

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



Oleh :

Dimas Bagus Firmansyah

2311102002

IF 11 02

**S1 TEKNIK INFORMATIKA**  
**TELKOM UNIVERSITY PURWOKERTO**  
**2024**

# I. DASAR TEORI

## 2.1 Struktur Program Go

Dalam kerangka program yang ditulis dalam bahasa pemrograman Go, program utama selalu mempunyai dua komponen berikut:

- **package main** merupakan penanda bahwa file ini berisi program utama.
- **func main()** berisi kode utama dari sebuah program Go.  
Komentar, bukan bagian dari kode program, dan dapat ditulis di mana saja di dalam program:
- Satu baris teks yang diawali dengan garis miring ganda (//) s.d. akhir baris, atau.
- Beberapa baris teks yang dimulai dengan pasangan karakter /\* dan diakhiri dengan \*/.

Contoh sebuah program dalam Bahasa pemrograman GO (nama file hello.go).

```
package main
import "fmt"

func main() {
    var greetings = "Selamat datang di dunia DAP"
    var a, b int

    fmt.Println(greetings)
    fmt.Scanln(&a, &b)
    fmt.Printf("%v + %v = %v\n", a, b, a+b)
}
```

### 1.) Koding, Kompilasi, dan Eksekusi Go

#### Koding

- Tidak berbeda dengan penulisan program sumber dalam bahasa lain, program Go harus dibuat menggunakan penyunting teks dan disimpan dalam format teks, bukan dalam format dokumen (doc, docx, atau lainnya).
- Setiap program Go disimpan dalam file teks dengan ekstensi .go, dengan nama bebas. Sebaiknya nama file adalah nama untuk program tersebut.
- Setiap satu program lengkap Go disimpan dalam satu folder tersendiri. Nama folder merupakan nama program tersebut. Karena itu secara prinsip, satu program Go dapat dipecah dalam beberapa file dengan ekstensi .go selama disimpan dalam folder yang sama.

#### Kompilasi

Beberapa bahasa pemrograman dirancang untuk diimplementasikan sebagai interpreter dan lainnya sebagai kompiler. Interpreter akan membaca setiap baris instruksi dan kemudian langsung mengeksekusinya, dengan hanya sedikit pemeriksaan apakah penulisan keseluruhan program sudah benar atau belum. Kompiler akan memeriksa keseluruhan program sumber dan kemudian mengubahnya menjadi program eksekutabel, sehingga konsistensi penulisan (seperti penggunaan tipe data) sudah diperiksa sebelum eksekusi. Selain itu karena program dibuat menjadi eksekutabel lebih dahulu, proses optimasi dapat dilakukan sehingga program menjadi sangat efisien.

Go diimplementasikan sebagai kompiler. Berikut adalah contoh sesi yang biasa dilakukan saat mengkompilasi dan mengeksekusi program dalam bahasa Go:

- Panggil shell atau terminal (program/utilitas cmd.exe di Windows)
- Masuk ke dalam (cd) folder program (normalnya ada di C:\Users\golang\src atau yang sejenis)
- Kemudian panggil perintah go build atau go build file.go untuk mengkompilasi file.go
- Jika gagal, akan muncul pesan error yang sesuai, pelajari dengan baik pesan tersebut, perbaiki teks program sumber, kemudian ulangi proses build-nya.
- Jika berhasil maka pada folder tersebut akan dibuat program dengan nama yang sama dan diakhiri dengan .exe (untuk Windows)
- Panggil program eksekutabel tersebut dari terminal yang sama. Jangan memanggil program tersebut dengan mengklik eksekutabel tersebut dari folder karena program lain hanya berbasis teks, bukan/belum dirancang dengan tampilan Windows.

### Catatan

Semua proses terkait bahasa Go dilakukan melalui utilitas go. Beberapa opsi dengan utilitas go:

- **go build:** mengkompilasi program sumber yang ada dalam folder menjadi sebuah program.
- **go build file.go:** mengkompilasi program sumber file.go saja.
- **go fmt:** membaca semua program sumber dalam folder dan mereformat penulisannya agar sesuai dengan standar penulisan program sumber Go.
- **go clean:** membersihkan file-file dalam folder sehingga tersisa program sumber nya saja.

## 2.2 Tipe Data dan Instruksi Dasar

### 1.) Data dan Variabel

Variabel adalah nama dari suatu lokasi di memori, yang data dengan tipe tertentu dapat disimpan.

- Nama variabel dimulai dengan huruf dan dapat diikuti dengan sejumlah huruf, angka, atau garis bawah.

Contoh: **ketemu, found, rerata, mhs1, data\_2, ...**

Notasi tipe dasar	Tipe dalam Go	Keterangan
integer	int int8 int32 //rune int64 uint uint8 //byte uint32 uint64	bergantung platform 8 bit: -128..127 32 bit: -10 <sup>9</sup> ..10 <sup>9</sup> 64 bit: -10 <sup>19</sup> ..10 <sup>19</sup> bergantung platform 0..255 0..4294967295 0..(2 <sup>64</sup> -1)
real	float32 float64	32bit: -3.4E+38 .. 3.4E+38 64bit: -1.7E+308 .. 1.7E+308
boolean (atau logikal)	bool	false dan true
karakter	byte //uint8 rune //int32	tabel ASCII/UTF-8 tabel UTF-16
string	string	

- Tipe data yang umum tersedia adalah integer, real, boolean, karakter, dan string. Lihat tabel berikut ini untuk variasi tipe data yang disediakan dalam bahasa Go.
- Nilai data yang tersimpan dalam variabel dapat diperoleh dengan menyebutkan langsung nama variabelnya. Contoh: Menyebutkan nama found akan mengambil nilai tersimpan dalam memori untuk variabel found, pastinya.
- Informasi alamat atau lokasi dari variabel dapat diperoleh dengan menambahkan prefiks & di depan nama variabel tersebut. Contoh: &found akan mendapatkan alamat memori untuk menyimpan data pada found.
- Jika variabel berisi alamat memori, prefiks \* pada variabel tersebut akan memberikan nilai yang tersimpan dalam memori yang lokasinya disimpan dalam variabel tersebut.

Contoh: mem akan mendapatkan data di memori yang alamatnya tersimpan di mem. Karenanya (&found) akan mendapatkan data dari lokasi memori variabel found berada, alias sama saja dengan menyebutkan langsung found &=).

- Operasi yang dapat dilakukan terhadap tipe data di atas adalah

Operator dalam Go	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 <b>per-bit</b> 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 <b>boolean</b> AND, OR, dan NOT
* &	variabel apasaja	mendapatkan data dari lokasi memori dan mendapatkan lokasi dari variabel

Contoh :

Operasi	Hasil
"non suffi" + "cit mundo"	"non sufficit mundo"
2019.01 + 1.0102	2020.0202
2020 / 20	22.22
20.2 * 1.1	101
2020 % 1999	21
2020 & 1111	2104
2020 ^ 1111	1663
2020 >> 2	505
"minutus" < "magnus"	false
2020 >= 1234	true
! false && true	true

- Bahasa Go menganut kesesuaian tipe data yang ketat. Tipe data yang berbeda tidak boleh dicampur dalam satu ekspresi, bahkan tipe data masih yang sejenis, misalnya masih sama-sama integer (int dan int32). Untuk menyesuaikan tipe data, ada beberapa cara yang dapat dilakukan:

Casting: `tipe(data)`, mengubah tipe data dari yang diberikan ke tipe yang diinginkan.

Memanfaatkan fungsi `Sprintf` dan `Scanf` dari paket `fmt`.

Memanfaatkan fungsi-fungsi dalam paket `strconv`, seperti `Atoi`, `Itoa`, dan `ParseBool`.

Contoh :

Operasi	Hasil
<code>2020.0 % 19</code>	will be an illegal expression error
<code>int(2020.0) % 19</code>	6

Konversi tipe	Data	Tipe baru	Keterangan
<code>tipe(data)</code>	integer	integer	format data tidak berubah, hanya penyesuaian jumlah bit. Kekurangan bit diisi bit 0 di sebelah kiri (MSB)
	real	real	format data tidak berubah, hanya penyesuaian jumlah bit. Kekurangan bit, maka bit mantisa diisi bit 0.
	real	integer	format data disesuaikan dengan tipe data tujuan
	integer	real	format data disesuaikan dengan tipe data tujuan
<code>fmt.Sprintf("%v",v)</code>	any type	string	tulis output ke string
<code>fmt.Sprintf("%c",v)</code>	karakter	string	tulis karakter ke string
<code>fmt.Sscanf(s,"%v",&amp;v)</code>	string	any type	baca string ke variabel dengan tipe tertentu
<code>fmt.Sscanf(s,"%c",&amp;v)</code>	string	karakter	baca string ke variabel bertipe karakter

- Variabel harus dideklarasikan dulu sebelum digunakan. Variabel juga harus diinisialisasi dulu (diisi data) agar nilai yang tersimpan diketahui dengan jelas dan eksekusi algoritma menjadi terprediksi. Dalam bahasa Go, variabel yang tidak diinisialisasi lebih dahulu otomatis diisi dengan nilai default yang ekuivalen dengan bit 0.

Nilai 0 untuk bilangan integer

0.0E+0 untuk bilangan real

**false** untuk boolean

**Karakter NUL (lihat tabel ASCII) untuk karakter**

**"" (string kosong, string dengan panjang 0) untuk string**

**nil** untuk alamat memori

## 2.) Instruksi Dasar

Notasi instruksi dasar	Penulisan dalam bahasa Go	Keterangan
<code>v1 &lt;- e1</code> <code>v1 &lt;- v1 + e1</code> <code>v1 &lt;- v1 - e1</code> <code>v1 &lt;- v1 + 1</code> <code>v1 &lt;- v1 - 1</code>	<code>v1 = e1</code> <code>v1 += e1 // atau v1 = v1 + e1</code> <code>v1 -= e1 // atau v1 = v1 - e1</code> <code>v1++ // atau v1 = v1 + 1</code> <code>v1-- // atau v1 = v1 - 1</code>	operasi assignment, mengisi data ke lokasi memori (variabel)
<code>input(v1, v2)</code>	<code>fmt.Scan( &amp;v1, &amp;v2 )</code> <code>fmt.Scanln( &amp;v1, &amp;v2 )</code> <code>fmt.Scanf( "%v %v", &amp;v1, &amp;v2 )</code>	Pembacaan data memerlukan alamat memori ke mana data akan disimpan.
<code>output(e1, e2)</code>	<code>fmt.Print( e1, e2 )</code> <code>fmt.Println( e1, e2 )</code> <code>fmt.Printf( "%v %v\n", e1, e2 )</code>	Penulisan data memerlukan nilai data yang akan ditulis.

## 3.) Konstanta Simbolik

Konstanta dapat diberi nama untuk memudahkan mengingat maksud dan manfaat dari nilai yang diberi nama tersebut. Seperti PI untuk merepresentasikan konstanta  $\pi$ .

## 2.3 Struktur Kontrol Perulangan

Go hanya mempunyai at kunci for untuk semua jenis perulangan yang kita pelajari dalam notasi algoritma. Dua bentuk yang kita gunakan disini adalah struktur while-loop dan repeat-until.

Dalam konsep pemrograman terstruktur, setiap rancangan algoritma harus memenuhi syarat: satu pintu masuk dan satu pintu keluar. Karena itu tidak diperkenankan untuk membuat program sumber yang mempunyai struktur loop yang mempunyai pintu keluar lebih dari satu, seperti:

- Satu pintu keluar dari kondisi for dan satu lagi dari instruksi if-break
- Atau mempunyai instruksi if-break yang lebih dari satu.

### 1) Bentuk While-Loop

Bentuk while-loop memastikan setiap kali memasuki loop, ada kondisi yang harus terpenuhi (benar/true). Ini juga saat keluar dari loop, maka nilai kondisi tersebut pasti salah/false!

### 2) Bentuk Repeat-Until

Bentuk repeat-until di perulangan dilakukan terus menerus sampai kondisi keluar terpenuhi. Artinya selama kondisi belum terpenuhi (salah/false) maka perulangan akan terus dilakukan. Pada saat keluar dari loop maka niali kondisi pasti benar/true!

Contoh penggunaan bentuk repeat-until untuk mencetak deret bilangan Fibonacci:

	Notasi Algoritma	Penulisan dalam bahasa Go
1	<code>maxF &lt;- 100</code>	<code>maxF := 100</code>
2	<code>f0 &lt;- 0</code>	<code>f0 := 0</code>
3	<code>f1 &lt;- 1</code>	<code>f1 := 1</code>
4	<code>f2 &lt;- 1</code>	<code>f2 := 1</code>
5	<code>output("Bilangan pertama:", f1 )</code>	<code>fmt.Println("Bilangan pertama:", f1)</code>
6	<code>repeat</code>	<code>for selesai:=false; !selesai; {</code>
7	<code>    f0 &lt;- f1</code>	<code>    f0 = f1</code>
8	<code>    f1 &lt;- f2</code>	<code>    f1 = f2</code>
9	<code>    f2 &lt;- f1 + f0</code>	<code>    f2 = f1 + f0</code>
10	<code>    output("Bilangan berikutnya:", f1)</code>	<code>    fmt.Println("Bilangan berikutnya:", f1)</code>
11	<code>berikutnya:", f1)</code>	<code>    selesai = f2 &gt; maxF</code>
12	<code>until f2 &gt; maxF</code>	<code>}</code>

**Perhatian:** Karena pernyataan kondisi ada di bawah pada bentuk repeat-until, **apapun kondisinya, badan loop pasti akan pernah dieksekusi** minimum satu kali!

Kode Go di bawah menggunakan algoritma yang sangat mirip dengan algoritma di atas, dengan perbedaan pada digunakannya bentuk while-loop. Umumnya keluaran kedua algoritma sama, **kecuali** saat **maxF** diinisialisasi dengan nilai 0 atau lebih kecil!

## 2.4 Struktur Kontrol Percabangan

Untuk Analisa kasus, Bahasa Go mendukung dua bentuk percabangan, yaitu If-else dan switch-case.

### 1) Bentuk if-Else

Berikut ini bentuk-bentuk if-else yang mungkin dilakukan dalam Bahasa GO. Semua bentuk di bawah merupakan satu instruksi if-else-endif saja (hanya satu endif). Bentuk if-else yang bersarang (dengan beberapa endif) dapat dibentuk dengan komposisi beberapa if-else-endif tersebut.

	Notasi algoritma	Penulisan dalam bahasa Go
1	if (kondisi) then	if kondisi {
2	.. kode untuk kondisi true	.. kode untuk kondisi true
3	endif	}
4	if (kondisi) then	if kondisi {
5	.. kode untuk kondisi true	.. kode untuk kondisi true
6	else	} else {
7	.. kode untuk kondisi false	.. kode untuk kondisi false
8	endif	}
9	if (kondisi-1) then	if kondisi_1 {
10	.. kode untuk kondisi-1 true	.. kode untuk kondisi_1 true
11	else if (kondisi-2) then	} else if kondisi_2 {
12	.. kode untuk kondisi-2 true	.. kode untuk kondisi_2 true
13	.. dst. dst.	.. dst. dst.
14	else	} else {
15	.. kode jika semua kondisi	.. kode jika semua kondisi
16	.. di atas false	.. di atas false
17	endif	}

### 2) Bentuk Switch-Case

Dalam Bahasa Go ada dua variasi bentuk switch-case. Bentuk yang bisa digunakan adalah ekspresi ditulis pada perintah switch dan nilai ditulis dalam setiap label case nya. Bentuk yang kedua mempunyai switch tanpa ekspresi, tetapi setiap case boleh berisi ekspresi Boolean. Tentuknya bentuk yang kedua lebih bersifat umum, dan merupakan penyederhanaan bentuk (atau alias dari) susunan suatu **if-elsif-..-else-endif**.

	Notasi algoritma	Penulisan dalam bahasa Go
1	depend on ekspresi	switch ekspresi {
2	nilai_1:	case nilai_1:
3	.. kode jika ekspresi bernilai_1	.. kode jika ekspresi bernilai_1
4	nilai_2:	case nilai_2:
5	.. kode jika ekspresi bernilai_2	.. kode jika ekspresi bernilai_2
6	.. dst. dst.	.. dst. dst.
7	}	default:
8		.. kode jika tidak ada nilai
9		.. yang cocok dengan ekspresi
10		}
11	depend on (daftar variabel)	switch {
12	kondisi_1:	case kondisi_1:
13	.. kode jika ekspresi_1 true	.. kode jika ekspresi_1 true
14	kondisi_2:	case kondisi_2:
15	.. kode jika ekspresi_2 true	.. kode jika ekspresi_2 true
16	.. dst. dst.	.. dst. dst.
17	}	default:
18		.. jika tidak ada ekspresi
19		.. yang bernilai true
20		}

## II. GUIDED

### GUIDED 1

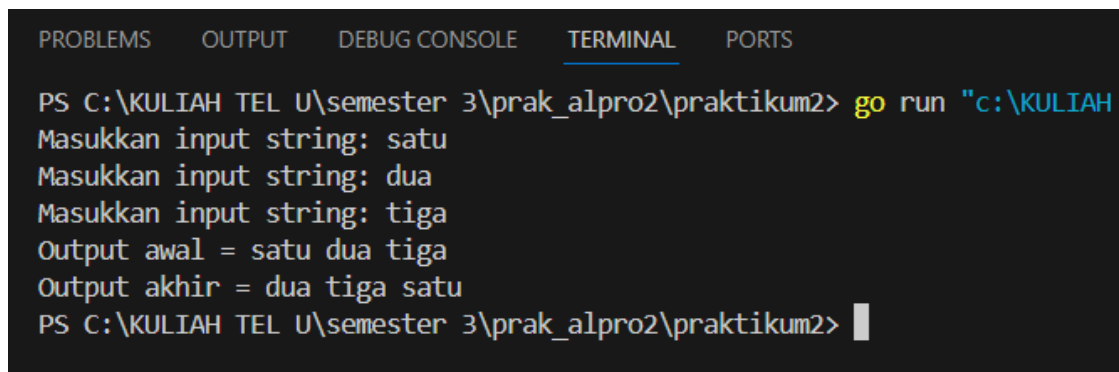
```
package main

import "fmt"

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

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

### Screenshoot Program



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2> go run "c:\KULIAH
Masukkan input string: satu
Masukkan input string: dua
Masukkan input string: tiga
Output awal = satu dua tiga
Output akhir = dua tiga satu
PS C:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2> █
```

### Deskripsi Program

Dari awal program meminta pengguna untuk menginputkan tiga string secara berturut-turu, yang disimpan di dalam variable **satu**, **dua**, **tiga**. Setelah itu, program akan mencetak ketiga string dalam urutan aslinya. String pertama yaitu satu disimpan sementara di variable temp, lalu string kedua yaitu dua akan ditempatkan di posisi string pertama, dan string ketiga yaitu tga dipindah ke posisi string kedua. Akhirnya string yang disimpan dalam temp ditempatkan di srting ketiga.



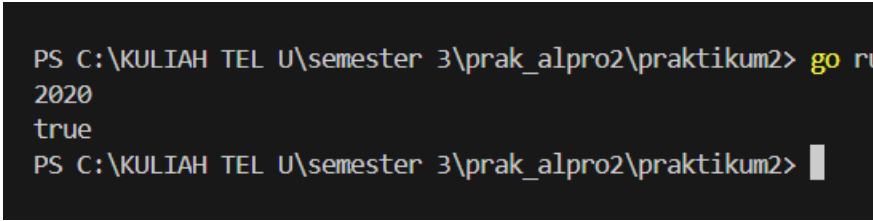
## GUIDED 2

```
package main

import "fmt"

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

### Screenshoot Program



```
PS C:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2> go run
2020
true
PS C:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2> █
```

### Deskripsi Program

Program awalnya akan meminta inputan angka dari user, yaitu tahun yang disimpan dalam variable **year**. Kemudian program mengecek apakah tahun yang diinputkan user itu habis dibagi empat dan tidak habis dibagi seratus. Jika kondisi benar maka program akan mencetak true, dan jika salah program akan menampilkan false.

## GUIDED 3

```
package main

import (
    "fmt"
    "math"
)

func main() {

    var jari, volume, luasKulit float64
    fmt.Print("Jejari = ")
    fmt.Scan(&jari)
    volume = math.Pi * (4.0 / 3.0) * math.Pow(jari, 3)
    luasKulit = 4 * math.Pi * math.Pow(jari, 2)
    fmt.Printf("Bola dengan jejari %v memiliki volume %.4f dan luas kulit %.4f\n", jari, volume, luasKulit)
}
```

## Screenshoot Program

```
PS C:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2> go run "c:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2\program1.go"
Jejari = 15
Bola dengan jejari 15 memiliki volume 14137.1669 dan luas kulit 2827.4334
PS C:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2> █
```

## Deskripsi Program

User memasukan nilai jari-jari (yang disimpan dalam variable jari), Program menggunakan rumus untuk menghitung volume dan luas permukaan bola. Volume boal dihitung dengan rumus  $\frac{4}{3} \times \pi \times \text{jari-jari}^3$ , sementara luas permukaan bola dihitung dengan rumus  $4 \times \pi \times \text{jari-jari}^2$ . Kemudian program mencetak hasil hitung luas dan volume.

### III. UNGUIDED

#### 2B

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.

#### SORCE CODE

```
package main

import (
    "fmt"
    "strings"
)

func main() {
    // Deklarasi urutan warna yang benar
    warnaBenar := [4]string{"merah", "kuning", "hijau", "ungu"}
    var berhasil bool = true

    // Loop untuk 5 kali percobaan
    for i := 1; i <= 5; i++ {
        var warnaInput [4]string
        fmt.Printf("Percobaan %d: ", i)
        for j := 0; j < 4; j++ {
            fmt.Scan(&warnaInput[j])
        }

        for k := 0; k < 4; k++ {
            if strings.ToLower(warnaInput[k]) != warnaBenar[k] {
                berhasil = false
            }
        }
    }

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

## Screenshoot Program

```
PS C:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2> g
o run "c:\KULIAH TEL U\semester 3\prak_alpro2\praktikum
2\unguided1.go"
Percobaan 1: merah kuning hijau ungu
Percobaan 2: merah kuning hijau ungu
Percobaan 3: merah kuning hijau ungu
Percobaan 4: merah kuning hijau ungu
Percobaan 5: merah kuning hijau ungu
BERHASIL: true
PS C:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2>
```

```
PS C:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2> g
o run "c:\KULIAH TEL U\semester 3\prak_alpro2\praktikum
2\unguided1.go"
Percobaan 1: merah kuning hiaju ungu
Percobaan 2: merah kuning hiaju ungu
Percobaan 3: merah kuning hiaju ungu
Percobaan 4: merah kuning hiaju ungu
Percobaan 5: unguh kuning hiaju merah
BERHASIL: false
PS C:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2> |
```

## Deskripsi Program

Program akan menerima input warna untuk 4 tabung reaksi dari user sebanyak 5 kali percobaan. Terus program menyimpan urutan warna warna dalam array **warnaBenar** . Setiap kali user memasukan warna, program akan memeriksa apakah urutannya sudah sesuai dengan urutan yang benar. Jika urutannya benar hasilnya ture, dan jika salah hasilnya akan false.

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.

## SOURCE CODE

```
package main

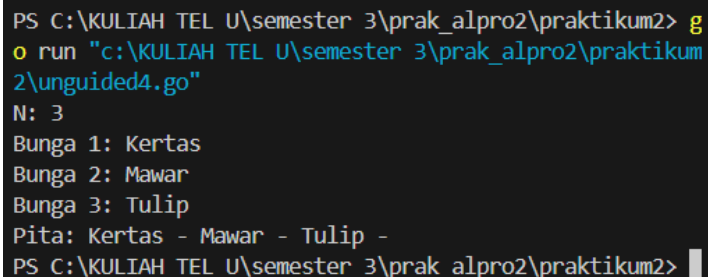
import (
    "fmt"
)

func main() {
    var N int
    var bunga, pita string

    fmt.Print("N: ")
    fmt.Scan(&N)

    if N > 0 {
        for i := 1; i <= N; i++ {
            fmt.Printf("Bunga %d: ", i)
            fmt.Scan(&bunga)
            pita += bunga + " - "
        }
        fmt.Println("Pita:", pita)
    } else {
        fmt.Println("Pita :")
    }
}
```

## Screenshoot Program



```
PS C:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2> go run "c:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2\unguided4.go"
N: 3
Bunga 1: Kertas
Bunga 2: Mawar
Bunga 3: Tulip
Pita: Kertas - Mawar - Tulip -
PS C:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2>
```

## Deskripsi Program

Program meminta inputan angka *n* dan nama Bungan sebanyak *n* ke user. Setiap nama bunganya digabungkan dengan tanda '-', lalu disimpan dalam variable **pita**. Setelah bunga dimasukan, program akan menampilkan isi pita.

Modifikasi program sebelumnya, proses input akan berhenti apabila user mengetikkan 'SELESAI'. Kemudian tampilkan isi pita beserta banyaknya bunga yang ada di dalam pita.

```

package main

import (
    "fmt"
    "strings"
)

func main() {
    var bunga, pita string
    var count int

    fmt.Println("Masukkan nama bunga (input 'SELESAI' untuk berhentiin
program):")

    for {
        fmt.Printf("Bunga %d: ", count+1)
        fmt.Scan(&bunga)

        if strings.ToLower(bunga) == "selesai" {
            break
        }

        pita += bunga + " - "
        count++
    }

    if count > 0 {
        fmt.Println("Pita:", pita)
        fmt.Printf("Banyaknya bunga dalam pita: %d\n", count)
    } else {
        fmt.Println("Pita :")
    }
}

```

### Screensoot Program

```

Masukkan nama bunga (input 'SELESAI' untuk berhentiin p
rogram):
Bunga 1: kertas
Bunga 2: mawar
Bunga 3: tulip
Bunga 4: SELESAI
Pita: kertas - mawar - tulip -
Banyaknya bunga dalam pita: 3
PS C:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2>

```

## Deskripsi Program

Program menggunakan fungsi `string.ToLower()` untuk menghentikan bahwa input selesai, dan setelah pengguna mengetik selesai, program akan berhenti dan menampilkan isi pita dan jumlah total bunga yang dimasukan.

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.

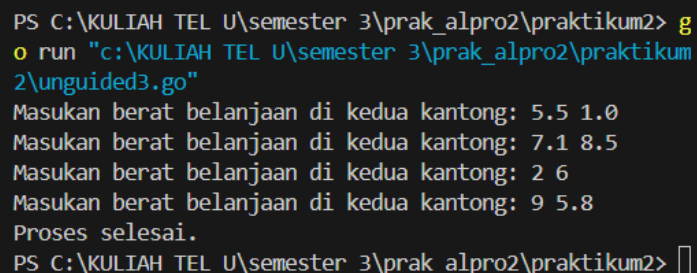
Buatlah program Pak Andi yang menerima input dua buah bilangan real positif yang menyatakan berat total masing-masing isi kantong terpal. Program akan terus meminta input bilangan tersebut hingga salah satu kantong terpal berisi 9 kg atau lebih.

## SOURCE CODE

```
package main
import (
    "fmt"
)
func main() {
    var berat1, berat2 float64
    for {
        fmt.Print("Masukan berat belanjaan di kedua kantong: ")
        fmt.Scan(&berat1, &berat2)

        if berat1 >= 9 || berat2 >= 9 {
            break
        }
    }
    fmt.Println("Proses selesai.")
}
```

## Screenshoot Program



```
PS C:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2> go run "c:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2\unguided3.go"
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.
PS C:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2> 
```

## Deskripsi Program

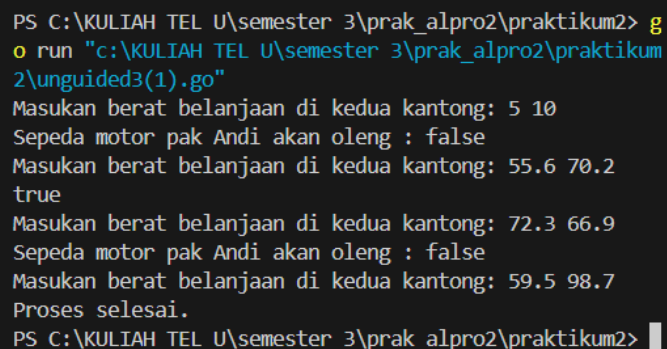
Program meminta inputan dari pengguna untuk mengisi berat belanjaan di kedua kantong menggunakan loop for. Jika salah satu kantong memiliki berat 9 kg atau lebih maka loop berhenti, dan program selesai.

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.

```
package main
import (
    "fmt"
    "math"
)
func main() {
    var berat1, berat2 float64
    for {
        fmt.Print("Masukan berat belanjaan di kedua kantong: ")
        fmt.Scan(&berat1, &berat2)
        // Hentikan program jika salah satu kantong negatif
        if berat1 < 0 || berat2 < 0 {
            break
        }

        // Hentikan program jika total berat melebihi 150 kg
        if berat1+berat2 > 150 {
            break
        }
        // Periksa selisih berat kedua kantong
        if math.Abs(berat1-berat2) >= 9 {
            fmt.Println("true")
        } else {
            fmt.Println("false")
        }
    }
    fmt.Println("Proses selesai.")
}
```

### Screenshoot Program



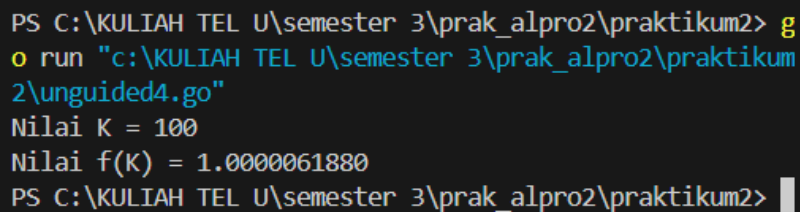
```
PS C:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2> go run "c:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2\unguided3(1).go"
Masukan berat belanjaan di kedua kantong: 5 10
Sepeda motor pak Andi akan oleng : false
Masukan berat belanjaan di kedua kantong: 55.6 70.2
true
Masukan berat belanjaan di kedua kantong: 72.3 66.9
Sepeda motor pak Andi akan oleng : false
Masukan berat belanjaan di kedua kantong: 59.5 98.7
Proses selesai.
PS C:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2>
```



#### 4. SOURCE CODE

```
package main
import (
    "fmt"
    "math"
)
func main() {
    var K float64
    // Meminta input dari pengguna untuk nilai K
    fmt.Print("Nilai K = ")
    fmt.Scan(&K)
    // Menghitung f(K) berdasarkan persamaan
    numerator := math.Pow(4*K+2, 2)
    denominator := (4*K + 1) * (4*K + 3)
    fK := numerator / denominator
    // Menampilkan hasil perhitungan
    fmt.Printf("Nilai f(K) = %.10f\n", fK)
}
```

#### Screenshoot Program



```
PS C:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2> go run "c:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2\unguided4.go"
Nilai K = 100
Nilai f(K) = 1.0000061880
PS C:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2>
```

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

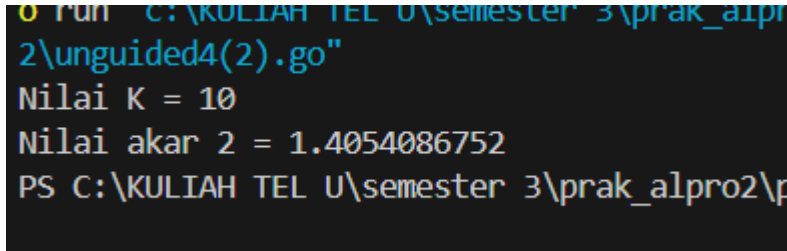
```
package main
import (
    "fmt"
    "math/big"
)
func calculateSqrt2(K int) *big.Float {
    result := big.NewFloat(1.0)
    for k := 0; k < K; k++ {
        numerator := float64((4*k + 2) * (4*k + 2))
        denominator := float64((4*k + 1) * (4*k + 3))
        term := big.NewFloat(numerator / denominator)
        result.Mul(result, term)
    }
    return result
}
func main() {
    var K int
```

```

    fmt.Print("Nilai K = ")
    fmt.Scan(&K)
    result := calculateSqrt2(K)
    resultStr := fmt.Sprintf("%.10f", result)
    fmt.Println("Nilai akar 2 =", resultStr)
}

```

### Screenshoot Program



```

PS C:\KULIAH TEL U\semester 3\prak_alpro2\p
2\unguided4(2).go"
Nilai K = 10
Nilai akar 2 = 1.4054086752
PS C:\KULIAH TEL U\semester 3\prak_alpro2\p

```

### Deskripsi program

Program ini akan meminta kamu memasukkan sebuah angka. Angka ini akan menentukan berapa kali sebuah perhitungan khusus akan diulang. Perhitungan ini bertujuan untuk mencari nilai  $\sqrt{2}$  (akar kuadrat dari 2) dengan ketelitian yang sangat tinggi, yaitu sampai 10 angka di belakang koma. Hasil perhitungan akan ditampilkan sesuai dengan jumlah pengulangan yang kamu masukkan

## UNGUIDED 2 C

### Unguided 1

```

package main

import (

    "fmt"

)

func hitungBiayaPos(berat int) (int, string, int) {

    // Konversi berat dalam gram menjadi kg dan sisanya

    kg := berat / 1000

    sisaGram := berat % 1000

    // Biaya dasar per kg adalah Rp. 10.000

```

```
biayaPerKg := 10000

biayaTambahan := 0

// Menghitung biaya pengiriman dasar berdasarkan berat dalam kg

totalBiaya := kg * biayaPerKg

// Menghitung biaya tambahan berdasarkan sisa berat gram

if sisaGram > 500 {

biayaTambahan = sisaGram * 15

} else if sisaGram > 0 {

biayaTambahan = sisaGram * 5

}

// Jika berat total lebih dari 10 kg, biaya sisa gram gratis

if kg >= 10 {

biayaTambahan = 0

}

totalBiaya += biayaTambahan

detailBerat := fmt.Sprintf("%d kg + %d gr", kg, sisaGram)

return totalBiaya, detailBerat, biayaTambahan

}

func main() {

totalBiaya, detailBerat, biayaTambahan := hitungBiayaPos(berat)

fmt.Println("Detail berat:", detailBerat)

fmt.Printf("Detail biaya: Rp. %d + Rp. %d\n", (totalBiaya - biayaTambahan),
biayaTambahan)

fmt.Println("Total biaya: Rp.", totalBiaya)

}
```

## Screenshoot Program

```
PS C:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2> go run "c:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2\unguidedc(5).go"
Berat parcel (gram): 8500
Detail berat: 8 kg + 500 gr
Detail biaya: Rp. 80000 + Rp. 2500
Total biaya: Rp. 82500
PS C:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2> |
```

## Deskripsi Program

Pengguna disuruh untuk memasukan berat parcel dan program akan menghitung biaya pengiriman berdasarkan tarif Rp. 10.000 per kilo. Jika berat lebih dari 500 gram, dikenakan biaya tambahan Rp. 15 per gram; jika kurang dari 500 gram, biaya tambahan sebesar Rp. 5 per gram diterapkan

## Nomer 2 jawaban

- jika nam adalah 80,1 keluaran dari program ini adalah AB. Eksekusi program tersebut tidak sesuai dengan spesifikasi soal karena nilai di atas 80 seharusnya mengeluarkan "A", tetapi karena kondisi dalam program hanya mengecek jika nilai sama dengan 80 atau lebih besar dari 80 pada baris 8, nilai 80.1
- Pada bagian program yang memeriksa nilai, ada kesalahan dalam urutan pengecekan nilainya. Akibatnya, beberapa nilai tidak diperiksa dengan benar. Misalkan, jika kita ingin memeriksa apakah nilai seseorang lebih dari 72.5, program ini tidak akan memeriksa dengan benar jika nilai tersebut antara 72.5 dan 80. Ini karena pengecekan nilai 80 dilakukan terlebih dahulu, sehingga nilai yang lebih kecil dari 80 tidak akan sampai pada pengecekan nilai 72.5. Masalah serupa juga terjadi pada rentang nilai lainnya.

## Seharusnya

Program ini harus memeriksa nilai siswa dari yang paling tinggi ke yang paling rendah. Kemudian, program akan memberikan nilai huruf (A, B, C, D, atau E) berdasarkan rentang nilainya. Misalnya, jika nilai siswa di atas 80, maka dia akan mendapat nilai A. Begitu juga untuk rentang nilai lainnya, seperti 72.5-80 untuk nilai B, dan seterusnya.

- Output

```
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 Program

```
PS C:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2> go
n "c:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2\ungu
edc(6).go"
Nilai akhir mata kuliah: 93.5
Nilai mata kuliah: A
PS C:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2>
```

```
PS C:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2>
n "c:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2\u
edc(6).go"
Nilai akhir mata kuliah: 70.6
Nilai mata kuliah: B
PS C:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2>
```

```
PS C:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2>
n "c:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2\u
edc(6).go"
Nilai akhir mata kuliah: 49.5
Nilai mata kuliah: D
PS C:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2>
```

## Nomer 3

```
package main

import (
    "fmt"
)

func main() {
    var b int

    fmt.Print("Masukkan bilangan bulat: ")
    fmt.Scan(&b)

    fmt.Printf("Bilangan: %d\n", b)
    fmt.Print("Faktor: ")

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

## Screenshoot Program

```
PS C:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2> go t
n "c:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2\ungu
edc(7).go"
Masukkan bilangan bulat: 12
Bilangan: 12
Faktor: 1 2 3 4 6 12
PS C:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2>
```

```
PS C:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2>
n "c:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2\u
edc(7).go"
Masukkan bilangan bulat: 7
Bilangan: 7
Faktor: 1 7
PS C:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2>
```

## Lanjutan Program

```
package main

import (
    "fmt"
)

func main() {
    var b int
    var count int

    fmt.Print("Masukkan bilangan bulat: ")
    fmt.Scan(&b)

    fmt.Printf("Bilangan: %d\n", b)
    fmt.Print("Faktor: ")

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

    if count == 2 {
        fmt.Println("Prima: true")
    } else {
        fmt.Println("Prima: false")
    }
}
```

## Screenshoot Program

```
PS C:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2> go
n "c:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2\ut
edc(7)(2).go"
Masukkan bilangan bulat: 12
Bilangan: 12
Faktor: 1 2 3 4 6 12
Prima: false
PS C:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2>
```

```
PS C:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2> go
n "c:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2\ut
edc(7)(2).go"
Masukkan bilangan bulat: 7
Bilangan: 7
Faktor: 1 7
Prima: true
PS C:\KULIAH TEL U\semester 3\prak_alpro2\praktikum2>
```

## Deskripsi Program

Jadi program meminta input bilangan bulat  $b$  dari pengguna. Lalu program mencetak factor factor dari bilangan tersebut dengan memeriksa setiap angka dari 1 hingga  $b$  dan mencetak faktor jika  $b \% f == 0$ . Yang terakhir program menghitung jumlah factor, jika hanya ada dua factor maka program bilang bahwa bilangan tersebut adalah bilangan prima( true ), sebaliknya maka bukan prima (false)