

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2
MODUL 2
REVIEW STRUKTUR KONTROL**



Oleh:

NAUFAL THORIQ MUZHAFAR

2311102078

IF-11-02

**S1 TEKNIK INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO**

2024

I. DASAR TEORI

Sintaks

Sintaks Go File Go terdiri dari bagian-bagian berikut:

- Deklarasi paket
- Paket impor
- Fungsi
- Pernyataan dan ekspresi

Contoh

```
package main

import ("fmt")


func main() {

    fmt.Println("Hello World!")

}
```

Baris 1: Dalam Go, setiap program merupakan bagian dari sebuah paket. Kita mendefinisikannya menggunakan kata kunci **package**. Dalam contoh ini, program tersebut termasuk dalam **main** package.

Baris 2: **import ("fmt")** memungkinkan kita mengimpor file yang disertakan dalam paket **fmt**.

Baris 3: Baris kosong. Go mengabaikan spasi. Adanya spasi dalam kode membuatnya lebih mudah dibaca.

Baris 4: **func main() {}** adalah sebuah fungsi. Kode apa pun di dalam kurung kurawal {} akan dieksekusi.

Baris 5: **fmt.Println()** adalah fungsi yang tersedia dari paket **fmt**. Fungsi ini digunakan untuk mengeluarkan/mencetak teks. Dalam contoh kita, fungsi ini akan mengeluarkan "Hello World!".

fmt.Println("Hello World!") adalah sebuah pernyataan. Dalam Go, pernyataan dipisahkan dengan mengakhiri baris (menekan tombol Enter) atau dengan titik koma ";". Menekan tombol Enter akan menambahkan ";" di akhir baris secara

implisit (tidak muncul dalam kode sumber). Tanda kurung kurawal kiri { tidak dapat muncul di awal baris.

Tipe Data

Tipe data merupakan konsep penting dalam pemrograman. Tipe data menentukan ukuran dan tipe nilai variabel. Go diketik secara statis, artinya setelah tipe variabel ditetapkan, variabel tersebut hanya dapat menyimpan data dengan tipe tersebut. Go memiliki tiga tipe data dasar:

- **bool**: mewakili nilai boolean dan bisa bernilai benar atau salah
- **Numerik**: mewakili tipe integer, nilai floating point, dan tipe kompleks
- **string**: mewakili nilai string

Tipe Data Boolean

Tipe data boolean dideklarasikan dengan kata kunci **bool** dan hanya dapat mengambil nilai **true** atau **false**. Nilai **default** dari tipe data boolean adalah **false**.

Tipe Data Integer

Tipe data integer digunakan untuk menyimpan bilangan bulat tanpa desimal, seperti 35, -50, atau 1345000. Tipe data integer memiliki dua kategori:

- **Signed Integer** - dapat menyimpan nilai positif dan negatif
- **Unsigned Integer** - hanya dapat menyimpan nilai non-negatif

Signed Integer, dideklarasikan dengan salah satu kata kunci **int**, dapat menyimpan nilai positif dan negatif: Go memiliki lima kata kunci/jenis signed integer:

Tipe	Size	Range
int	Tergantung pada platform: 32 bit dalam sistem 32 bit dan 64 bit dalam sistem 64 bit	<ul style="list-style-type: none">• -2147483648 hingga 2147483647 dalam system 32 bit dan• -9223372036854775808 hingga 9223372036854775807 dalam system 64 bit
int8	8 bits/1 byte	-128 hingga 127
int16	16 bits/2 byte	-32768 hingga 32767
int32	32 bits/4 byte	-2147483648 hingga 2147483647
int64	64 bits/8 byte	-9223372036854775808 hingga 9223372036854775807

Unsigned Integer, dideklarasikan dengan salah satu kata kunci **uint**, hanya dapat menyimpan nilai non-negatif: Go memiliki lima kata kunci/jenis unsigned integer:

Tipe	Size	Range
uint	Tergantung pada platform: 32 bit dalam sistem 32 bit dan 64 bit dalam sistem 64 bit	<ul style="list-style-type: none">0 hingga 4294967295 dalam system 32 bit dan0 hingga 18446744073709551615 dalam system 64 bit
uint8	8 bits/1 byte	0 hingga 255
uint16	16 bits/2 byte	0 hingga 65535
uint32	32 bits/4 byte	0 hingga 4294967295
uint64	64 bits/8 byte	0 hingga 18446744073709551615

Tipe Data Float

Tipe data float digunakan untuk menyimpan angka positif dan negatif dengan titik desimal, seperti 35,3, -2,34, atau 3597,34987. Tipe data float memiliki dua kata kunci:

Tipe	Size	Range
float32	32 bit	-3.4e+38 to 3.4e+38.
float64	64 bit	-1.7e+308 to +1.7e+308.

Tipe Data String

Tipe data **string** digunakan untuk menyimpan serangkaian karakter (teks). Nilai string harus diapit oleh tanda kutip ganda:

Mendeklarasikan/membuat Variabel

Dalam Go, ada dua cara untuk mendeklarasikan variabel:

- Dengan kata kunci **var**:
Gunakan kata kunci **var**, diikuti dengan nama dan tipe variabel
- Dengan tanda **:=**:
Gunakan tanda **:=**, diikuti dengan nilai variable

Aturan Penamaan Variabel Go

Variabel dapat memiliki nama pendek (seperti x dan y) atau nama yang lebih deskriptif (usia, harga, nama mobil, dll.).

Aturan penamaan variabel Go:

- Nama variabel harus dimulai dengan huruf atau karakter garis bawah (_)
- Nama variabel tidak dapat dimulai dengan angka
- Nama variabel hanya dapat berisi karakter alfanumerik dan garis bawah (a-z, A-Z, 0-9, dan _)
- Nama variabel peka huruf besar-kecil (usia, Usia, dan USIA adalah tiga variabel yang berbeda)
- Tidak ada batasan panjang nama variabel
- Nama variabel tidak boleh berisi spasi
- Nama variabel tidak boleh berupa kata kunci Go apa pun

Fungsi Output

Go memiliki tiga fungsi untuk mengeluarkan teks:

- `Print()`
- `Println()`
- `Printf()`

Fungsi **`Print()`** mencetak argumennya dengan format default.

Fungsi **`Println()`** mirip dengan **`Print()`**, bedanya spasi ditambahkan di antara argumen, dan baris baru ditambahkan di akhir

Fungsi **`Printf()`** pertama-tama memformat argumennya berdasarkan kata kerja pemformatan yang diberikan, lalu mencetaknya

Di sini kita akan menggunakan dua kata kerja pemformatan:

- **`%v`** digunakan untuk mencetak **nilai** argument
- **`%T`** digunakan untuk mencetak **tipe** argument

Kata kerja berikut dapat digunakan dengan tipe data integer:

- **`%b`** Basis 2
- **`%d`** Basis 10
- **`%+d`** Basis 10 dan selalu menunjukkan tanda
- **`%o`** Basis 8
- **`%O`** Basis 8, diawali 0o
- **`%x`** Basis 16, huruf kecil
- **`%X`** Basis 16, huruf besar
- **`%#x`** Basis 16, diawali 0x

- %4d Mengisi dengan spasi (lebar 4, rata kanan)
- %-4d Mengisi dengan spasi (lebar 4, rata kiri)
- %04d Mengisi dengan nol (lebar 4)

Input User

Go memiliki beberapa cara untuk menerima input dari pengguna:

- Scan()
- Scanln()
- Scanf()

Fungsi **Scan()** menerima input pengguna dalam format mentah sebagai nilai yang dipisahkan spasi dan menyimpannya dalam variabel. Baris baru dihitung sebagai spasi.

Fungsi **Scanln()** mirip dengan **Scan()**, tetapi berhenti memindai input pada baris baru (saat tombol Enter ditekan).

Fungsi **Scanf()** menerima masukan dan menyimpannya berdasarkan format yang ditentukan untuk argumennya.

Operator

Operator digunakan untuk melakukan operasi pada variabel dan nilai.

Operator	Tipe data terkait	Keterangan
+	string integer dan real	konkatenasi 2 string operasi penjumlahan
- * /	integer dan real	operasi pengurangan, perkalian, dan pembagian
%	integer	operasi sisa pembagian integer (modulo)
& ^ & ^	integer	operasi per-bit AND, OR, XOR, AND-NOT
<< >>	integer dan unsigned integer	operasi geser bit kiri/kanan sebanyak unsigned integer yang diberikan
< <= >= == !=	selain boolean	komparasi menghasilkan nilai boolean komparasi karakter sesuai dengan posisi karakter tersebut dalam tabel ASCII/UTF-16 komparasi string sesuai dengan operasi karakter per karakter, dimulai dari karakter paling kiri (awal)
&& !	boolean	operasi boolean AND, OR, dan NOT

* &	variabel apasaja	mendapatkan data dari lokasi memori dan mendapatkan lokasi variabel
-----	------------------	---

Perulangan For

Perulangan for melakukan perulangan melalui blok kode sejumlah waktu tertentu. Perulangan berguna jika Anda ingin menjalankan kode yang sama berulang-ulang, setiap kali dengan nilai yang berbeda. Setiap eksekusi perulangan disebut **iterasi**. Perulangan **for** dapat mengambil hingga tiga pernyataan:

```
for statement1; statement2; statement3 {
    // code yang akan dieksekusi untuk setiap
    iterasi
}
```

statement1 Menginisialisasi nilai penghitung loop.

statement2 Dievaluasi untuk setiap iterasi loop. Jika dievaluasi menjadi TRUE, loop berlanjut. Jika dievaluasi menjadi FALSE, loop berakhir.

statement3 Menambah nilai penghitung loop.

Percabangan

Go memiliki pernyataan kondisional berikut:

- Gunakan **if** untuk menentukan blok kode yang akan dieksekusi, jika kondisi yang ditentukan benar
- Gunakan **else** untuk menentukan blok kode yang akan dieksekusi, jika kondisi yang sama salah
- Gunakan **else if** untuk menentukan kondisi baru untuk diuji, jika kondisi pertama salah
- Gunakan **switch** untuk menentukan banyak blok kode alternatif yang akan dieksekusi

```
if condition1 {
    // code yang akan dieksekusi jika kondisi1
    bernilai true
} else if condition2 {
    // code yang akan dieksekusi jika kondisi2
    bernilai true false dan kondisi1 bernilai false
}
```

```
} else {  
    // code yang akan dieksekusi jika kedua kondisi  
    bernilai false  
}
```

Gunakan pernyataan **switch** untuk memilih satu dari banyak blok kode yang akan dieksekusi. Pernyataan **switch** di Go mirip dengan yang ada di C, C++, Java, JavaScript, dan PHP. Perbedaannya adalah ia hanya menjalankan case yang cocok sehingga tidak memerlukan pernyataan **break**.

```
switch expression {  
case x:  
    // code  
case y:  
    // code  
case z:  
    ...  
default:  
    // code  
}
```

- Ekspresi dievaluasi sekali
- Nilai ekspresi **switch** dibandingkan dengan nilai setiap **case**
- Jika ada kecocokan, blok kode terkait dieksekusi
- Kata kunci **default** bersifat opsional. Kata kunci ini menentukan beberapa kode untuk dijalankan jika tidak ada kecocokan **case**

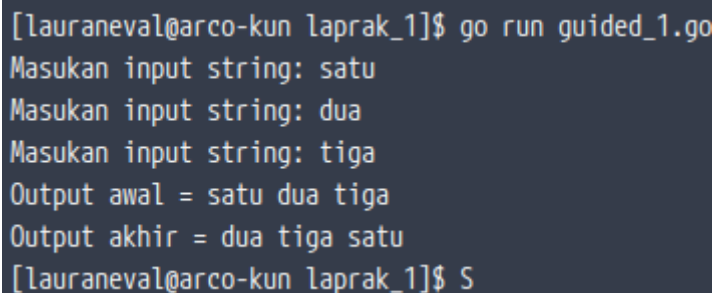
II. GUIDED

GUIDED 1

Telusuri program berikut dengan cara mengkompilasi dan mengeksekusi program. Silakan masukan data yang sesuai sebanyak yang diminta program. Perhatikan keluaran yang diperoleh. Coba terangkan apa sebenarnya yang dilakukan program tersebut?

```
1 package main
2 import "fmt"
3
4 func main() {
5     var (
6         satu, dua, tiga string
7         temp string
8     )
9     fmt.Print("Masukan input string: ")
10    fmt.Scanln(&satu)
11    fmt.Print("Masukan input string: ")
12    fmt.Scanln(&dua)
13    fmt.Print("Masukan input string: ")
14    fmt.Scanln(&tiga)
15    fmt.Println("Output awal = " + satu + " " + dua + " " + tiga)
16    temp = satu
17    satu = dua
18    dua = tiga
19    tiga = temp
20    fmt.Println("Output akhir = " + satu + " " + dua + " " + tiga)
21 }
```

Screenshoot Output



```
[lauraneval@arco-kun laprak_1]$ go run guided_1.go
Masukan input string: satu
Masukan input string: dua
Masukan input string: tiga
Output awal = satu dua tiga
Output akhir = dua tiga satu
[lauraneval@arco-kun laprak_1]$ S
```

Penjelasan Program

Pendeklarasian nama dari variable satu, dua, tiga dan temp dengan tipe data string. Lalu user diminta untuk memasukan data string ke dalam variable satu, dua, tiga yang masing masing diawali dengan output Masukan input string. Setelah itu akan muncul output awalan dari ketiga input user yangurut sesuai dengan apa yang user inputkan. Lalu variable temp menampung nilai pada variable satu, variable satu menampung nilai pada variable dua (input ke dua telah berpindah ke variabel satu), variable dua menampung nilai pada variable

tiga (input ke tiga telah berpindah ke variabel dua) dan variable tiga menampung nilai pada variable temp (input ke satu telah berpindah ke variabel tiga). Terakhir akan muncul output akhir yang menampilkan ketiga inputan user yang dimajukan satu range.

GUIDED 2

Tahun kabisat adalah tahun yang habis dibagi 400 atau habis dibagi 4 tetapi tidak habis dibagi 100. Buatlah sebuah program yang menerima input sebuah bilangan bulat dan memeriksa apakah bilangan tersebut merupakan tahun kabisat (**true**) atau bukan (**false**).

(Contoh input/output, Teks bergaris bawah adalah input dari user):

1	Tahun: <u>2016</u> Kabisat: true
2	Tahun: <u>2000</u> Kabisat: true
3	Tahun: <u>2018</u> Kabisat: false

Source Code

```
package main

import "fmt"

func main(){
    year := 0
    fmt.Print("Tahun: ")
    fmt.Scan(&year)
    fmt.Println("Kabisat: ", year%4 == 0 &&
year%1000 != 0)
}
```

Screenshoot Output

```
[lauraneval@arco-kun laprak_1]$ go run guided_2.go
Tahun: 2016
Kabisat: true
[lauraneval@arco-kun laprak_1]$ go run guided_2.go
Tahun: 2017
Kabisat: false
[lauraneval@arco-kun laprak_1]$ _
```

Penjelasan Program

Program meminta input angka tahun dari pengguna. Kemudian, mengecek apakah tahun tersebut habis dibagi 4 dan tidak habis dibagi 1000 ($\text{year \% } 1000 \neq 0$). Logika ini salah, karena seharusnya pengecekan dilakukan untuk $\text{year \% } 100 \neq 0$ (tidak habis dibagi 100), atau tahun harus habis dibagi 400. Jika kedua kondisi tersebut benar, maka program mencetak "Kabisat: true", jika tidak, mencetak "Kabisat: false".

GUIDED 3

Buat program **Bola** yang menerima input jari-jari suatu bola (bilangan bulat). Tampilkan Volume dan Luas kulit bola. $\text{volumebola} = \frac{4}{3}\pi r^3$ dan $\text{luasbola} = 4\pi r^2$ ($\pi \approx 3.1415926535$).

(Contoh input/output, Teks bergaris bawah adalah input dari user):

```
Jejari = 5
Bola dengan jejari 5 memiliki volume 523.5988 dan luas kulit 314.1593
```

Source Code

```
package main

import (
    "fmt"
    "math"
)

func main() {
    var jari float64

    fmt.Print("Jejari = ")
```

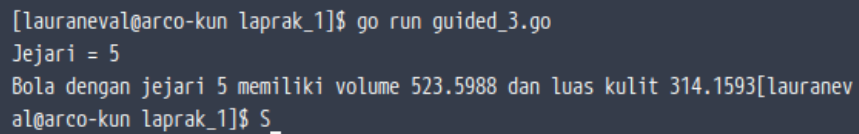
```
    fmt.Scanln(&jari)

    volume := (4.0 / 3.0) * math.Pi * math.Pow(jari,
3)

    luas := 4 * math.Pi * math.Pow(jari, 2)

    fmt.Printf("Bola dengan jejari %v memiliki
volume %.4f dan luas kulit %.4f", jari, volume, luas)
}
```

Screenshoot Output

A screenshot of a terminal window with a dark background. The text shows a user running a Go program. The output includes the input 'Jejari = 5' and the calculated volume and surface area of a sphere with radius 5, formatted to four decimal places.

```
[lauraneval@arco-kun laprak_1]$ go run guided_3.go
Jejari = 5
Bola dengan jejari 5 memiliki volume 523.5988 dan luas kulit 314.1593[lauranev
al@arco-kun laprak_1]$ S_
```

Penjelasan Program

Program meminta input nilai jari-jari bola (jari) dari pengguna lalu jalan perhitungan untuk volume bola dan luas permukaan bola. Hasil perhitungan volume dan luas permukaan bola dicetak dalam format yang rapi dengan 4 angka di belakang koma.

III. UNGUIDED

UNGUIDED 1

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 berturut-turut 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):

Percobaan 1:	<u>merah</u>	<u>kuning</u>	<u>hijau</u>	<u>ungu</u>
Percobaan 2:	<u>merah</u>	<u>kuning</u>	<u>hijau</u>	<u>ungu</u>
Percobaan 3:	<u>merah</u>	<u>kuning</u>	<u>hijau</u>	<u>ungu</u>
Percobaan 4:	<u>merah</u>	<u>kuning</u>	<u>hijau</u>	<u>ungu</u>
Percobaan 5:	<u>merah</u>	<u>kuning</u>	<u>hijau</u>	<u>ungu</u>
BERHASIL: true				
Percobaan 1:	<u>merah</u>	<u>kuning</u>	<u>hijau</u>	<u>ungu</u>
Percobaan 2:	<u>merah</u>	<u>kuning</u>	<u>hijau</u>	<u>ungu</u>
Percobaan 3:	<u>merah</u>	<u>kuning</u>	<u>hijau</u>	<u>ungu</u>
Percobaan 4:	<u>ungu</u>	<u>kuning</u>	<u>hijau</u>	<u>merah</u>
Percobaan 5:	<u>merah</u>	<u>kuning</u>	<u>hijau</u>	<u>ungu</u>
BERHASIL: false				

Source Code

```
package main

import "fmt"

func main() {
    warnaTrue := [4]string{"merah", "kuning",
    "hijau", "ungu"}

    benar := true
```

```

        for i := 1; i <= 5; i++ {

            var warna1, warna2, warna3, warna4 string

            fmt.Printf("Percobaan %d: ", i)

            fmt.Scan(&warna1, &warna2, &warna3,
&warna4)

            if warna1 != warnaTrue[0] || warna2 !=
warnaTrue[1] || warna3 != warnaTrue[2] ||
warna4 != warnaTrue[3] {

                benar = false

            }

        }

        if benar {

            fmt.Println("BERHASIL: true")

        } else {

            fmt.Println("BERHASIL: false")

        }

    }
}

```

Screenshot Output

```

[lauraneval@arco-kun laprak_1]$ go run unguided_3.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
[lauraneval@arco-kun laprak_1]$ go run unguided_3.go
Percobaan 1: merah kuning hijau ungu
Percobaan 2: merah kuning hijau ungu
Percobaan 3: merah kuning hijau ungu
Percobaan 4: merah kuning hijau biru
Percobaan 5: merah kuning hijau ungu
BERHASIL: false
[lauraneval@arco-kun laprak_1]$ SS_

```

Deskripsi Program

Program ini bertujuan untuk memeriksa apakah urutan warna yang dimasukkan oleh pengguna untuk 5 percobaan berturut-turut sesuai dengan urutan warna yang benar: merah, kuning, hijau, ungu. Program pertama-tama mendefinisikan urutan warna yang benar dalam array warnaTrue.

Kemudian, program menjalankan loop sebanyak 5 kali (untuk 5 percobaan). Dalam setiap iterasi, pengguna diminta memasukkan 4 warna. Program membandingkan input pengguna dengan urutan warna yang benar. Jika urutan warna tidak sesuai di salah satu percobaan, variabel benar akan diubah menjadi false. Setelah 5 percobaan selesai, program mencetak "BERHASIL: true" jika semua urutan warna benar, atau "BERHASIL: false" jika ada kesalahan dalam urutan warna pada salah satu percobaan.

UNGUIDED 2

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.

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

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	

Source Code

```
package main

import "fmt"

func main() {

    var bunga string

    var pita string
```



```

var i int = 1

for {

    fmt.Printf("Bunga %d: ", i)

    fmt.Scan(&bunga)

    if bunga == "SELESAI" {

        break

    }

    pita += bunga + " - "

    i ++

}

fmt.Println("Pita:", pita)

fmt.Println("Bunga:", i-1)

}

```

Screenshot Output

```

[lauraneval@arco-kun laprak_1]$ go run unguided_2.go
Bunga 1: KERTAS
Bunga 2: MAWAR
Bunga 3: TULIP
Bunga 4: SELESAI
Pita: KERTAS - MAWAR - TULIP -
Bunga: 3
[lauraneval@arco-kun laprak_1]$ go run unguided_2.go
Bunga 1: SELESAI
Pita:
Bunga: 0
[lauraneval@arco-kun laprak_1]$

```

Deskripsi Program

Program memulai dengan meminta input nama bunga satu per satu. Jika pengguna memasukkan "SELESAI", program akan menghentikan proses input. Setiap nama bunga yang dimasukkan akan ditambahkan ke variabel pita, dipisahkan dengan tanda " - ". Setelah loop berakhir, program menampilkan string gabungan bunga yang dimasukkan dan menghitung berapa banyak bunga yang telah dimasukkan (dikurangi 1 untuk tidak menghitung "SELESAI").

Output akhirnya berupa:

- Daftar bunga dalam format yang ditentukan.
- Jumlah bunga yang dimasukkan oleh pengguna.

UNGUIDED 3

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.
```

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.
```

Source Code

```
package main

import (
    "fmt"
    "math"
)

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

```

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

        fmt.Scan(&kantong1, &kantong2)

        if kantong1+kantong2 > 150 || kantong1 <
0 || kantong2 < 0 {

            fmt.Println("Proses selesai.")

            break

        }

        diff := math.Abs(kantong1 - kantong2)

        if diff >= 9 {

            fmt.Println("Sepeda motor pak Andi
akan oleng: true")

        } else {

            fmt.Println("Sepeda motor pak Andi
akan oleng: false")

        }

    }

}

```

Screenshot Output

```

[lauraneval@arco-kun laprak_1]$ go run unguided_3.go
Masukan berat belanjaan di kedua kantong: 5 10
Sepeda motor pak Andi akan oleng: false
Masukan berat belanjaan di kedua kantong: 55.6 78.2
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.
[lauraneval@arco-kun laprak_1]$ S_

```

Deskripsi Program

Program meminta pengguna memasukkan berat dari dua kantong belanjaan. Jika total berat kedua kantong lebih dari 150 atau berat salah satu kantong kurang dari 0, program menghentikan proses input dengan pesan "Proses selesai".

Program menghitung selisih absolut antara berat kedua kantong. Jika selisih berat lebih dari atau sama dengan 9, program menampilkan pesan "Sepeda motor Pak Andi akan oleng: true", yang berarti motor berpotensi oleng. Jika selisih berat kurang dari 9, program menampilkan pesan "Sepeda motor Pak Andi akan oleng: false", artinya motor tidak akan oleng.

UNGUIDED 4

Diberikan sebuah persamaan sebagai berikut ini.

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

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 = <u>100</u> Nilai $f(K)$ = 1.0000061880

$\sqrt{2}$ merupakan bilangan irasional. Meskipun demikian, nilai tersebut dapat dihamperi dengan rumus berikut:

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

Modifikasi program sebelumnya yang menerima input integer K dan menghitung $\sqrt{2}$ untuk K tersebut. Hampiran $\sqrt{2}$ dituliskan dalam ketelitian 10 angka di belakang koma.

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

1	Nilai K = <u>10</u> Nilai akar 2 = 1.4062058441
2	Nilai K = <u>100</u> Nilai akar 2 = 1.4133387072
3	Nilai K = <u>1000</u> Nilai akar 2 = 1.4141252651

Source Code

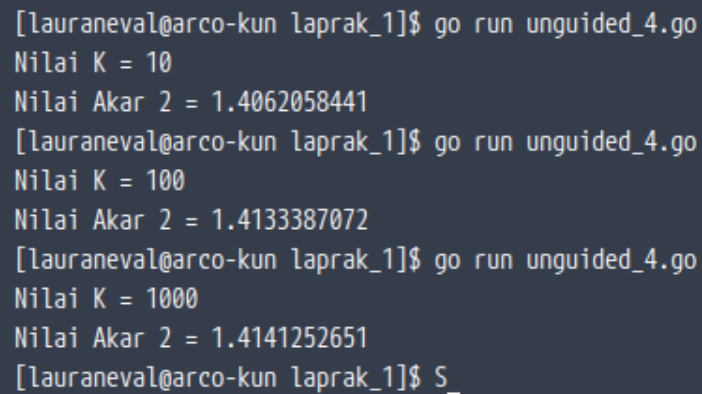
```
package main

import (
    "fmt"
    "math"
)

func main() {
    var k int
    var hasil float64 = 1.0
    fmt.Print("Nilai K = ")
    fmt.Scan(&k)
    for i := 0; i <= k; i++ {
        hasil *= math.Pow(4*float64(i)+2, 2) /
        ((4*float64(i) + 1) * (4*float64(i) + 3))
    }
```

```
        fmt.Printf("Nilai Akar 2 = %.10f \n", hasil)
    }
}
```

Screenshoot Output



```
[lauraneval@arco-kun laprak_1]$ go run unguided_4.go
Nilai K = 10
Nilai Akar 2 = 1.4062058441
[lauraneval@arco-kun laprak_1]$ go run unguided_4.go
Nilai K = 100
Nilai Akar 2 = 1.4133387072
[lauraneval@arco-kun laprak_1]$ go run unguided_4.go
Nilai K = 1000
Nilai Akar 2 = 1.4141252651
[lauraneval@arco-kun laprak_1]$ S_
```

Deskripsi Program

Program meminta pengguna untuk memasukkan nilai integer k. Variabel hasil diinisialisasi dengan nilai 1.0. Program menjalankan loop dari 0 hingga k, di mana pada setiap iterasi, nilai hasil dikalikan dengan hasil dari sebuah perhitungan sesuai rumus yang melibatkan bilangan i. Setelah perhitungan selesai, program mencetak hasil aproksimasi akar 2 dengan presisi 10 desimal. Dengan kata lain, program ini mendekati nilai akar 2 dengan menggunakan metode iteratif yang hasilnya akan semakin akurat seiring meningkatnya nilai k.

UNGUIDED 5

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	Contoh #1 Berat parcel (gram): <u>8500</u> Detail berat: 8 kg + 500 gr Detail biaya: Rp. 80000 + Rp. 2500 Total biaya: Rp. 82500
2	Contoh #2 Berat parcel (gram): <u>9250</u> Detail berat: 9 kg + 250 gr Detail biaya: Rp. 90000 + Rp. 3750 Total biaya: Rp. 93750
3	Contoh #3 Berat parcel (gram): <u>11750</u> 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, biaya, biayaTambahan, totalBiaya
int
    fmt.Print("Berat parcel (gram): ")
    fmt.Scanf("%d", &berat)
    kg := berat / 1000
    gram := berat % 1000
    biaya = kg * 10000
    if gram > 0 {
        if gram >= 500 {
            biayaTambahan = gram * 5
        } else if gram < 500 {
            biayaTambahan = gram * 15
        }
        if kg >= 10 {
            biayaTambahan = 0
        }
    }
    totalBiaya = biaya + biayaTambahan
    fmt.Printf("Berat parcel: %d kg + %d gram\n",
kg, gram)
    fmt.Printf("Detail biaya: Rp. %d + Rp. %d\n",
biaya, biayaTambahan)
    fmt.Printf("Total biaya: Rp. %d\n",
totalBiaya)
}
```

Screenshoot Output

```
[lauraneval@arco-kun laprak_1]$ go run unguided_5.go
Berat parcel (gram): 8500
Berat parcel: 8 kg + 500 gram
Detail biaya: Rp. 80000 + Rp. 2500
Total biaya: Rp. 82500
[lauraneval@arco-kun laprak_1]$ go run unguided_5.go
Berat parcel (gram): 9250
Berat parcel: 9 kg + 250 gram
Detail biaya: Rp. 90000 + Rp. 3750
Total biaya: Rp. 93750
[lauraneval@arco-kun laprak_1]$
```

Deskripsi Program

Pengguna diminta memasukkan berat parcel dalam gram. Berat parcel diubah menjadi kilogram dan sisa gram. Biaya dihitung berdasarkan kilogram, yaitu Rp. 10.000 per kilogram.

Jika ada sisa gram:

- Jika sisa gram lebih dari atau sama dengan 500 gram, biaya tambahan adalah 5 kali lipat dari sisa gram.
- Jika sisa gram kurang dari 500 gram, biaya tambahan adalah 15 kali lipat dari sisa gram.
- Jika berat parcel lebih dari atau sama dengan 10 kilogram, biaya tambahan diabaikan (dihitung 0).

Total biaya adalah jumlah dari biaya per kilogram dan biaya tambahan. Program mencetak berat parcel dalam kilogram dan gram, serta rincian biaya dan total biaya pengiriman.

UNGUIDED 6

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.

```
1 package main
2 import "fmt"
3 func main() {
4     var nam float64
5     var nmk string
6     fmt.Print("Nilai akhir mata kuliah: ")
7     fmt.Scanln(&nam)
8     if nam > 80 {
9         nam = "A"
10    }
11    if nam > 72.5 {
12        nam = "AB"
13    }
14    if nam > 65 {
15        nam = "B"
16    }
17    if nam > 57.5 {
18        nam = "BC"
19    }
20    if nam > 50 {
21        nam = "C"
22    }
23    if nam > 40 {
24        nam = "D"
25    } else if nam <= 40 {
26        nam = "E"
27    }
28    fmt.Println("Nilai mata kuliah: ", nmk)
29 }
```

Jawablah pertanyaan-pertanyaan berikut:

- Jika **nam** diberikan adalah 80.1, apa keluaran dari program tersebut? Apakah eksekusi program tersebut sesuai spesifikasi soal?
- Apa saja kesalahan dari program tersebut? Mengapa demikian? Jelaskan alur program seharusnya!
- Perbaiki program tersebut! Ujilah dengan masukan: 93.5; 70.6; dan 49.5. Seharusnya keluaran yang diperoleh adalah 'A', 'B', dan 'D'.

- a. Jika input yang diberikan ialah 80.1 maka outputnya akan memunculkan NMK A dan itu sesuai dengan NMK pada table dimana A mempunyai kondisi diatas 80.0, maka $80.1 > 80.0$ bernilai true.
- b. Kesalahan pertama ialah penggunaan **if** yang berulang, seharusnya jika menggunakan banyak kondisi maka penggunaan **else if** lebih tepat. Karena dengan menggunakan banyak if efisiensi program akan menjadi lebih lambat, setiap kondisi akan dievaluasi, meskipun sebelumnya sudah ada kondisi yang terpenuhi. Ini dapat memperlambat program jika banyak kondisi yang harus dievaluasi secara terpisah.
Kesalahan selanjutnya yaitu pada setiap statement dalam if dimana inialisasi dari NMK string menggunakan variable nam yang seharusnya variable nam menyimpan nilai bilangan bulat. Dari contoh program diatas semua variable nam dalam statement dapat diganti menjadi nmk agar tipe data yang disimpan sesuai dengan program.
- c. Penjelasan dibawah ini

Source Code

```
package main

import "fmt"

func main() {

    var nam float64

    var nmk string

    fmt.Print("Nilai akhir mata kuliah: ")

    fmt.Scanln(&nam)

    if nam > 80 {

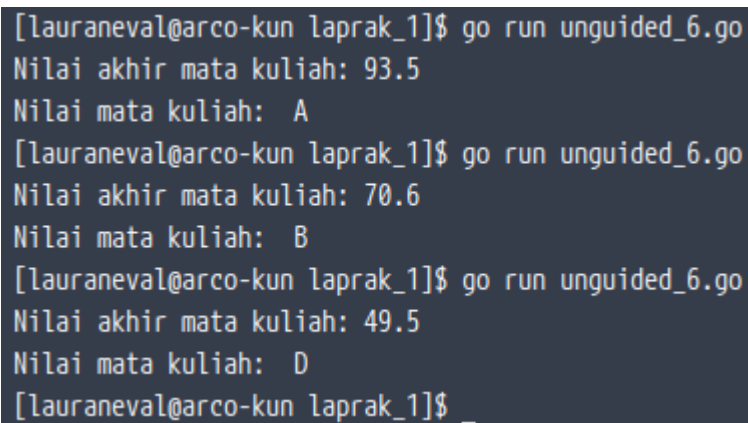
        nmk = "A"

    } 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 {  
    nmk = "E"  
}  
  
fmt.Println("Nilai mata kuliah: ", nmk)  
}
```

Screenshoot Output



```
[lauraneval@arco-kun laprak_1]$ go run unguided_6.go  
Nilai akhir mata kuliah: 93.5  
Nilai mata kuliah: A  
[lauraneval@arco-kun laprak_1]$ go run unguided_6.go  
Nilai akhir mata kuliah: 70.6  
Nilai mata kuliah: B  
[lauraneval@arco-kun laprak_1]$ go run unguided_6.go  
Nilai akhir mata kuliah: 49.5  
Nilai mata kuliah: D  
[lauraneval@arco-kun laprak_1]$ _
```

Deskripsi Program

Program meminta pengguna memasukkan nilai akhir mata kuliah (NAM) dalam bentuk angka desimal (float64). Berdasarkan nilai yang

dimasukkan, program melakukan percabangan untuk menentukan nilai mutu kuliah (NMK):

- Jika nilai > 80 , NMK = "A"
- Jika nilai > 72.5 , NMK = "AB"
- Jika nilai > 65 , NMK = "B"
- Jika nilai > 57.5 , NMK = "BC"
- Jika nilai > 50 , NMK = "C"
- Jika nilai > 40 , NMK = "D"
- Jika nilai ≤ 40 , NMK = "E"

Program menampilkan nilai mutu kuliah (NMK) yang sesuai dengan nilai akhir yang dimasukkan oleh pengguna.

UNGUIDED 7

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"

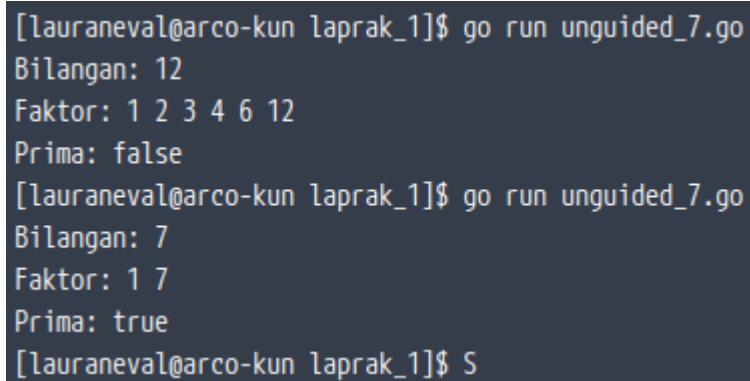
func main() {
    var b int
    var prisma bool

    fmt.Print("Bilangan: ")
    fmt.Scan(&b)
    fmt.Print("Faktor: ")
    for i := 1; i <= b; i++ {
        if b%i == 0 {
            fmt.Print(i, " ")
        }
    }
    fmt.Println()
    if b < 2 {
        prisma = false
    } else {
        prisma = true
    }
    for i := 2; i*i <= b; i++ {
        if b%i == 0 {
            prisma = false
        }
    }
}
```

```
}

if prisma == true {
    fmt.Println("Prima: true")
} else {
    fmt.Println("Prima: false")
}
}
```

Screenshoot Output



```
[lauraneval@arco-kun laprak_1]$ go run unguided_7.go
Bilangan: 12
Faktor: 1 2 3 4 6 12
Prima: false
[lauraneval@arco-kun laprak_1]$ go run unguided_7.go
Bilangan: 7
Faktor: 1 7
Prima: true
[lauraneval@arco-kun laprak_1]$ S
```

Deskripsi Program

Menerima input berupa bilangan bulat b . Menampilkan semua faktor dari bilangan b . Faktor adalah bilangan yang habis membagi b . Memeriksa apakah bilangan b merupakan bilangan prima. Bilangan prima adalah bilangan yang hanya memiliki dua faktor, yaitu 1 dan dirinya sendiri.

- Jika b kurang dari 2, bilangan otomatis bukan prima.
- Jika b hanya habis dibagi oleh 1 dan b sendiri, maka bilangan tersebut adalah prima.

Hasil akhir program menampilkan apakah b adalah bilangan prima dengan Prima: true atau Prima: false.