

    // Menghitung volume bola
    volume := (4.0 / 3.0) * math.Pi *
    math.Pow(float64(radius), 3)

    // Menghitung luas permukaan bola
    luas := 4 * math.Pi * math.Pow(float64(radius), 2)

    // Tampilkan hasil
    fmt.Printf("Volume bola = %.2f\n", volume)
    fmt.Printf("Luas permukaan bola = %.2f\n", luas)
}
```

Screenshoot Output



```
PS C:\golang> go run "e:\golang semester 3\modul1_guided3.go"
Masukkan jari-jari bola: 7
Volume bola = 1436.76
Luas permukaan bola = 615.75
PS C:\golang> 
```

Deskripsi Program

Program yang dibuat adalah program untuk menghitung volume dan luas permukaan sebuah bola. Program ini menggunakan algoritma matematika untuk menghitung nilai volume dan luas permukaan bola berdasarkan jari-jari yang diinput oleh pengguna.

Cara kerja program ini adalah sebagai berikut:

Program meminta pengguna untuk memasukkan jari-jari bola melalui fungsi `fmt.Print` dan `fmt.Scan`. Kemudian, program menggunakan rumus matematika untuk menghitung volume dan luas permukaan bola. Rumus yang digunakan adalah:

$$\text{Volume bola} = \frac{4}{3} \times \pi \times r^3$$

$$\text{Luas permukaan bola} = 4 \times \pi \times r^2$$

Setelah menghitung nilai volume dan luas permukaan bola, program menampilkan hasilnya kepada pengguna melalui fungsi `fmt.Printf`. Hasil yang ditampilkan adalah nilai volume dan luas permukaan bola dengan dua digit di belakang koma.

III. UNGUIDED

1. 2a

Sourcecode

```
package main

import (
    "fmt"
    "strconv"
)

func main() {
    var celsiusStr string
    fmt.Print("Masukkan suhu dalam Celcius: ")
    fmt.Scanln(&celsiusStr)

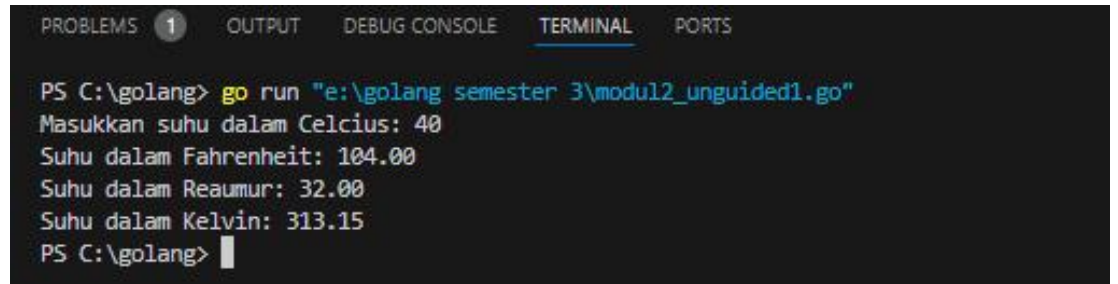
    celsius, err := strconv.ParseFloat(celsiusStr, 64)
    if err != nil {
        fmt.Println("Input tidak valid. Mohon masukkan angka.")
        return
    }

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

    fmt.Printf("Suhu dalam Fahrenheit: %.2f\n", fahrenheit)
    fmt.Printf("Suhu dalam Reaumur: %.2f\n", reamur)
    fmt.Printf("Suhu dalam Kelvin: %.2f\n", kelvin)
```

```
}
```

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 command executed is 'go run "e:\golang semester 3\modul2_unguided1.go"'. The program prompts for temperature in Celsius, and the user enters '40'. The program then displays the converted values: Fahrenheit (104.00), Reaumur (32.00), and Kelvin (313.15).

```
PS C:\golang> go run "e:\golang semester 3\modul2_unguided1.go"
Masukkan suhu dalam Celcius: 40
Suhu dalam Fahrenheit: 104.00
Suhu dalam Reaumur: 32.00
Suhu dalam Kelvin: 313.15
PS C:\golang>
```

Deskripsi Program

Program ini mengkonversi suhu dari Celcius ke Fahrenheit, Reaumur, dan Kelvin.

Algoritma:

Input: Program meminta pengguna untuk memasukkan suhu dalam derajat Celsius.

Konversi: Program mengkonversi suhu Celsius ke Fahrenheit, Reaumur, dan Kelvin menggunakan rumus berikut:

Fahrenheit = $(\text{Celsius} * 9 / 5) + 32$

Reaumur = $\text{Celsius} * 4 / 5$

Kelvin = $\text{Celsius} + 273.15$

Output: Program menampilkan hasil konversi suhu dalam Fahrenheit, Reaumur, dan Kelvin.

Cara Kerja Program:

Membaca Input: Program pertama-tama membaca input suhu dalam Celcius dari pengguna melalui keyboard.

Validasi Input: Program memastikan bahwa input yang dimasukkan adalah angka. Jika bukan angka, program akan menampilkan pesan kesalahan dan berhenti.

Mengkonversi Suhu:

Program menghitung suhu dalam Fahrenheit menggunakan rumus: $(\text{Celsius} * 9 / 5) + 32$

Program menghitung suhu dalam Reaumur menggunakan rumus: $\text{Celsius} * 4 / 5$

Program menghitung suhu dalam Kelvin menggunakan rumus: $\text{Celsius} + 273.15$

Menampilkan Output: Setelah melakukan konversi, program akan menampilkan hasil perhitungan suhu dalam Fahrenheit, Reaumur, dan Kelvin ke layar.

Contoh:

Jika pengguna memasukkan suhu 50 derajat Celsius, maka program akan menampilkan output berikut:

Suhu dalam Fahrenheit: 122.00

Suhu dalam Reaumur: 40.00

Suhu dalam Kelvin: 323.15

2. 2a

Sourcecode

```
package main

import "fmt"

func main() {
    var numbers [5]int
    var chars [3]byte

    // Membaca 5 buah data integer
    fmt.Println("Masukkan 5 buah data integer (32-127):")
    for i := 0; i < 5; i++ {
        fmt.Scanf("%d", &numbers[i])
    }

    // Membaca 3 buah karakter
    fmt.Println("Masukkan 3 buah karakter:")
    for i := 0; i < 3; i++ {
        fmt.Scanf("%c", &chars[i])
    }

    // Mencetak representasi karakter dari data integer
    fmt.Print("Representasi karakter: ")
    for _, num := range numbers {
        fmt.Printf("%c", num)
    }
    fmt.Println()

    // Mencetak 3 karakter setelah karakter yang diberikan
    fmt.Print("3 karakter setelahnya: ")
    for _, char := range chars {
        fmt.Printf("%c", char+1)
    }
    fmt.Println()
}
```

```
}
```

Screenshoot Output

```
PS C:\golang> go run "e:\golang semester 3\modul2_unguided2.go"
Masukkan 5 buah data integer (32-127):
66 97 103 117 115
Masukkan 3 buah karakter:
SNO
Representasi karakter: Bagus
3 karakter setelahnya:
TO
```

Deskripsi Program

Program ini dirancang untuk melakukan beberapa operasi sederhana pada data integer dan karakter. Program ini akan:

- Meminta input: Pengguna diminta untuk memasukkan 5 buah angka bulat (integer) dalam rentang 32-127 dan 3 buah karakter.
- Mengubah integer ke karakter: Angka-angka bulat yang dimasukkan akan diinterpretasikan sebagai kode ASCII dan kemudian dicetak sebagai karakter yang sesuai.
- Menggeser karakter: Setiap karakter yang dimasukkan akan digeser satu posisi ke depan dalam tabel ASCII, lalu dicetak.

Algoritma:

- Deklarasi variabel: Program mendeklarasikan dua buah array, yaitu numbers untuk menyimpan 5 angka integer dan chars untuk menyimpan 3 karakter.
- Membaca input: Program menggunakan fmt.Scanf untuk membaca input dari pengguna, baik itu angka maupun karakter.
- Konversi integer ke karakter: Setiap angka dalam array numbers secara langsung dikonversi menjadi karakter menggunakan format %c dalam fmt.Printf. Ini karena setiap karakter memiliki kode ASCII yang unik, dan angka-angka tersebut diasumsikan sebagai kode ASCII.
- Menggeser karakter: Setiap karakter dalam array chars ditambahkan 1. Karena tabel ASCII disusun secara berurutan, menambahkan 1 pada kode ASCII suatu karakter akan menghasilkan karakter berikutnya dalam tabel. Mencetak hasil: Program mencetak hasil konversi integer ke karakter dan karakter yang telah digeser.

Cara Kerja:

- Misalnya, pengguna memasukkan angka 65, 66, 67, 68, 69 dan karakter 'a', 'b', 'c'.

- Program menyimpan angka-angka tersebut dalam array numbers dan karakter dalam array chars.
- Ketika program mencoba mencetak %c untuk angka 65, ia akan mencari karakter yang memiliki kode ASCII 65. Dalam tabel ASCII, kode 65 mewakili karakter 'A'. Begitu pula dengan angka-angka lainnya.
- Ketika program menambahkan 1 pada karakter 'a' (yang memiliki kode ASCII 97), hasilnya adalah karakter 'b' (kode ASCII 98).

Output:

Output dari program ini akan berupa dua baris:

Baris pertama: Menampilkan representasi karakter dari 5 angka integer yang dimasukkan.

Baris kedua: Menampilkan 3 karakter setelah karakter yang dimasukkan.

3. 2b

Sourcecode

```
package main

import (
    "fmt"
    "math"
)

func main() {
    var kantongKiri, kantongKanan float64
    totalBerat := 0.0
    prosesSelesai := false

    for {
        fmt.Print("Masukkan berat belanjaan di kedua kantong: ")
        fmt.Scanln(&kantongKiri, &kantongKanan)

        fmt.Printf("Masukan berat belanjaan di kedua
kantong: %.1f %.1f\n", kantongKiri, kantongKanan)

        totalBerat += kantongKiri + kantongKanan

        if kantongKiri >= 9 || kantongKanan >= 9 {
            prosesSelesai = true
        }

        selisih := math.Abs(kantongKiri - kantongKanan)
```

```

        if selisih >= 9 {
            fmt.Println("Sepeda motor pak Andi akan oleng:
true")
        } else {
            fmt.Println("Sepeda motor pak Andi akan oleng:
false")
        }
        if totalBerat > 150 {
            prosesSelesai = true
        }
        if prosesSelesai {
            fmt.Println("Proses selesai.")
            break
        }
    }
}

```

Screenshoot Output

```

PS C:\golang> go run "e:\golang semester 3\modul2_unguided3.go"
Masukkan berat belanjaan di kedua kantong: 5 10
Masukan berat belanjaan di kedua kantong: 5.0 10.0
Sepeda motor pak Andi akan oleng: false

```

Deskripsi Program

Program di atas adalah simulasi untuk menghitung berat total belanjaan yang dimasukkan ke dua kantong (kantong kiri dan kantong kanan) secara berulang, dengan kondisi tertentu yang mengakhiri proses perhitungan, serta mengecek apakah perbedaan berat antara kedua kantong menyebabkan sepeda motor "Pak Andi" oleng.

Algoritma dan Cara Kerja Program:

Inisialisasi Variabel:

Dua variabel kantongKiri dan kantongKanan untuk menyimpan berat belanjaan di masing-masing kantong.

totalBerat diinisialisasi dengan nilai 0, untuk melacak total berat belanjaan di kedua kantong.

prosesSelesai adalah boolean yang menandakan apakah proses perhitungan harus dihentikan (diinisialisasi dengan nilai false).

Perulangan (Loop):

Program berjalan dalam sebuah perulangan for tanpa batas (for {}), di mana pada setiap iterasi pengguna diminta untuk memasukkan berat belanjaan di kedua kantong.

Setelah menerima input, program mencetak berat yang dimasukkan.

Penghitungan Berat Total:

Berat dari kedua kantong dijumlahkan ke dalam totalBerat.

Kondisi untuk Mengakhiri Proses:

Proses akan dihentikan jika salah satu dari kondisi berikut terpenuhi:

Salah satu kantong (kiri atau kanan) memiliki berat 9 kg atau lebih.

Total berat dari kedua kantong melebihi 150 kg.

Memeriksa Keseimbangan Kantong:

Program menghitung selisih antara berat kantong kiri dan kanan menggunakan `math.Abs()`, yang mengembalikan nilai absolut dari selisih tersebut.

Jika selisih berat antara kantong kiri dan kanan adalah 9 kg atau lebih, program akan mencetak bahwa "Sepeda motor pak Andi akan oleng: true".

Jika selisihnya kurang dari 9 kg, program mencetak bahwa sepeda motor tidak oleng: "false".

Menghentikan Proses:

Jika salah satu kondisi penghentian tercapai (berat salah satu kantong ≥ 9 atau total berat > 150 kg), variabel `prosesSelesai` diubah menjadi `true`, dan program mencetak pesan "Proses selesai", kemudian keluar dari loop dan mengakhiri eksekusi.

Penjelasan Output:

Program akan terus meminta input berat belanjaan dari pengguna untuk kedua kantong dan menghitung total beratnya.

Pada setiap iterasi, program juga akan mengecek apakah selisih berat kantong cukup besar untuk menyebabkan sepeda motor oleng, dan mencetak hasilnya.

Jika salah satu kantong mencapai berat ≥ 9 kg, atau total berat dari kedua kantong melebihi 150 kg, program akan berhenti dengan mencetak "Proses selesai."

4. 2b

Sourcecode

```
package main

import (
    "fmt"
    "strings"
)

func main() {
    var jumlahBunga int
    var bunga []string
    var pita string

    for {
        fmt.Print("Masukkan jumlah bunga: ")
        fmt.Scanln(&jumlahBunga)

        if jumlahBunga <= 0 {
            fmt.Println("Jumlah bunga harus lebih dari 0")
            continue
        }

        bunga = make([]string, jumlahBunga)
```

```

        for i := 0; i < jumlahBunga; i++ {
            fmt.Printf("Bunga %d: ", i+1)
            fmt.Scanln(&bunga[i])
        }

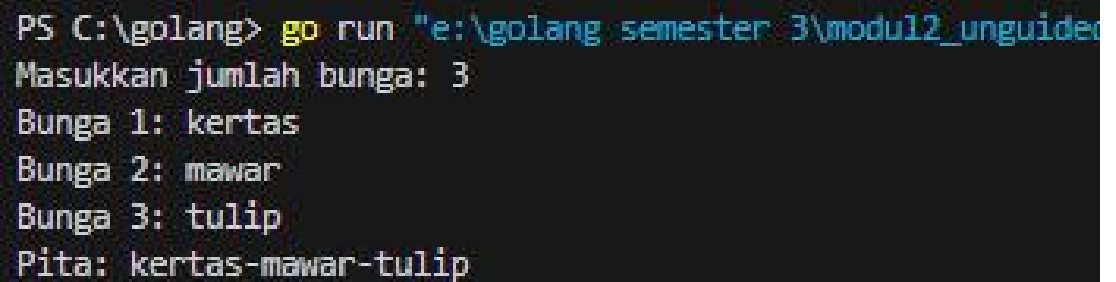
        pita = strings.Join(bunga, "-")
        fmt.Println("Pita:", pita)

        var lanjut string
        fmt.Print("Lanjutkan? (ya/tidak): ")
        fmt.Scanln(&lanjut)

        if strings.ToLower(lanjut) != "ya" {
            break
        }
    }
}

```

Screenshoot Output



```

PS C:\golang> go run "e:\golang semester 3\modul2_unguided...
Masukkan jumlah bunga: 3
Bunga 1: kertas
Bunga 2: mawar
Bunga 3: tulip
Pita: kertas-mawar-tulip

```

Deskripsi Program

Program ini dirancang untuk membuat sebuah "pita bunga" virtual. Pita bunga di sini diartikan sebagai sebuah string (rantai karakter) yang berisi nama-nama bunga yang dipisahkan oleh tanda hubung (-). Program ini memungkinkan pengguna untuk memasukkan sejumlah bunga yang diinginkan, lalu menggabungkan nama-nama bunga tersebut menjadi satu baris.

Cara Kerja:

Meminta Input:

Program mengawali dengan meminta pengguna untuk memasukkan jumlah bunga yang ingin didaftarkan.

Setelah itu, program akan meminta pengguna untuk memasukkan nama-nama bunga satu per satu sesuai dengan jumlah yang telah ditentukan.

Membuat Daftar Bunga:

Nama-nama bunga yang diinputkan oleh pengguna akan disimpan dalam sebuah struktur data yang disebut slice. Slice ini akan menampung semua nama bunga yang telah dimasukkan.

Menggabungkan Nama-nama Bunga:

Setelah semua nama bunga terkumpul, program akan menggunakan fungsi `strings.Join()` untuk menggabungkan semua elemen dalam slice menjadi satu string. Tanda hubung (-) digunakan sebagai pemisah antara setiap nama bunga.

Menampilkan Hasil:

Hasil penggabungan nama-nama bunga (yaitu, "pita bunga") akan ditampilkan di layar.

Iterasi:

Program akan terus meminta input dari pengguna hingga pengguna memutuskan untuk berhenti.

Penjelasan Bagian-bagian Kode:

`jumlahBunga`: Variabel integer untuk menyimpan jumlah bunga yang ingin didaftarkan.

`bunga`: Slice of string untuk menyimpan nama-nama bunga yang diinputkan.

`pita`: Variabel string untuk menyimpan hasil penggabungan nama-nama bunga.

for loop: Digunakan untuk membuat program berjalan secara berulang, sehingga pengguna dapat memasukkan banyak daftar bunga.
strings.Join(): Fungsi bawaan Go untuk menggabungkan elemen-elemen dalam slice menjadi satu string.

Contoh Penggunaan:

Misalkan kamu ingin membuat pita bunga dengan 3 jenis bunga, yaitu mawar, melati, dan tulip. Ketika kamu menjalankan program, kamu akan diminta untuk memasukkan jumlah bunga (3). Kemudian, kamu akan diminta untuk memasukkan nama-nama bunga satu per satu. Setelah itu, program akan menampilkan hasil seperti ini:

Pita: mawar-melati-tulip

5. 2b

Sourcecode

```
package main

import (
    "fmt"
    "math"
)

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

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

        fmt.Scan(&kantong1, &kantong2)

        if kantong1 < 0 || kantong2 < 0 {
```

```

        fmt.Println("Proses selesai.")

        break
    }

    if kantong1+kantong2 > 150 {
        fmt.Println("Proses selesai.")

        break
    }

    oleng := math.Abs(kantong1-kantong2) >= 9

    fmt.Printf("Sepeda motor Pak Andi akan oleng: %v\n", oleng)

    if kantong1 >= 9 || kantong2 >= 9 {
        fmt.Println("Proses selesai.")

        break
    }
}
}

```

Screenshoot Output

```

PS C:\golang> go run "e:\golang semester 3\modul2_unguided5.go"
Masukan berat belanjaan di kedua kantong: 5 10
Sepeda motor Pak Andi akan oleng: false

```

Deskripsi Program

Program ini dirancang untuk mensimulasikan kondisi keseimbangan beban saat Pak Andi mengendarai sepeda motor dengan membawa dua kantong belanjaan. Program ini akan terus meminta pengguna untuk memasukkan berat kedua kantong belanjaan hingga pengguna memasukkan nilai negatif

atau total berat melebihi 150. Program kemudian akan menghitung apakah sepeda motor Pak Andi akan oleng berdasarkan selisih berat kedua kantong dan memberikan output berupa nilai boolean (true atau false).

Algoritma yang Digunakan

Program ini menggunakan algoritma perulangan (loop) yang akan terus berjalan hingga kondisi tertentu terpenuhi. Dalam kasus ini, perulangan akan berhenti jika:

- Pengguna memasukkan nilai negatif untuk salah satu atau kedua kantong belanjaan.
- Total berat kedua kantong belanjaan melebihi 150.
- Salah satu kantong belanjaan memiliki berat 9 kg atau lebih.

Cara Kerja Program

1. Inisialisasi: Program dimulai dengan mendefinisikan sebuah fungsi main yang merupakan titik awal eksekusi program. Di dalam fungsi ini, terdapat sebuah loop for yang akan berjalan terus-menerus.

2. Input: Dalam setiap iterasi loop, program akan meminta pengguna untuk memasukkan berat belanjaan di kedua kantong. Nilai yang dimasukkan pengguna akan disimpan dalam variabel kantong1 dan kantong2.

3. Kondisi Berhenti: Setelah menerima input, program akan memeriksa beberapa kondisi:

- Nilai Negatif: Jika salah satu atau kedua nilai yang dimasukkan negatif, maka loop akan dihentikan dan program akan mencetak pesan "Proses selesai."

- Total Berat Melebihi 150: Jika total berat kedua kantong melebihi 150, maka loop akan dihentikan dan program akan mencetak pesan "Proses selesai."

- Salah Satu Kantong 9 Kg atau Lebih: Jika salah satu kantong memiliki berat 9 kg atau lebih, maka loop akan dihentikan dan program akan mencetak pesan "Proses selesai."

4. Perhitungan: Jika tidak ada kondisi berhenti yang terpenuhi, program akan menghitung selisih berat kedua kantong menggunakan fungsi `math.Abs`. Selisih berat ini kemudian dibandingkan dengan nilai 9. Jika selisihnya sama dengan atau lebih besar dari 9, maka dianggap sepeda motor akan oleng.

5. Output: Hasil perhitungan akan dicetak ke layar dalam bentuk kalimat yang menyatakan apakah sepeda motor Pak Andi akan oleng atau tidak.

6. Ulangi: Proses di atas akan diulang terus-menerus hingga salah satu kondisi berhenti terpenuhi.

Output yang Dihasilkan

Program akan menghasilkan output berupa:

Pesan permintaan input: "Masukan berat belanjaan di kedua kantong:"
Hasil perhitungan: "Sepeda motor Pak Andi akan oleng: true" atau "Sepeda motor Pak Andi akan oleng: false"
Pesan selesai: "Proses selesai."
Intinya: Program ini memberikan simulasi sederhana untuk menentukan keseimbangan beban pada sepeda motor berdasarkan berat kedua kantong belanjaan. Program ini menggunakan logika perbandingan dan perulangan untuk mencapai tujuannya.

Peningkatan:

Program ini dapat ditingkatkan dengan:

Validasi input: Menambahkan pengecekan untuk memastikan bahwa input yang dimasukkan pengguna adalah angka.

Unit pengukuran: Menambahkan unit pengukuran (misalnya, kg) pada output untuk memberikan informasi yang lebih jelas.

Skala yang lebih realistis: Menyesuaikan batas berat maksimum dan ambang batas selisih berat agar lebih sesuai dengan kondisi nyata.

6. 2b

Sourcecode

```
package main

import (
    "fmt"
    "math"
)

func fK(K float64) float64 {
    numerator := (4*K + 2) * (4*K + 2)
    denominator := (4*K + 1) * (4*K + 3)
    return numerator / denominator
}

func sqrt2_approx(K int) float64 {
    result := 1.0
    for i := 0; i <= K; i++ {
        numerator := (4*float64(i) + 2) * (4*float64(i) + 2)
        denominator := (4*float64(i) + 1) * (4*float64(i) + 3)
        result *= numerator / denominator
    }
    return result
}
```

```

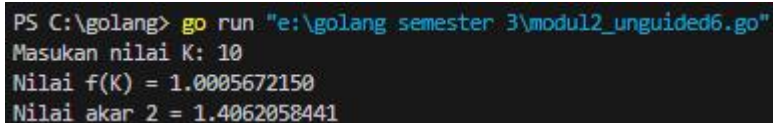
func main() {
    var K float64
    fmt.Print("Masukan nilai K: ")
    fmt.Scan(&K)

    resultF := fK(K)
    fmt.Printf("Nilai f(K) = %.10f\n", resultF)

    Kint := int(math.Round(K))
    resultSqrt2 := sqrt2_approx(Kint)
    fmt.Printf("Nilai akar 2 = %.10f\n", resultSqrt2)
}

```

Screenshot Output



```

PS C:\golang> go run "e:\golang semester 3\modul2_unguided6.go"
Masukan nilai K: 10
Nilai f(K) = 1.0005672150
Nilai akar 2 = 1.4062058441

```

Deskripsi Program

Program di atas adalah sebuah program dalam bahasa Go (Golang) yang menghitung dua hal utama: nilai dari sebuah fungsi matematika $f(K)f(K)f(K)$ dan nilai pendekatan akar kuadrat dari 2 berdasarkan input integer K.

Pendekatan Akar Kuadrat dari 2:

Fungsi `sqrt2_approx(K int) float64` menghitung pendekatan nilai akar 2 menggunakan iterasi berdasarkan parameter integer K. Proses perhitungannya adalah sebagai berikut:

Fungsi memulai dengan `result = 1.0`.

Sebuah loop berjalan dari 0 hingga K, di mana pada setiap iterasi dilakukan penghitungan dengan rumus yang mirip dengan $f(K)f(K)f(K)$, tetapi menggunakan indeks iterasi `iii` sebagai nilai.

Hasil dari setiap iterasi dikalikan ke dalam `result`, yang pada akhirnya mendekati nilai akar 2 jika jumlah iterasi K cukup besar.

Main Function (Fungsi Utama):

Fungsi utama dimulai dengan meminta pengguna untuk memasukkan nilai K, yang dapat berupa bilangan desimal.

Program kemudian menghitung nilai $f(K)f(K)f(K)$ dengan memanggil fungsi $f(K)$ dan menampilkan hasilnya dengan format presisi 10 desimal. Setelah itu, nilai K dibulatkan menggunakan fungsi $\text{math.Round}(K)$ untuk menghasilkan versi integer dari K , yang digunakan sebagai input untuk fungsi $\text{sqrt2_approx}(K_{\text{int}})$. Fungsi ini menghitung pendekatan nilai akar 2 berdasarkan jumlah iterasi yang dihasilkan dari nilai K bulat tersebut. Hasil pendekatan akar 2 kemudian ditampilkan juga dengan format presisi 10 desimal.

Penjelasan Output:

Nilai $f(K)f(K)f(K)$: Program akan menampilkan hasil dari fungsi $f(K)f(K)f(K)$ dengan nilai desimal yang sangat presisi.

Nilai Akar 2: Program juga menghitung nilai pendekatan akar 2 berdasarkan proses iterasi hingga sebanyak K kali.

Contoh output:

Masukan nilai K : 5

Nilai $f(K) = 1.0370370370$

Nilai akar 2 = 1.4142135624