

**LAPORAN PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL 2  
RIVIEW STRUKTUR KONTROL**



**Disusun Oleh :**

**Loisa Vanica Saragih/2311102280**

**S1 IF11-05**

**Dosen Pengampu :**

**Arif Amrulloh, S.Kom., M.Kom.**

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**

**FAKULTAS INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**

**2024**

## **I. DASAR TEORI**

### **Struktur Kontrol**

Struktur kontrol di dalam bahasa pemrograman adalah perintah berbentuk (struktur) tertentu dimana fungsi (control) dipakai untuk mengatur jalannya suatu program. Percabangan struktur control dipakai apabila dalam suatu perintah diperlukan untuk dijalankan dengan syarat-syarat atau kondisi-kondisi tertentu.

Berikut adalah jenis-jenis struktur kontrol dasar yang dikenal:

#### **1. Pemilihan**

-Selection

Tata kontrol ini memberi program kemampuan untuk memilih. Contoh yang paling familiar dari struktur ini adalah if...else.

- Switch

Otomatisasi lain dari pemilihan adalah switch yang memungkinkan kamu untuk memilih antara banyak kemungkinan.

#### **2. Perulangan (Loop)**

Done when one wants to repeat a task then the repetition is used. Perulangan bersyarat, termasuk for, while, dan do...while.

#### **3. Pengendalian Aliran Lanjutan**

Berikut ini adalah fitur lain untuk kontrol lebih lanjut seperti break, yang dapat stop pada suatu loop dan continue yang dapat hanya melimpah pada satu tahap dan kelanjutan pada tahap selanjutnya.

## II. GUIDED

1. Program untuk membaca dan menampilkan nama.

### Sourcecode

```
package main
import "fmt"
func main() {
    nama := "Loisa Vanica Saragih"
    fmt.Print(nama)
}
```

### Screenshoot Output



### Deskripsi Program

Algoritma Program di mulai dengan mendefinisikan package utama yaitu package bernama main, di mana artinya bahwa program ini adalah program yang dapat berdiri sendiri dan dapat dieksekusi langsung. library fmt digunakan dengan cara mennggilction input/output seperti mencetak nilai pada layar. main() menyediakan fungsi dasar dari program tersebut ditulis secara khusus. Menurut main() sebuah variable bernama nama ditetapkan dengan tipe data string dan nilai yang diisi adalah Loisa Vanica Saragih. Tugas fmt.Print(nama) untuk menampilkan output bersifat string dari variabel nama tersebut. Program berakhir.

Cara Kerja Program ini dimulai dengan menjalankan fungsi min yang terdapat didalamnya. Variabel nama juga di deklarasikan dan di inisialisasikan dengan string dari “Loisa Vanica Saragih”. Sebenarnya terdapat dua subroutine fmt.Println() dan fmt.Print() dimana subroutine fmt.Print(nama) dijalankan yang mana variabel nama akan dicetak di layar. Lalu mencetak string dan program selesai.

Output yang Dihasilkan hanya menyimpan nama “Loisa Vanica Saragih” dalam sebuah variable dan setelah itu dicetak ke layar.

2. Buatlah sebuah program yang menerima input berupa warna dari ke 4 gelas reaksi sebanyak 5 kali percobaan. Siswa kelas IPA di salah satu sekolah menengah atas di Indonesia sedang mengadakan praktikum kimia. Di setiap percobaan akan menggunakan 4 tabung reaksi, yang mana susunan warna cairan di setiap tabung akan menentukan hasil

percobaan. Siswa diminta untuk mencatat hasil percobaan tersebut. Percobaan dikatakan berhasil apabila susunan warna zat cair pada gelas 1 hingga gelas 4 secara berturutan adalah 'merah', 'kuning', 'hijau', dan 'ungu' selama 5 kali percobaan berulang. Buatlah sebuah program yang menerima input berupa warna dari ke 4 gelas reaksi sebanyak 5 kali percobaan. Kemudian program akan menampilkan true apabila urutan warna sesuai dengan informasi yang diberikan pada paragraf sebelumnya, dan false untuk urutan warna lainnya. Perhatikan contoh sesi interaksi program seperti di bawah ini (teks bergaris bawah adalah input/read)

### Sourcecode

```
package main

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

func main() {
    //urutan warna yang benar
    correctOrder := []string{"merah", "kuning", "hijau",
    "ungu"}

    //membaca input untuk 5 percobaan
    reader := bufio.NewReader(os.Stdin)
    success := true

    for i := 1; i <= 5; i++ {
        fmt.Printf("Percobaan %d: ", i)

        //membaca input dari pengguna
        input, _ := reader.ReadString('\n')
        input = strings.TrimSpace(input)

        //memisahkan input berdasarkan spasi
        colors := strings.Split(input, " ")

        //mengecek apakah urutan warna sesuai
        for j := 0; j < 4; j++ {
            if colors[j] != correctOrder[j] {
                success = false
                break
            }
        }

        //jika ada percobaan yang tidak sesuai, keluar
        dari loop
        if !success {
```

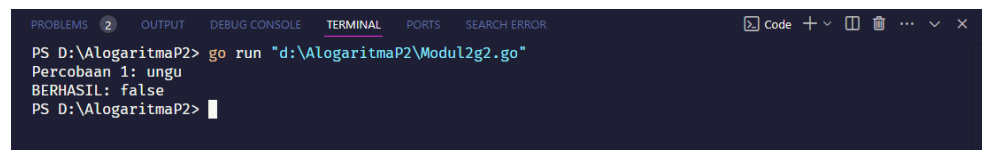
```

        break
    }
}

//menampilkan hasil
if success {
    fmt.Println("BERHASIL: true")
} else {
    fmt.Println("BERHASIL: false")
}
}
package main
import "fmt"
func main() {
    nama := "Loisa Vanica Saragih"
    fmt.Print(nama)
}

```

## Screenshoot Output



```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR
PS D:\AlogaritmaP2> go run "d:\AlogaritmaP2\Modul2g2.go"
Percobaan 1: ungu
BERHASIL: false
PS D:\AlogaritmaP2>

```

## Deskripsi Program

Program meminta pengguna untuk menginputkan urutan warna sebanyak lima kali dan selain itu program akan menyediakan juga pengecekan apakah urutan warna yang diinputkan pengguna tersebut sesuai dengan urutan warna yang sudah di(setting) oleh program. Jika seluruh percobaan input sesuai dengan urutan yang benar, program akan mencetak pesan "BERHASIL: sebagai contoh, dengan nilai benar maka program akan mencetak "BERHASIL: true", jika tidak sesuai maka program akan di stop dan mencetak "BERHASIL: false".

### Algoritma dan Cara kerjanya

Memasukkan urutan warna nilai-nilai ini ke dalam variabel correctOrder, yang nilainya adalah merah, kuning, hijau, ungu. Program menggunakan bufio.NewReader dari tipe datanya input pembaca. Pengguna melakukan 5 percobaan untuk memasukkan urutan warna. Program berupa string dari user, membaginya berdasarkan spasi, kemudian menambahkan warna-warna yang dimasukkan user ke dalam sebuah slice bernama colors.

Program ini kemudian menggunakan input warna pengguna dan mengurutkan untuk di bandingkan dengan urutan warna yang benar. Jika

ada satu warna tidak sesuai, program segera memberikan tanda bahwa percobaan tersebut tidak berhasil dan keluar dari loop. Setelah semua percobaan selesai, program akan menampilkan hasil:

Jika semua input sesuai urutan yang benar, program mencetak "BERHASIL: true". Jika ada satu input yang salah, program mencetak "BERHASIL: false" maka algoritma akan berhenti lebih awal.

#### Output yang Dihasilkan

Jika pengguna memasukkan warna dengan urutan yang benar pada semua percobaan, outputnya true, jika pengguna memasukkan warna dengan urutan yang salah pada percobaan maka outputnya false.

### 3. Penjumlahan dari angka yang diinputkan pengguna

#### Sourcecode

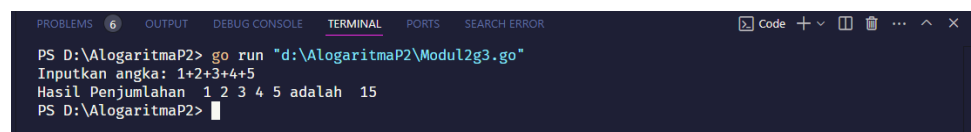
```
package main

import (
    "fmt"
)

func main(){
    var a,b,c,d,e int
    var hasil int
    fmt.Print("Inputkan angka: ")
    fmt.Scanln(&a, &b, &c, &d, &e)

    hasil = a+b+c+d+e
    fmt.Println("Hasil Penjumlahan ",a,b,c,d,e, "adalah",hasil)
}
```

#### Screenshoot Output



```
PS D:\AlogaritmaP2> go run "d:\AlogaritmaP2\Modul2g3.go"
Inputkan angka: 1+2+3+4+5
Hasil Penjumlahan 1 2 3 4 5 adalah 15
PS D:\AlogaritmaP2>
```

#### Deskripsi Program

Alogaritma dan cara kerja

Menentukan lima variabel integer (a, b, c, d, e) untuk memasukkan angka serta variabel hasil untuk hasil penjumlahan. Input dari user program ditunjukkan oleh pesan "Inputkan angka di layar. Mendeklarasikan lima variabel integer (a, b, c, d, e) untuk menampung angka yang diinputkan,

serta variabel hasil untuk menyimpan hasil penjumlahan. Meminta Input dari Pengguna Program menampilkan pesan inputkan angka di layar. Seterusnya, pengguna diingatkan untuk memasukkan lima angka integer dan nilainya akan dirubah secara serentak dengan `fmt.Scanln(&a, &b, &c, &d, &e)`. Penjumlahan setelah lima angka diterima dari pengguna, program menjumlahkan semua angka tersebut dengan melakukan operasi hasil  $a+b+c+d+e$  Nilai hasil penjumlahan simpan di variabel hasil. Meminta Input dari Pengguna Program menampilkan pesan Inputkan angka di layar. Pengguna kemudian diminta untuk memasukkan lima angka integer, yang akan dibaca sekaligus oleh program menggunakan `fmt.Scanln(&a, &b, &c, &d, &e)`. Menghitung penjumlahan, setelah lima angka diterima dari pengguna, program menjumlahkan semua angka tersebut dengan melakukan operasi hasil  $a+b+c+d+e$  nilai hasil penjumlahan disimpan di variabel hasil. Menampilkan hasil penjumlahan program ini mencetak pesan yang berisi angka yang dimasukkan oleh pengguna, dan hasil penjumlahan dari lima angka tersebut. Contohnya, jika pengguna mengisi angka 3, 5, 7, 9, dan 11, maka output yang akan dihasilkan akan menunjukkan hasil penjumlahan dari angka-angka yang diinput di dalamnya. Nampung angka yang diinputkan, serta variabel hasil untuk menyimpan hasil penjumlahan. Meminta Input dari Pengguna Program menampilkan pesan inputkan angka di layar. Pengguna kemudian diminta untuk memasukkan lima angka integer, yang akan dibaca sekaligus oleh program menggunakan `fmt.Scanln(&a, &b, &c, &d, &e)`. Menghitung penjumlahan, setelah lima angka diterima dari pengguna, program menjumlahkan semua angka tersebut dengan melakukan operasi hasil  $a+b+c+d+e$  nilai hasil penjumlahan disimpan di variabel hasil. Menampilkan hasil penjumlahan program mencetak pesan yang menampilkan angka-angka yang dimasukkan oleh pengguna, serta hasil dari penjumlahan kelima angka tersebut.

#### Output yang Dihasilkan

Jika pengguna memasukkan angka 3, 5, 7, 9, dan 11, maka outputnya akan menampilkan hasil penjumlahan dari angka yang dimasukkan diawal.

4. Diberikan sebuah nilai akhir mata kuliah (NAM) [0...100] dan standar penilaian data 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

### Sourcecode

```
package main

import "fmt"

func main() {
    var nam float32
    var nmk string

    // Meminta input nilai
    fmt.Print("Masukkan nilai : ")
    fmt.Scan(&nam)

    // Logika penentuan nilai huruf berdasarkan nilai
    numerik
    if nam > 80 {
        nmk = "A"
    } else if nam > 72.5 {
        nmk = "B"
    } else if nam > 65 {
        nmk = "C"
    } else if nam > 50 {
        nmk = "D"
    } else if nam > 40 {
        nmk = "E"
    } else {
        nmk = "F"
    }

    // Menampilkan hasil
    fmt.Printf("Nilai Indeks untuk nilai %.2f adalah
    %s\n", nam, nmk)
}
```



## Screenshoot Output



```
PS D:\AlogaritmaP2> go run "d:\AlogaritmaP2\Modul3g3.go"
Masukkan nilai : 82
Nilai Indeks untuk nilai 82.00 adalah A
PS D:\AlogaritmaP2>
```

## Deskripsi Program

Program ini melakukan alogaritma dan cara kerjanya yaitu Inisialisasi Variabel: Program dimulai dengan meng declaret dua variable, yaitu varibale numerik: nam dan ordinal : nmk. Meminta Input dari Pengguna: Program ini memerintahkan di layar untuk menuliskan Masukkan nilai: kemudian program menunggu untuk mengambil nilai yang diinput pada Tombol angka saja. Sejumlah nilai yang diimportst oleh pengguna akan disimpan di dalam variabel yaitu: nam. Menjalankan Kondisi if-else: Setelah nilai dimasukkan, program akan menjalankan beberapa kondisi untuk mengetahui berapa besar nilai tersebut, serta mengubah nya menjadi nilai huruf di rentang tertentu:

Jika nilai lebih dari 80, maka pengguna mendapat “A”.

For values between 72.5 and 80, the user gets the ‘B’.

Sekiranya nilai terletak antara 65 sampai 72.5, maka pengguna diberikan “C”.

Jika nilai pada rentang 50 hingga 65, pengguna mendapatkan “D”.

Jika nilai antara 40 hingga 50, maka pengguna diberikan “E”.

Jika nilai kurang dari atau sama dengan 40, maka pengguna akan mendapatkan “F”.

Menampilkan Hasil: Sesudah nilai huruf tersebut diberikan, program menampilkan hasil yang berbentuk nilai numerik yang diinput oleh pengguna beserta nilai grade yang sesuai dalam format Di mana, X.XX merupakan nilai numerik yang input oleh pengguna dan Y adalah grade. Output yang Dihasilkan

Jika pengguna memasukkan nilai 85, maka outputnya: A

Jika pengguna memasukkan nilai 68, maka outputnya: C

Jika pengguna memasukkan nilai 35, maka outputnya: F

## III. UNGUIDED

1. Suatu pita (string) berisi kumpulan nama-nama bunga yang dipisahkan oleh spasi dan ‘–’, contoh pita diilustrasikan seperti berikut ini. Pita: mawar – melati – tulip – teratai – kamboja – anggrek Buatlah sebuah

program yang menerima input sebuah bilangan bulat positif (dan tidak nol) N, kemudian program akan meminta input berupa nama bunga secara berulang sebanyak N kali dan nama tersebut disimpan ke dalam pita. (Petunjuk: gunakan operasi penggabungan string dengan operator “+”). Tampilkan isi pita setelah proses input selesai. Buatlah sebuah program yang menerima input sebuah bilangan bulat positif (dan tidak nol) N, kemudian program akan meminta input berupa nama bunga secara berulang sebanyak N kali dan nama tersebut disimpan ke dalam pita.

N: <u>3</u>	N : <u>0</u>
Bunga 1: <u>Kertas</u>	Pita :
Bunga 2: <u>Mawar</u>	
Bunga 3: <u>Tulip</u>	
Pita: Kertas - Mawar - Tulip -	

Modifikasi program sebelumnya, proses input akan berhenti apabila user mengetikkan ‘SELESAI’. Kemudian tampilkan isi pita beserta banyaknya bunga yang ada di dalam pita Perhatikan contoh sesi interaksi program seperti di bawah ini (teks bergaris bawah adalah input/read):

Bunga 1: <u>Kertas</u>	Bunga 1: <u>SELESAI</u>
Bunga 2: <u>Mawar</u>	Pita :
Bunga 3: <u>Tulip</u>	Bunga: 0
Bunga 4: <u>SELESAI</u>	
Pita: Kertas - Mawar - Tulip -	
Bunga: 3	

package main

### Sourcecode

```

IV package main
V
VI import (
II  "fmt"
II  "strings"
IX )
X
XI func main() {
II  var banyakBunga int
II  var bunga []string
IV
XV  for {
VI      var input string

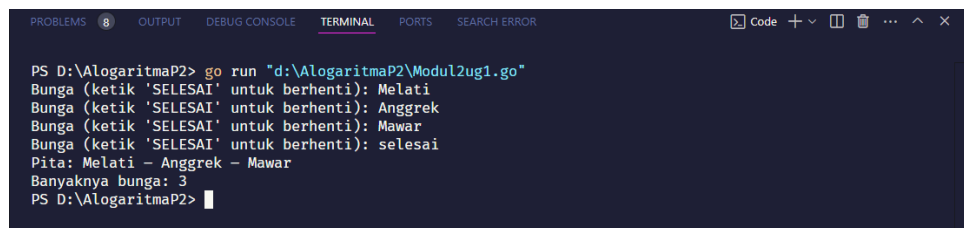
```

```

II      fmt.Println("Bunga (ketik 'SELESAI' untuk
berhenti): ")
II      fmt.Scanln(&input)
IX
XX      if strings.ToUpper(input) == "SELESAI" {
XI          break
II      }
II
IV      bunga = append(bunga, input)
XV      banyakBunga++
VI      }
II
II      fmt.Println("Pita:", strings.Join(bunga, " -
"))
IX      fmt.Println("Banyaknya bunga:", banyakBunga)
XX      }

```

## Screenshoot Output



```

PS D:\AlogaritmaP2> go run "d:\AlogaritmaP2\Modul2ug1.go"
Bunga (ketik 'SELESAI' untuk berhenti): Melati
Bunga (ketik 'SELESAI' untuk berhenti): Anggrek
Bunga (ketik 'SELESAI' untuk berhenti): Mawar
Bunga (ketik 'SELESAI' untuk berhenti): selesai
Pita: Melati - Anggrek - Mawar
Banyaknya bunga: 3
PS D:\AlogaritmaP2>

```

## Deskripsi Program

Alogaritma dan cara kerja yaitu pembentukan variabel banyakBunga di mulai dengan memo set sebesar 0 untuk mengelola penjumlahan bunga yang disisipkan. Variable slice bunga di deklarasikan kosong, karena dapat menampung setiap nama bunga yang diinputkan. Program memasuki loop tak terbatas dan menampilkan pesan "Bunga (ketik 'SELESAI' untuk berhenti):". Nama bunga diinputkan oleh pengguna melalui input yang diambil dari fungsi `fmt.Scanln(&input)`. Jika input dari pengguna adalah sebuah string "SELESAI" (baik merupakan huruf besar maupun kecil dikarenakan dengan `strings.ToUpper()` akan membuat sebuah string menjadi menjadi besar semua), maka program akan keluar dari loop dengan menggunakan perintah `break`. Menambahkan Bunga ke Slice: Namun jika input bukan "SELESAI" nama bunga akan dice topping pada slice bunga menggunakan `append(bunga, input)`. Variabel banyakBunga ditambah satu untuk penentuan jumlah bunga yg diisu. Bergantung dengan sistem bahasa, setelah keluar dari loop, program menggunakan

string.Join() untuk menggabungkan seluruh elemen bunga menjadi sebuah kalimat berupa string yang memisahkan dalam satu tanda “-”. Program terakhir ini menyediakan daftar bunga dan jumlah bunga yang dimasukkan ke dalam program tersebut.

Output yang Dihasilkan

Pengguna dapat memasukkan bunga sebanyak yang diinginkan, dan mengetik "SELESAI" untuk berhenti. Setelah itu, program menampilkan semua bunga yang dimasukkan dalam satu baris, dipisahkan oleh tanda "-", serta jumlah total bunga yang dimasukkan.

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

```
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: 2 6
Masukan berat belanjaan di kedua kantong: 9 5.8
Proses selesai.
```

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.

Pada modifikasi program tersebut, program akan menampilkan true jika selisih kedua isi kantong lebih dari atau sama dengan 9 kg. Program berhenti memproses apabila total berat isi kedua kantong melebihi 150 kg atau salah satu kantong beratnya negatif. Perhatikan contoh sesi interaksi program seperti di bawah ini (teks bergaris bawah adalah input/read):

```
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.
```

## Sourcecode

```
package main

import (
    "bufio"
    "fmt"
    "os"
    "strconv"
    "strings"
)

func main() {
    reader := bufio.NewReader(os.Stdin)

    for {
        berat1, berat2 := getBerat(reader)
        if berat1 < 0 || berat2 < 0 {
            fmt.Println("Sepeda motor pak Andi akan  
oleng: false")
            break
        }
        if isOleng(berat1, berat2) {
            fmt.Println("Sepeda motor pak Andi akan  
oleng: true")
        } else {
            fmt.Println("Sepeda motor pak Andi akan  
oleng: false")
        }
    }

    fmt.Println("Proses selesai.")
}

func getBerat(reader *bufio.Reader) (float64, float64) {
    fmt.Print("Masukan berat belanjaan di kedua kantong:  
")
    input, _ := reader.ReadString('\n')
    input = strings.TrimSpace(input)
    berat := strings.Split(input, " ")

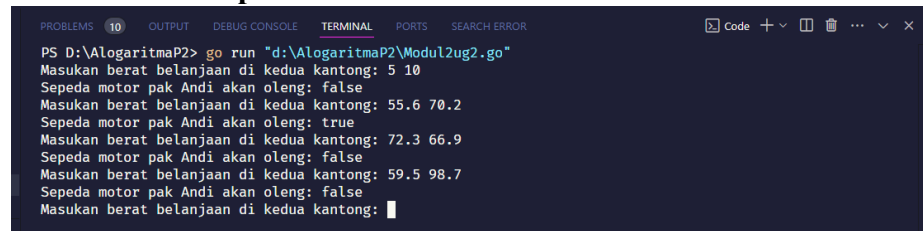
    if len(berat) != 2 {
        fmt.Println("Input tidak valid. Masukkan dua  
angka.")
        return -1, -1 // Indikasi input tidak valid
    }

    berat1, _ := strconv.ParseFloat(berat[0], 64)
    berat2, _ := strconv.ParseFloat(berat[1], 64)

    return berat1, berat2
}
```

```
func isOlang(berat1, berat2 float64) bool {
    if berat1+berat2 > 150 {
        return false
    }
    selisih := berat1 - berat2
    if selisih < 0 {
        selisih = -selisih
    }
    return selisih >= 9
}
```

## Screenshoot Output



```
PS D:\AlogaritmaP2> go run "d:\AlogaritmaP2\Modul2ug2.go"
Masukan berat belanja di kedua kantong: 5 10
Sepeda motor pak Andi akan oleng: false
Masukan berat belanja di kedua kantong: 55.6 70.2
Sepeda motor pak Andi akan oleng: true
Masukan berat belanja di kedua kantong: 72.3 66.9
Sepeda motor pak Andi akan oleng: false
Masukan berat belanja di kedua kantong: 59.5 98.7
Sepeda motor pak Andi akan oleng: false
Masukan berat belanja di kedua kantong: 
```

## Deskripsi Program

Program dan algoritma yang digunakan mulai dari loop tak terbatas meminta pengguna untuk memasukkan dua berat (berat di kantong kanan dan kiri motor). Ini adalah input string, di mana berat-berat tersebut dibaca sebagai suatu string dan kemudian diurai dan dikonversi ke dalam tipe data float64. Input input yang tidak dalam format dua angka akan dikategorikan tidak valid oleh program dan kemudian tampilan pesan error dan mengambil inputan kembali. Setelah mendapatkan input yang valid, program memeriksa apakah sepeda motor oleng atau tidak dengan dua kondisi, jika berat gabungan dari kedua kantong melebihi 150 kg, maka motor dianggap tidak oleng. Jika berat kantong kiri besar dan kanan kecil atau variasi maximum antara kedua kantong lebih dari 9 kg, maka motor dikategorikan oleng. Apabila pengguna memasukkan angka negatif, maka program keluar dari loop dan menampilkan pesan akhir "Proses selesai."

### Output yang Dihasilkan

Program ini menerima input berat dari pengguna untuk dua kantong belanjaan dan mengambil keputusan apakah sepeda motor Pak Andi itu oleng atau tidak dengan menunggu berat kantong belanjaan dan total berat yang di Transfer ke Sekitar. Program juga mengecek massa maksimal (150kg) dan menggunakan kondisi untuk memastikan bahwa motor tetap seimbang ketika membawa belanjaan.

3.

Diberikan sebuah persamaan sebagai berikut:

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

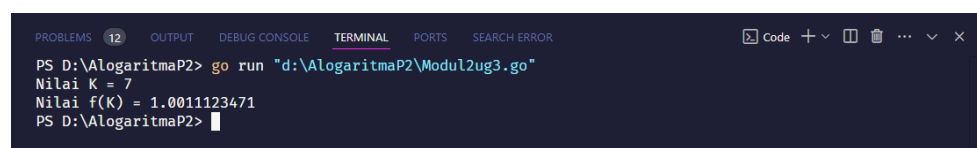
Buatlah sebuah program yang menerima input sebuah bilangan sebagai K, kemudian menghitung dan menampilkan nilai f(K) sesuai persamaan di atas. Perhatikan contoh sesi interaksi program seperti di bawah ini (teks bergaris bawah adalah input/read):

```
Nilai K = 100  
Nilai f(K) = 1.0000061880
```

### Sourcecode

```
package main  
  
import (  
    "fmt"  
)  
  
func main() {  
  
    var k float64  
  
    fmt.Print("Nilai K = ")  
    fmt.Scanln(&k)  
  
    f_k := (4*k + 2) * (4*k + 2) / ((4*k + 1) * (4*k + 3))  
  
    fmt.Printf("Nilai f(K) = %.10f\n", f_k)  
}
```

### Screenshoot Output



```
PROBLEMS 12 OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR  
PS D:\AlogaritmaP2> go run "d:\AlogaritmaP2\Modul2ug3.go"  
Nilai K = 7  
Nilai f(K) = 1.0011123471  
PS D:\AlogaritmaP2>
```

### Deskripsi Program

Alogaritma dan cara kerjanya program dimulai dengan deklarasi variabel k di mana-mana hingga untuk menampung inputan dari pengguna.

Menkripsi dan memecahkan program menampilkan pesan “Nilai K = ” untuk meminta pengguna memasukkan nilai k. Input ini digunakan oleh fungsi `fmt.Scanln(&k)` dan dimasukkan dalam variabel k.

Menentukan Pemilihan Penyandian program bersifat menentukan penyandian dalam memilih k nilai yang diinputkan. Perhitungan dilakukan sebagai berikut pertama, bagian pembilang dihitung:  $(4*k + 2) * (4*k + 2)$ . Kedua, bagian penyebut dihitung:  $(4*k + 1) * (4*k + 3)$ . Ini dia hasil pembagian pembilang dengan penyebut yang disimpan dalam variabel `f_k`. Setelah perhitungan selesai, nilai `f_k` ditampilkan dengan presisi 10 angka di belakang koma dengan menggunakan fungsi `fmt.Printf` pada Menampilkan Hasil.

Output yang dihasilkan

Program ini menggunakan operasi aritmatika dasar untuk perhitungan dan kemudian menampilkan hasil yang precision hingga dua puluh sukuf desimal.

**4. 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):



1	<p>Contoh #1</p> <p>Berat parcel (gram): <b><u>8500</u></b></p> <p>Detail berat: 8 kg + 500 gr</p> <p>Detail biaya: Rp. 80000 + Rp. 2500</p> <p>Total biaya: Rp. 82500</p>
2	<p>Contoh #2</p> <p>Berat parcel (gram): <b><u>9250</u></b></p> <p>Detail berat: 9 kg + 250 gr</p> <p>Detail biaya: Rp. 90000 + Rp. 3750</p> <p>Total biaya: Rp. 93750</p>
3	<p>Contoh #3</p> <p>Berat parcel (gram): <b><u>11750</u></b></p> <p>Detail berat: 11 kg + 750 gr</p> <p>Detail biaya: Rp. 110000 + Rp. 3750</p> <p>Total biaya: Rp. 110000</p>

### Sourcecode

```
package main

import "fmt"

func hitungBiaya(beratGram int) (int, int, int) {
    beratKg := beratGram / 1000
    sisaGram := beratGram % 1000

    biayaKg := beratKg * 10000

    var biayaSisa int
    if sisaGram >= 500 {
        biayaSisa = sisaGram * 5
    } else if sisaGram > 0 && beratKg <= 10 {
        biayaSisa = sisaGram * 15
    } else {
        biayaSisa = 0
    }

    return biayaKg, biayaSisa, beratKg
}

func main() {
    var beratGram int
    fmt.Print("Masukkan berat parcel (dalam gram): ")
    fmt.Scanln(&beratGram)
```

```

        biayaKg, biayaSisa, beratKg :=
hitungBiaya(beratGram)

        fmt.Printf("Total berat: %d kg %d gram\n", beratKg,
beratGram%1000)
        fmt.Printf("Biaya pengiriman: Rp. %d\n",
biayaKg+biayaSisa)
        fmt.Printf("    - Biaya per kg: Rp. %d\n", biayaKg)
        fmt.Printf("    - Biaya sisa gram: Rp. %d\n",
biayaSisa)
    }

```

## Screenshoot Output

```

PS D:\AlogaritmaP2> go run "d:\AlogaritmaP2\Modul2ug4.go"
Masukkan berat parcel (dalam gram): 45
Total berat: 0 kg 45 gram
Biaya pengiriman: Rp. 675
    - Biaya per kg: Rp. 0
    - Biaya sisa gram: Rp. 675
PS D:\AlogaritmaP2>

```

## Deskripsi Program

Program meminta pengguna memasukkan berat parcel dalam gram melalui input `fmt.Scanln(&beratGram)`. Menghitung Berat dan Biaya Pengiriman program memanggil fungsi `hitungBiaya(beratGram)` untuk menghitung berat dalam kilogram, sisa gram, dan biaya pengiriman.

Output hasilnya Program ini menghitung biaya pengiriman berdasarkan berat parcel dalam gram, dengan tarif tertentu untuk berat kilogram penuh dan sisa gram. Program memisahkan perhitungan biaya per kilogram dan biaya untuk sisa gram, serta menampilkan hasil perhitungan kepada pengguna dengan detail yang jelas. Berat kilogram (`beratKg`) dihitung dengan membagi berat dalam gram dengan 1000. Sisa gram (`sisaGram`) adalah hasil modulus dari berat dalam gram terhadap 1000. Biaya untuk kilogram dihitung dengan mengalikan berat kilogram dengan Rp. 10.000. Biaya sisa gram tergantung pada nilai sisa gram dan berat kilogram. Jika sisa gram lebih dari atau sama dengan 500 gram, maka biaya per gram adalah Rp. 5. Jika sisa gram kurang dari 500 gram dan berat total kurang dari 10 kg, biaya per gram adalah Rp. 15. Jika berat total lebih dari 10 kg, tidak ada biaya tambahan untuk sisa gram.

### Menampilkan Output

Setelah biaya dihitung, program menampilkan. Total berat parcel dalam kilogram dan sisa gram.

Biaya pengiriman total (biaya kilogram + biaya sisa gram).

Rincian biaya per kilogram dan biaya sisa gram.

5.

Buatlah program yang menerima input sebuah bilangan bulat  $b$  dan  $b > 1$ . Program harus dapat mencari dan menampilkan semua faktor dari bilangan tersebut!

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

### Sourcecode

```
package main

import (
    "fmt"
)

func main() {

    var b int

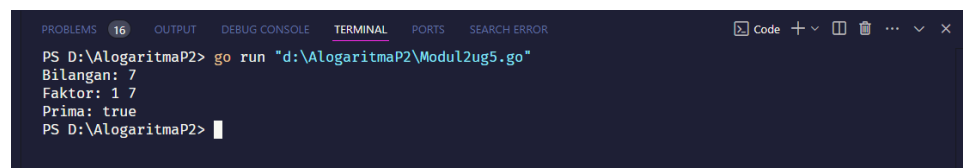
    fmt.Print("Bilangan: ")
    fmt.Scanln(&b)

    fmt.Print("Faktor: ")
    for i := 1; i <= b; i++ {
        if b%i == 0 {
            fmt.Print(i, " ")
        }
    }

    prima := true
    if b <= 1 {
        prima = false
    } else {
```

```
for i := 2; i*i <= b; i++ {  
    if b%i == 0 {  
        prima = false  
        break  
    }  
}  
  
fmt.Println("\nPrima:", prima)  
}
```

## Screenshoot Output



```
PROBLEMS 16 OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR  
PS D:\AlogaritmaP2> go run "d:\AlogaritmaP2\Modul2ug5.go"  
Bilangan: 7  
Faktor: 1 7  
Prima: true  
PS D:\AlogaritmaP2>
```

## Deskripsi Program

Program memulai dengan mendeklarasikan variabel *b* untuk menampung bilangan dari input pengguna, serta variabel *prima* untuk menentukan apakah bilangan tersebut prima atau tidak. Meminta Input dari Pengguna: Program menampilkan pesan "Bilangan: " dan menunggu pengguna memasukkan sebuah bilangan integer. Bilangan yang diinput oleh pengguna disimpan dalam variabel *b*. Menampilkan Faktor-faktor Bilangan: Program menggunakan loop `for i := 1; i <= b; i++` untuk memeriksa semua angka dari 1 hingga *b*.

Jika bilangan *b* habis dibagi oleh *i* (dengan `b % i == 0`), maka *i* dianggap sebagai faktor dari *b* dan dicetak ke layar. Jika bilangan *b* kurang dari atau sama dengan 1, maka bilangan tersebut bukan prima dan variabel *prima* diatur menjadi `false`. Jika bilangan *b* lebih besar dari 1, program memeriksa pembagiannya mulai dari angka 2 hingga akar kuadrat dari *b*. Jika ada angka yang dapat membagi *b* tanpa sisa, bilangan tersebut bukan prima dan variabel *prima* diatur menjadi `false`. Jika tidak ada angka pembagi, maka bilangan tersebut adalah prima. Setelah memeriksa faktor dan bilangan prima, program menampilkan hasil dengan format `true/false`  
Output Hasilnya

Program ini menerima sebuah bilangan dari pengguna, menampilkan semua faktor dari bilangan tersebut, lalu mengecek apakah bilangan tersebut merupakan bilangan prima atau tidak. Hasilnya berupa daftar faktor dan penentuan apakah bilangan tersebut prima (`true`) atau bukan prima (`false`).

#### **IV. Kesimpulan**

Dalam bahasa pemrograman, struktur kontrol adalah perintah dengan pola tertentu yang digunakan untuk mengendalikan program. Terdapat dua jenis struktur kontrol yang utama yakni pemilihan dan pengulangan. Pentingnya memahami struktur kontrol di aplikasi adalah sebagai pengembang perangkat lunak kita tentu ingin membuat aplikasi yang canggih dan memiliki respon yang cepat dan efisien hal ini hanya bisa didapat dengan memahami percabangan dan pengulangan. Kinerja struktur kontrol ditinjau untuk mengetahui apakah firma telah digunakan secara efektif, dan meningkatkan efisiensi, keterbacaan, dan pemeliharaan kode serta mengidentifikasi redundansi dan pengoverapan yang bisa menghambat kinerja program.

## **DAFTAR PUSTAKA**

Naughton. 1996. Konsep Dasar Pemograman Java. Andi Yogyakarta.  
<https://codingstudio.id/blog/struktur-kontrol-percabangan/>  
<https://sko.dev/wiki/struktur-kontrol>