

**LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI
OBJEK**

**MODUL II
REVIEW STRUKTUR KONTROL**



Oleh :

NAMA : FAISAL KHOIRUDDIN

NIM : 2311102046

Kelas : IF-11-02

**S1 TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
2024**

I. DASAR TEORI

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 '*/'.

```
1 // Setiap program utama dimulai dengan "package main"
2 package main
3
4 // Impor paket yang dibutuhkan, "fmt" berisi proses I/O standar
5 import "fmt"
6
7 // Kode program utama dalam "fungsi main"
8 func main() {
9     ...
10 }
```

Contoh sebuah program dalam bahasa pemrograman Go (nama file hello.go).

```
1 package main
2 import "fmt"
3 func main() {
4     var greetings = "Selamat datang di dunia DAP"
5     var a, b int
6
7     fmt.Println(greetings)
8     fmt.Scanln(&a, &b)
9     fmt.Printf("%v + %v = %v\n", a, b, a+b)
10 }
```

```

C:\users\go\src\hello>dir
Directory of C:\users\jimmyt\go\src\hello
6/29/2019  7:15 PM      1,727  hello.go
C:\users\go\src\hello>go build hello.go
C:\users\go\src\hello>dir
Directory of C:\users\jimmyt\go\src\hello
6/29/2019  7:15 PM      1,727  hello.go
6/29/2019  7:18 PM  2,198,528  hello.exe
C:\users\go\src\hello>hello
Selamat datang di dunia DAP
7 5
7 + 5 = 12
C:\users\go\src\hello>

```

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

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

- Panggil shell atau terminal (program/utiliti cmd.exe di Windows) Masuk ke dalam (cd) folder program (normalnya ada di C:\Users\go\src\ atau yang sejenis)
- Kemudian panggil perintah go build atau go build file.go untuk

mengkompilasi file.go

- Jika gagal, akan muncul pesan eror 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 kalian 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. 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 garisbawah. 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 ⁹ ..10 ⁹ 64 bit: -10 ¹⁹ ..10 ¹⁹ bergantung platform 0..255 0..4294967295 0..(2 ⁶⁴ -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** 8=).

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 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 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 samasama integer (**int** dan **int32**). Untuk menyesuaikan tipe data, ada beberapa cara yang dapat dilakukan:

➤ Casting, **tipe(data)**, mengubah tipe dari data 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**. Lihat lampiran untuk contoh penggunaan.

Contoh:

Operasi	Hasil
2020.0 % 19	will be an illegal expression error
int(2020.0) % 19	6

Konversi tipe	Data	Tipe baru	Keterangan
tipe(data)	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
fmt.Sprintf("%v",v)	any type	string	tulis output ke string
fmt.Sprintf("%c",v)	karakter	string	tulis karakter ke string
fmt.Sscanf(s,"%v",&v)	string	any type	baca string ke variabel dengan tipe tertentu
fmt.Sscanf(s,"%c",&v)	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

Notasi deklarasi variabel	Penulisan dalam Go	Keterangan
kamus a : tipe	var a tipe	a diinisialisasi dengan nilai default
kamus a : tipe algoritma a <- nilai_awal	var a tipe = nilai_awal var a = (tipe)nilai_awal	a diinisialisasi dengan nilai_awal
	a := nilai_awal a := (tipe)nilai_awal	secara implisit , tipe variabel a ditentukan dari nilai inisialisasinya

Contoh:

```

1 func main() {
2     var a int
3     a = 2019
4     r := 2019.0707
5     b := false
6     c := 'x'
7     s := "a string is a string"
8 }

```

2. Instruksi Dasar

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

Contoh:

```

1 package main
2 import "fmt"
3
4 func main() {
5     var a, b, c float64
6     var hipotenusa bool
7
8     fmt.Scanln( &a, &b, &c )
9     hipotenusa = (c*c) == (a*a + b*b)
10    fmt.Println( "Sisi c adalah hipotenusa segitiga a,b,c: ", hipotenusa)
11 }

```

Contoh:


```

1 package main
2 import "fmt"
3
4 func main() {
5     var a, b, c float64
6     var hipotenusa bool
7
8     fmt.Scanln( &a, &b, &c )
9     hipotenusa = (c*c) == (a*a + b*b)
10    fmt.Println( "Sisi c adalah hipotenusa segitiga a,b,c: ", hipotenusa)
11 }

```

3. Konstanta Simbolik

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

```

1 const PI = 3.1415926535897932384626433
2 const MARKER = "AKHIR"

```

3. Struktur Kontrol Perulangan

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

Bentuk perulangan dalam bahasa Go

```

1 for inisialisasi; kondisi; update {
2     // .. for-loop ala C
3     // .. ke-3 bagian opsional, tetapi ";" tetap harus ada
4 }
5 for kondisi {
6     // .. ulangi kode di sini selama kondisi terpenuhi
7     // .. sama seperti "for ; kondisi; {"
8 }
9 for {
10    // .. tanpa kondisi, berarti loop tanpa henti (perlu if-break)
11 }
12 for ndx, var := range slice/array {
13    // .. iterator mengunjungi seluruh isi slice/array
14    // .. pada setiap iterasi ndx diset indeks dan var diisi nilainya
15 }

```

Dalam konsep pemrograman terstruktur, setiap rancangan algoritma harus memenuhi syarat satu pintu masuk dan satu pintu keluar. Karena itu **tidaklah 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, adakondisi yang harus terpenuhi (benar/**true**). Ini juga berarti saat keluar dari loop, maka nilai kondisi tersebut pasti salah/**false**!

	Notasi algoritma	Penulisan dalam bahasa Go
1	while (kondisi) do	for kondisi {
2	... kode yang diulang	.. kode yang diulang
3		
4	endwhile	}

Contoh penggunaan bentuk while-loop untuk menghitung $y = \sqrt{x}$ adalah sebagai berikut:

	Notasi algoritma	Penulisan dalam bahasa Go
1	e <- 0.000001	e := 0.000001
2	x <- 2.0	x := 2.0
3	y <- 0.0	y := 0.0
4	y1 <- x	y1 := x
5	while y1-y > e or y1-y < -e do	for y1-y > e y1-y < -e {
6	y <- y1	y = y1
7	y1 <- 0.5*y + 0.5*(x/y)	y1 = 0.5*y + 0.5*(x/y)
8	endwhile	}
9	output("sqrt(",x,")=",y)	fmt.Printf("sqrt(%v)=%v\n", x, y)

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 nilai kondisi pasti benar/true!

	Notasi Algoritma	Penulisan dalam bahasa Go
1	repeat	for selesai:=false; !selesai; {
2	.. kode yang diulang	.. kode yang diulang
3	until (kondisi)	selesai = kondisi
4		}
5		for selesai:=false; !selesai; {
6		selesai=kondisi {
7		.. kode yang diulang
8		}
9		}

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

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

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!

```

1  maxF := 100
2  f0 := 0
3  f1 := 1
4  f2 := 1
5  fmt.Println("Bilangan pertama:", f1 )
6  for f2 <= maxF {
7      f0 = f1
8      f1 = f2
9      f2 = f1 + f0
10     fmt.Println("Bilangan berikutnya:", f1 )
11 }

```

4. Struktur Kontrol Percabangan

Untuk analisa kasus, bahasa Go mendukung dua bentuk

percabangan, yaitu if-else dan switchcase.

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

Contoh konversi (nilai, tubes, kehadiran) menjadi indeks nilai.

```
1  if nilai > 75 && adaTubes {  
2      indeks = 'A'  
3  } else if nilai > 65 {  
4      indeks = 'B'  
5  } else if nilai > 50 && pctHadir > 0.7 {  
6      indeks = 'C'  
7  } else {  
8      indeks = 'F'  
9  }  
10 fmt.Printf( "Nilai %v dengan kehadiran %v%% dan buat tubes=%v, mendapat  
11 indeks %c\n", nilai, pctHadir, adaTubes, indeks )
```

Contoh konversi (nilai, tubes, kehadiran) menjadi indeks nilai

```

1  if nilai > 75 && adaTubes {
2      indeks = 'A'
3  } else if nilai > 65 {
4      indeks = 'B'
5  } else if nilai > 50 && pctHadir > 0.7 {
6      indeks = 'C'
7  } else {
8      indeks = 'F'
9  }
10 fmt.Printf( "Nilai %v dengan kehadiran %v%% dan buat tubes=%v, mendapat
11 indeks %c\n", nilai, pctHadir, adaTubes, indeks )

```

2. Bentuk Switch-Case

Dalam bahasa Go ada dua variasi bentuk switch-case. Bentuk yang biasa 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**. Tentunya bentuk yang kedua lebih bersifat umum, dan merupakan penyederhanaan bentuk (atau alias dari) susunan suatu **if-elseif-else-endif**.

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

Contoh menentukan batas nilai untuk suatu indeks:

```

1  switch indeks {
2  case 'A':
3      batasA = 100
4      batasB = 75
5  case 'B':
6      batasA = 75
7      batasB = 65
8  case 'C':
9      batasA = 65
10     batasB = 50
11 default:
12     batasA = 50
13     batasB = 0
14 }
15 fmt.Printf( "Rentang nilai %v adalah: %v..%v\n", indeks, batasB, batasA )

```

```

16 switch {
17 case nilai > 75 && adaTubes:
18     indeks = 'A'

```

```

19 case nilai > 65:
20     indeks = 'B'
21 case nilai > 50 && pctHadir > 0.7:
22     indeks = 'C'
23 default:
24     indeks = 'F'
25 }
26 fmt.Printf( "Nilai %v dengan kehadiran %v%% dan buat tubes=%v, mendapat
27 indeks %c\n", nilai, pctHadir, adaTubes, indeks )

```

II. GUIDED

Soal Latihan Modul 2A

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 }
```

Source Code

```
package main
import "fmt"

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

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

```

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

```

Screenshots Output

```

PS C:\Semester3Go\Modul 2\2A no 1> go run guidedsoal2ano1.go
Masukan input string: satu
Masukan input string: dua
Masukan input string: tiga
Output awal = satu dua tiga
Output akhir = dua tiga satu
PS C:\Semester3Go\Modul 2\2A no 1>

```

Deskripsi:

Program tersebut merupakan program mengambil tiga input dari user dan menukar bilangan yang diinput. Output program tersebut yaitu menampilkan hasil setelah penukaran nilai menjadi dua, tiga, satu.

- package main → paket utama program golang
- import "fmt" → mengimpor fmt
- func main() { → merupakan fungsi utama
- var (→ deklarasi variabel
- satu, dua, tiga string → deklarasi var satu, dua, tiga dengan tipe

data string

- `temp string` → temp dengan tipe data string
- `fmt.Print("Masukan input string: ")` → menampilkan statement untuk masukkan input string
- `fmt.Scanln(&satu)` → membaca input satu dari pengguna
- `fmt.Print("Masukan input string: ")` → menampilkan statement untuk masukkan input string
- `fmt.Scanln(&dua)` → membaca input dua dari pengguna
- `fmt.Print("Masukan input string: ")` → menampilkan statement untuk masukkan input string
- `fmt.Scanln(&tiga)` → membaca input tiga dari pengguna
- `fmt.Println("Output awal = " + satu + " " + dua + " " + tiga)` → menampilkan output awal yaitu satu, dua, tiga
- `temp = satu` → temp = satu
- `satu = dua` → satu = dua
- `dua = tiga` → dua = tiga
- `tiga = temp` → tiga = temp
- `fmt.Println("Output akhir = " + satu + " " + dua + " " + tiga)` → menampilkan output akhir setelah penukaran

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

```
package main

import "fmt"

func main() {

    var tahun int

    fmt.Printf("Tahun: ")

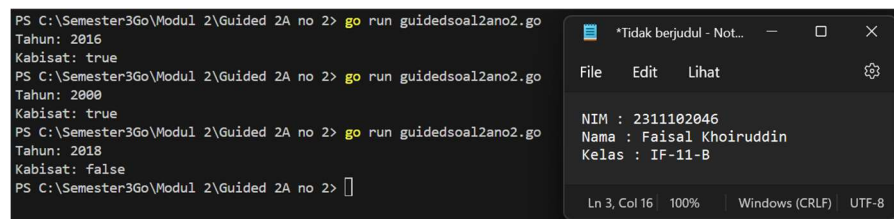
    fmt.Scan(&tahun)

    cekTahunKabisat := (tahun%4 == 0 &&
tahun%100 != 0) || tahun%400 == 0

    fmt.Println("Kabisat:",
cekTahunKabisat)

}
```

Screenshots Output



Deskripsi:

Program tersebut merupakan program menentukan tahun kabisat. Pengguna diminta menginputkan tahun contohnya tahun 2016, 2000, dan 2018. Output dari program tersebut yaitu hasil tahun kabisat yaitu tahun 2016 bernilai true, tahun 2000 bernilai true, dan tahun 2018 bernilai false.

- package main → paket utama program golang
- import "fmt" → mengimpor fmt
- func main() { → merupakan fungsi utama
- var tahun int → deklarasi variabel tahun dengan tipe data integer
- fmt.Printf("Tahun: ") → menampilkan pesan untuk menginput tahun
- fmt.Scan(&tahun) → membaca input dari user dan menyimpan ke variabel tahun
- cekTahunKabisat := (tahun%4 == 0 && tahun%100 != 0) || tahun%400 == 0 → mengecek tahun kabisat
- fmt.Println("Kabisat:", cekTahunKabisat) → menampilkan hasil true/false pada tahun kabisat

3. Buat program Bola yang menerima input jari-jari suatu bola (bilangan bulat). Tampilkan Volume dan Luas kulit bola.

$$\text{volume bola} = \frac{4}{3}\pi r^2 \text{ dan } \text{luas bola} = 4\pi r^2 \quad (\pi \approx 3.1415926535).$$

(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
```

```
package main
```

```

import (

    "fmt"

    "math"

)

func main() {

    var r float64

    fmt.Print("Jejari: ")

    fmt.Scan(&r)

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

    luasKulit := 4 * math.Pi * math.Pow(r,
2)

    fmt.Printf("Bola dengan jejari %.0f
memiliki volume %.4f dan luas kulit
%.4f\n", r, volume, luasKulit)

}

```

Screenshot

```

PS C:\Semester3Go\Modul 2\Guided 2A no 3> go run guidedsoal2ano3.go
Jejari: 5
Bola dengan jejari 5 memiliki volume 523.5988 dan luas kulit 314.1593
PS C:\Semester3Go\Modul 2\Guided 2A no 3>

```

*Tidak berjudul - Not...

File Edit Lihat

NIM : 2311102046
Nama : Faisal Khoiruddin
Kelas : IF-11-B

Ln 3, Col 16 | 100% | Windows (CRLF) | UTF-8

Deskripsi :

Program tersebut merupakan program menghitung volume dan luas kulit pada bola. Pengguna diminta menginput jari-jari sebuah bola. Output dari program tersebut yaitu hasil volume dan luas kulit sebuah bola.

- `package main` → paket utama program golang
- `import` → mengimpor
- `"fmt"` → mengimpor `fmt`
- `"math"` → mengimpor `math`
- `func main() {` → merupakan fungsi utama
- `var r float64` → deklarasi variabel `r` dengan tipe data `float64`
- `fmt.Print("Jari: ")` → menampilkan statement untuk memasukkan jari-jari
- `fmt.Scan(&r)` → membaca input pengguna dan menyimpan ke dalam variabel `r`
- `volume := (4.0 / 3.0) * math.Pi * math.Pow(r, 3)` → menghitung volume
- `luasKulit := 4 * math.Pi * math.Pow(r, 2)` → menghitung luas kulit
- `fmt.Printf("Bola dengan jejari %.0f memiliki volume %.4f dan luas kulit %.4f\n", r, volume, luasKulit)` → menampilkan bola dengan jari-jari tertentu memiliki volume dan luas kulit

III. Unguided

Soal Latihan Modul 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 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):

```
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

Percobaan 1: merah kuning hijau ungu
Percobaan 2: merah kuning hijau ungu
Percobaan 3: merah kuning hijau ungu
Percobaan 4: ungu kuning hijau merah
Percobaan 5: merah kuning hijau ungu
BERHASIL: false
```

Source Code

```
package main

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

```

func main() {
    scanner := bufio.NewScanner(os.Stdin)

    fmt.Print("N: ")
    scanner.Scan()
    N := 0
    fmt.Sscanf(scanner.Text(), "%d", &N)

    pita := ""

    for i := 1; i <= N; i++ {
        fmt.Printf("Bunga %d: ", i)
        scanner.Scan()
        namaBunga := scanner.Text()

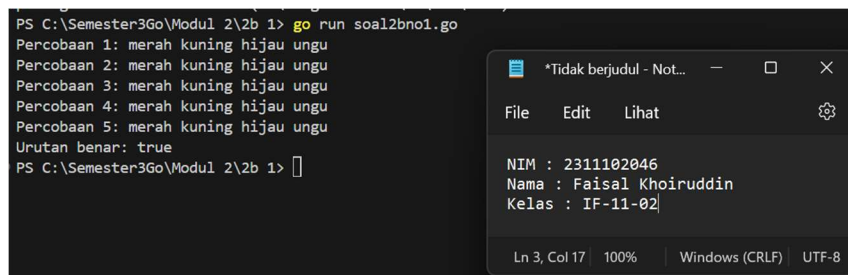
        pita += namaBunga

        if i <= N {
            pita += " - "
        }
    }

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

```

Screenshots Output



```

PS C:\Semester3Go\Modul 2\2b 1> go run soal2bno1.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
Urutan benar: true
PS C:\Semester3Go\Modul 2\2b 1>

```

Notepad window content:

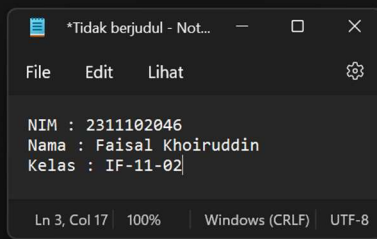
```

NIM : 2311102046
Nama : Faisal Khoiruddin
Kelas : IF-11-02

```

Terminal status bar: Ln 3, Col 17 | 100% | Windows (CRLF) | UTF-8

```
PS C:\Semester3Go\Modul 2\2b 1> go run soal2bno1.go
Percobaan 1: merah kuning hijau ungu
Percobaan 2: merah kuning hijau ungu
Percobaan 3: merah kuning hijau ungu
Percobaan 4: ungu kuning hijau merah
Percobaan 5: merah kuning hijau ungu
Urutan benar: false
PS C:\Semester3Go\Modul 2\2b 1> 
```



Deskripsi:

Program tersebut merupakan program perulangan 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.

- `package main` → paket utama program golang
- `import` → mengimpor
- `"fmt"` → mengimpor "fmt"
- `func main() {` → merupakan fungsi utama
- `var eksperimen [5][4]string` → deklarasi var eksperimen dengan array 2 dimensi ukuran 5x4
- `urutanBenar := [4]string{"merah", "kuning", "hijau", "ungu"}` → deklarasi variabel urutan benar sebagai array dengan 4 elemen berisi warna yang benar
- `for i := 0; i < 5; i++ {` → perulangan for `i := 0; i < 5; i++`
- `fmt.Printf("Percobaan %d: ", i+1)` → pencetak percobaan
- `for j := 0; j < 4; j++ {` → perulangan for dengan `j := 0; j < 4; j++`
- `fmt.Scan(&eksperimen[i][j])` → membaca input dari user dan menyimpan di `eksperimen[i][j]`
- `berhasil := true` → pengecekan berhasil `:= true`

- for _, percobaan := range eksperimen { ➔ melakukan pengecekan urutan setiap percobaan yaitu baris dan dibandingkan dengan var urutanBenar
- for i := range percobaan { ➔ perulangan for i := range percobaan
- if percobaan[i] != urutanBenar[i] { ➔ jika percobaan indeks ke- i tidak sama dengan urutanBenar[i]
- berhasil = false ➔ berhasil = false
- break ➔ break pada perulangan
- if !berhasil { ➔ jika !berhasil
- break ➔ break pada perulangan
- fmt.Println("Urutan benar:", berhasil) ➔ menampilkan Urutan benar:", berhasil

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 -	

Source Code

```
package main

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

func main() {
    scanner := bufio.NewScanner(os.Stdin)

    fmt.Print("N: ")
    scanner.Scan()
    N := 0
    fmt.Sscanf(scanner.Text(), "%d", &N)

    pita := ""

    for i := 1; i <= N; i++ {
        fmt.Printf("Bunga %d: ", i)
        scanner.Scan()
        namaBunga := scanner.Text()

        pita += namaBunga
    }
}
```

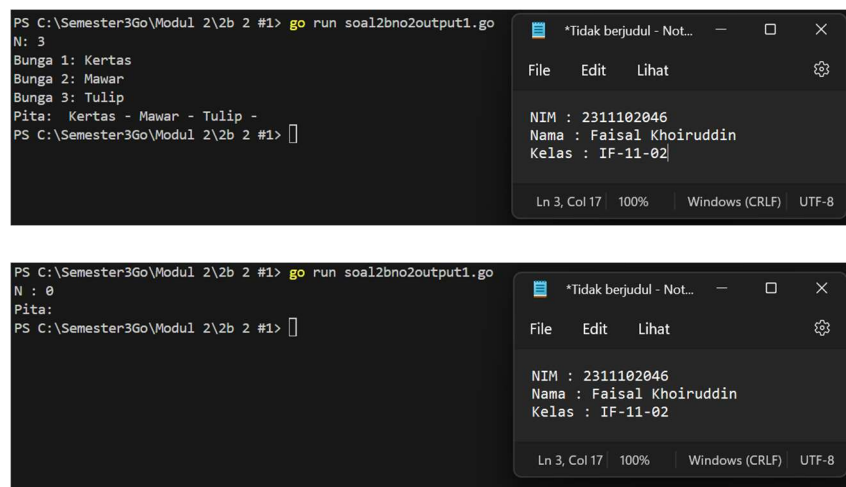
```

        if i <= N {
            pita += " - "
        }
    }

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

```

Screenshot Output



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
```

```
import (
    "bufio"

```

```
"fmt"
"os"
"strings"
)

func main() {
    scanner := bufio.NewScanner(os.Stdin)

    pita := ""

    count := 1

    for {
        fmt.Println("Bunga ", count, ": ")
        scanner.Scan()
        namaBunga := scanner.Text()

        if strings.ToUpper(namaBunga) ==
"SELESAI" {
            break
        }

        pita += namaBunga

        count++
        if count > 1 {
            pita += " - "
        }
    }

    fmt.Println("Pita: ", pita)
```

```
fmt.Println("Bunga: ", count-1)
}
```

Screenshot Output

```
PS C:\Semester3Go\Modul 2\2b 2 #2> go run soal2bno2output2.go
Bunga 1: Kertas
Bunga 2: Tulip
Bunga 3: Mawar
Bunga 4: SELESAI
Pita: Kertas - Tulip - Mawar -
Bunga: 3
PS C:\Semester3Go\Modul 2\2b 2 #2> 
```

```
PS C:\Semester3Go\Modul 2\2b 2 #2> go run soal2bno2output2.go
Bunga 1: SELESAI
Pita:
Bunga: 0
PS C:\Semester3Go\Modul 2\2b 2 #2> 
```

Deskripsi :

Program tersebut merupakan 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 dengan output tiga buah bunga yaitu kertas, mawar dan tulip serta tanda “-” sebagai pita. Kemudian program tersebut dimodifikasi menjadi proses input akan berhenti apabila user mengetikkan ‘SELESAI’. Kemudian tampilkan isi pita beserta banyaknya bunga yang ada di dalam pita.

- package main ➔ paket utama program golang
- import (➔ mengimpor
- "bufio" ➔ mengimpor "bufio"
- "fmt" ➔ mengimpor "fmt"
- "os" ➔ mengimpor "os"

- `"strings"` → mengimpor `"strings"`
- `func main() {` → merupakan fungsi utama
- `scanner := bufio.NewScanner(os.Stdin)` → membuat scanner baru untuk membaca input dari pengguna melalui `os.Stdin`
- `pita := ""` → menginisialisasi var `pita` sebagai string kosong untuk menyimpan nama-nama bunga
- `count := 1` → menginisialisasi var `count` mulai dari 1
- `for {` → perulangan `for`
- `fmt.Println("Bunga ", count, ": ")` → menampilkan nama bunga dan nomor urut bunga
- `scanner.Scan()` → membaca input dari pengguna
- `namaBunga := scanner.Text()` → mengambil teks yang dimasukkan oleh pengguna menyimpan di var `namaBunga`
- `if strings.ToUpper(namaBunga) == "SELESAI" {` → percabangan `if` jika pengguna input string `"SELESAI"`
- `break` → jika ya aka program berhenti
- `pita += namaBunga` → menggabungkan nama bunga ke var `pita`
- `count++` → `count++(increment)`
- `if count > 1 {` → jika `count > 1`
- `pita += " - "` → menambahkan tanda `"-"`
- `fmt.Println("Pita: ", pita)` → menampilkan pita bunga dan memanggil var `pita`
- `fmt.Println("Bunga: ", count-1)` → menampilkan jumlah bunga

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.

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.
```

Source Code

```
package main

import "fmt"

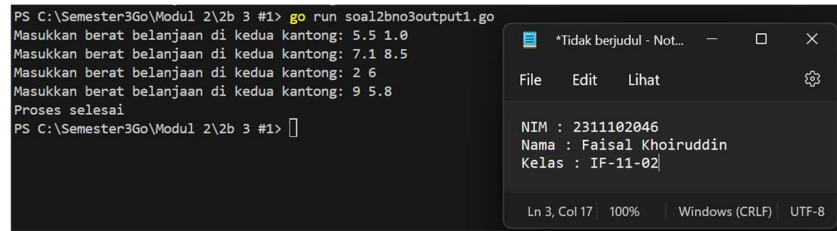
func main() {
    var beratKiri, beratKanan float64

    for beratKiri < 9 && beratKanan < 9 {
        fmt.Printf("Masukkan berat belanjaan di kedua kantong: ")
        fmt.Scan(&beratKiri, &beratKanan)

        if beratKiri >= 9 {
            fmt.Println("Proses selesai")
        }
    }
}
```

```
}  
  
}
```

Screenshot Output



The screenshot shows a terminal window with the following output:

```
PS C:\Semester3Go\Modul 2\2b 3 #1> go run soal2bno3output1.go  
Masukkan berat belanjaan di kedua kantong: 5.5 1.0  
Masukkan berat belanjaan di kedua kantong: 7.1 8.5  
Masukkan berat belanjaan di kedua kantong: 2 6  
Masukkan berat belanjaan di kedua kantong: 9 5.8  
Proses selesai  
PS C:\Semester3Go\Modul 2\2b 3 #1> 
```

Overlaid on the terminal is a Notepad window titled '*Tidak berjudul - Notepad...' containing the following text:

```
NIM : 2311102046  
Nama : Faisal Khoiruddin  
Kelas : IF-11-02
```

The Notepad window also shows status information at the bottom: 'Ln 3, Col 17 | 100% | Windows (CRLF) | UTF-8'.

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.
```

```
package main  
  
import "fmt"  
  
func main() {  
    var beratKiri, beratKanan float64  
    for (beratKiri+beratKanan) <= 150 &&  
        beratKiri >= 0 && beratKanan >= 0 {  
        fmt.Printf("Masukkan berat belanjaan di  
kedua kantong: ")
```



```

        fmt.Scan(&beratKiri, &beratKanan)

        if (beratKiri+beratKanan) > 150 ||
        beratKiri < 0 || beratKanan < 0 {
            break
        }

        sepedaMotorOlang := beratKiri-beratKanan
        >= 9 || beratKanan-beratKiri >= 9
        fmt.Printf("Sepeda motor pak Andi akan
        oleng: %v\n", sepedaMotorOlang)
    }
    fmt.Println("Proses selesai.")
}

```

Screenshot Output

```

PS C:\Semester3Go\Modul 2\2b 3 #2> go run soal2bno3output2.go
Masukkan berat belanja di kedua kantong: 5 10
Sepeda motor pak Andi akan oleng: false
Masukkan berat belanja di kedua kantong: 55.6 70.2
Sepeda motor pak Andi akan oleng: true
Masukkan berat belanja di kedua kantong: 72.3 66.9
Sepeda motor pak Andi akan oleng: false
Masukkan berat belanja di kedua kantong: 59.5 98.7
Proses selesai.
PS C:\Semester3Go\Modul 2\2b 3 #2>

```

Deskripsi :

Program tersebut merupakan 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. Kemudian program tersebut dimodifikasi menjadi 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 negative.

- `package main` → paket utama program golang
- `import "fmt"` → mengimpor "fmt"
- `func main() {` → merupakan fungsi utama
- `var beratKiri, beratKanan float64` → deklarasi `var beratKiri, beratKanan` dengan tipe data `float64`
- `for (beratKiri+beratKanan) <= 150 && beratKiri >= 0 && beratKanan >= 0 {` → perulangan `for` akan berjalan selama `(beratKiri+beratKanan) <= 150 && beratKiri >= 0 && beratKanan >= 0`
- `fmt.Printf("Masukkan berat belanja di kedua kantong: ")` → menampilkan statement untuk memasukkan berat belanja di kedua kantong
- `fmt.Scan(&beratKiri, &beratKanan)` → membaca input dari user dan menyimpan ke `beratKiri, beratKanan`
- `if (beratKiri+beratKanan) > 150 || beratKiri < 0 || beratKanan < 0 {` → percabangan jika `(beratKiri+beratKanan) > 150 || beratKiri < 0 || beratKanan < 0`
- `break` → berhenti
- `sepedaMotorOlang := beratKiri-beratKanan >= 9 || beratKanan-beratKiri >= 9` → `sepedaMotorOlang := beratKiri-beratKanan >= 9 || beratKanan-beratKiri >= 9`
- `fmt.Printf("Sepeda motor pak Andi akan oleng: %v\n", sepedaMotorOlang)` → menampilkan output Sepeda motor pak Andi akan oleng:
- `fmt.Println("Proses selesai.")` → menampilkan statement Proses selesai

4. Diberikan sebuah persamaan sebagai berikut ini.

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

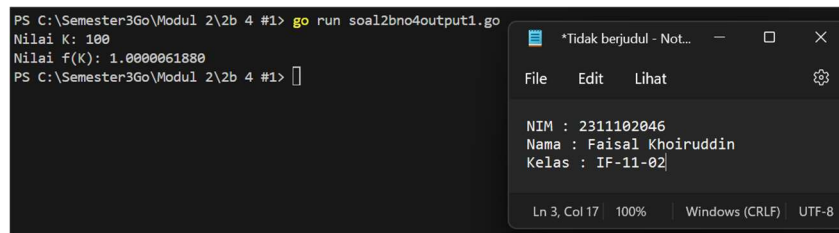
Buatlah sebuah program yang menerima input sebuah bilangan sebagai K, kemudian menghitung dan menampilkan nilai f(K) sesuai persamaan di atas. Perhatikan contoh sesi interaksi program seperti di bawah ini (teks bergaris bawah adalah input/read):

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

Source Code

```
package main  
  
import (  
    "fmt"  
    "math"  
)  
  
func main() {  
    var K float64  
    fmt.Printf("Nilai K: ")  
    fmt.Scan(&K)  
  
    pembilang := math.Pow(4*K+2, 2)  
    penyebut := (4*K + 1) * (4*K + 3)  
    fK := pembilang / penyebut  
  
    fmt.Printf("Nilai f(K): %.10f\n", fK)  
}
```

Screenshot Output



The screenshot shows a terminal window with the following output:

```
PS C:\Semester3Go\Modul 2\2b 4 #1> go run soa12bno4output1.go
Nilai K: 100
Nilai f(K): 1.0000061880
PS C:\Semester3Go\Modul 2\2b 4 #1>
```

Overlaid on the terminal is a Notepad window titled "*Tidak berjudul - Not...". It contains the following text:

```
NIM : 2311102046
Nama : Faisal Khoiruddin
Kelas : IF-11-02
```

The Notepad window also shows status information at the bottom: "Ln 3, Col 17 | 100% | Windows (CRLF) | UTF-8".

$\sqrt{2}$ merupakan bilangan irasional. Meskipun demikian, nilai tersebut dapat dihampiri 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 float64
```

```

    fmt.Print("Nilai K: ")
    fmt.Scan(&K)

    akar2 := 1.0
    for k := 0; k <= int(K); k++ {
        pembilang := math.Pow(4*float64(k)+2, 2)
        penyebut := (4*float64(k) + 1) *
(4*float64(k) + 3)
        akar2 *= pembilang / penyebut
    }
    fmt.Printf("Nilai akar 2 = %.10f\n", akar2)
}

```

Screenshot Output

```

PS C:\Semester3Go\Modul 2\2b 4 #2> go run soal2bno4output2.go
Nilai K: 10
Nilai akar 2 = 1.4062058441
PS C:\Semester3Go\Modul 2\2b 4 #2> go run soal2bno4output2.go
Nilai K: 100
Nilai akar 2 = 1.4133387072
PS C:\Semester3Go\Modul 2\2b 4 #2> go run soal2bno4output2.go
Nilai K: 1000
Nilai akar 2 = 1.4141252651
PS C:\Semester3Go\Modul 2\2b 4 #2>

```

*Tidak berjudul - Notepad

File Edit Lihat

NIM : 2311102046
Nama : Faisal Khoiruddin
Kelas : IF-11-02

Ln 3, Col 17 100% Windows (CRLF) UTF-8

Deskripsi :

Program tersebut merupakan program yang menerima input sebuah bilangan sebagai K , kemudian menghitung dan menampilkan nilai $f(K)$. Output dari program yaitu nilai input $K = 100$ maka output nilai $f(K) = 1.0000061880$. Kemudian program tersebut dimodifikasi yang sebelumnya menerima input integer K dan menghitung $\sqrt{2}$ untuk K tersebut. Hampiran $\sqrt{2}$ dituliskan dalam ketelitian 10 angka di belakang koma.

- package main → paket utama program golang
- import → mengimpor
- "fmt" → mengimpor "fmt"

- `"math"` → mengimpor `"math"`
- `func main() {` → merupakan fungsi utama
- `var K float64` → deklarasi var K bertipe data float64
- `fmt.Print("Nilai K: ")` → menampilkan statement untuk memasukkan nilai K
- `fmt.Scan(&K)` → membaca input dari pengguna dan menyimpan di variabel K
- `akar2 := 1.0` → inisialisasi var akar2 dengan nilai awal 1.0
- `for k := 0; k <= int(K); k++ {` → perulangan for k := 0; k <= int(K); k++(increment)
- `pembilang := math.Pow(4*float64(k)+2, 2)` → menghitung pembilang
- `penyebut := (4*float64(k) + 1) * (4*float64(k) + 3)` → menghitung penyebut
- `akar2 *= pembilang / penyebut` → menghitung akar2 dengan pembilang / penyebut
- `fmt.Printf("Nilai akar 2 = %.10f\n", akar2)` → menampilkan hasil dari Nilai akar 2

Soal Latihan Modul 2C

1. 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 beratParsel int
    fmt.Print("Berat parcel (gram): ")
    fmt.Scan(&beratParsel)

    berat_kg := beratParsel / 1000
    sisa_gram := beratParsel % 1000
```

```
biayaPerKg := 10000

var tambahanBiaya int
if sisa_gram >= 500 {
    tambahanBiaya = sisa_gram * 5
} else {
    tambahanBiaya = sisa_gram * 15
}

totalBiaya := (berat_kg * biayaPerKg) +
    tambahanBiaya

if berat_kg > 10 {
    totalBiaya -= tambahanBiaya
}

detailBerat := fmt.Sprintf("%d kg + %d gr",
    berat_kg, sisa_gram)
detailBiaya := fmt.Sprintf("Rp. %d + Rp.
    %d", berat_kg*biayaPerKg, tambahanBiaya)

fmt.Printf("Detail berat: %s\n",
    detailBerat)
fmt.Printf("Detail biaya: %s\n",
    detailBiaya)
fmt.Printf("Total biaya pengiriman: Rp.
    %d\n", totalBiaya)
}
```

Screenshot Output


```
PS C:\Semester3Go\Modul 2\2c 1> go run soal2cno1.go
Berat parcel (gram): 8500
Detail berat: 8 kg + 500 gr
Detail biaya: Rp. 80000 + Rp. 2500
Total biaya pengiriman: Rp. 82500
PS C:\Semester3Go\Modul 2\2c 1> go run soal2cno1.go
Berat parcel (gram): 9250
Detail berat: 9 kg + 250 gr
Detail biaya: Rp. 90000 + Rp. 3750
Total biaya pengiriman: Rp. 93750
PS C:\Semester3Go\Modul 2\2c 1> go run soal2cno1.go
Berat parcel (gram): 11750
Detail berat: 11 kg + 750 gr
Detail biaya: Rp. 110000 + Rp. 3750
Total biaya pengiriman: Rp. 110000
PS C:\Semester3Go\Modul 2\2c 1> 
```

*Tidak berjudul - Not...

File Edit Lihat

NIM : 2311102046
Nama : Faisal Khoiruddin
Kelas : IF-11-02

Ln 3, Col 17 100% Windows (CRLF) UTF-8

Deskripsi :

Program tersebut merupakan program BiayaPos untuk menghitung biaya pengiriman tersebut. 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.

- package main ➔ paket utama program golang
- import "fmt" ➔ mengimpor "fmt"
- func main() { ➔ merupakan fungsi utama
- var beratParcel int ➔ deklarasi variabel beratParcel bertipe data int
- fmt.Print("Berat parcel (gram): ") ➔ menampilkan statement untuk memasukkan Berat parcel (gram):
- fmt.Scan(&beratParcel) ➔ membaca input berat parcel dari user dan menyimpan di variabel beratParcel
- berat_kg := beratParcel / 1000 ➔ menghitung berat pengiriman per kilogram
- sisa_gram := beratParcel % 1000 ➔ menghitung berat pengiriman sisa gram

- `biayaPerKg := 10000` → biaya pengiriman perkilogram = 10000
- `var tambahanBiaya int` → deklarasi variabel `tambahanBiaya` bertipe data integer
- `if sisa_gram >= 500 {` → percabangan if jika `sisa_gram >= 500`
- `tambahanBiaya = sisa_gram * 5` → `tambahanBiaya = sisa_gram * 5`
- `}` else { → kalau tidak
- `tambahanBiaya = sisa_gram * 15` → `tambahanBiaya = sisa_gram * 15`
- `totalBiaya := (berat_kg * biayaPerKg) + tambahanBiaya` → → menghitung total biaya
- `if berat_kg > 10 {` → jika `berat_kg > 10`
- `totalBiaya -= tambahanBiaya` → kurangi tambahan biaya
- `detailBerat := fmt.Sprintf("%d kg + %d gr", berat_kg, sisa_gram)` → perhitungan detail berat
- `detailBiaya := fmt.Sprintf("Rp. %d + Rp. %d", berat_kg*biayaPerKg, tambahanBiaya)` → perhitungan detail biaya
- `fmt.Printf("Detail berat: %s\n", detailBerat)` → menampilkan Detail berat
- `fmt.Printf("Detail biaya: %s\n", detailBiaya)` → menampilkan Detail biaya
- `fmt.Printf("Total biaya pengiriman: Rp. %d\n", totalBiaya)` → menampilkan Total biaya pengiriman

2. 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?

Jika nam diberikan adalah 80.1, keluaran dari program tersebut yaitu "A", Namun, eksekusi program ini tidak sesuai dengan spesifikasi soal.

- b. Apa saja kesalahan dari program tersebut? Mengapa demikian? Jelaskan alur program seharusnya!

Kesalahan dari program tersebut yaitu variabel `nam` bertipe data `float64` yaitu bilangan decimal ditempatkan pada nilai bertipe data `string`. Karena penempatan variabel dan nilai dari variabel harus bertipe data sama, kalau berbeda, maka program error. Pada soal seharusnya variabel `nam` ditempatkan pada `if nam > 80` sampai `if` kondisi terakhir yaitu `40` dan variabel `nmk` ditempatkan pada `nmk = "A"` hingga `"E"`.

Alur program yang benar adalah memeriksa input pengguna untuk setiap rentang dimulai dari nilai yang tertinggi sampai kondisi terpenuhi. Jika kondisi sudah terpenuhi program tidak perlu memeriksa kondisi lainnya.

- c. Perbaiki program tersebut! Ujilah dengan masukan: 93.5; 70.6; dan 49.5. Seharusnya keluaran yang diperoleh adalah 'A', 'B', dan 'D'.

Source Code

```
package main

import "fmt"

func main() {
    var nam float64
    var nmk string
    fmt.Print("Nilai akhir mata kuliah: ")
    fmt.Scanln(&nam)
```

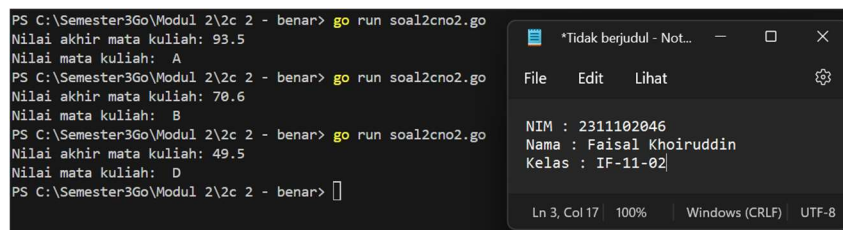
```

        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)
    }

```

Screenshot Output



```

PS C:\Semester3Go\Modul 2\2c 2 - benar> go run soal2cno2.go
Nilai akhir mata kuliah: 93.5
Nilai mata kuliah: A
PS C:\Semester3Go\Modul 2\2c 2 - benar> go run soal2cno2.go
Nilai akhir mata kuliah: 70.6
Nilai mata kuliah: B
PS C:\Semester3Go\Modul 2\2c 2 - benar> go run soal2cno2.go
Nilai akhir mata kuliah: 49.5
Nilai mata kuliah: D
PS C:\Semester3Go\Modul 2\2c 2 - benar>

```

*Tidak berjudul - Not...

File Edit Lihat

NIM : 2311102046
Nama : Faisal Khoiruddin
Kelas : IF-11-02

Ln 3, Col 17 | 100% | Windows (CRLF) UTF-8

Deskripsi :

Program tersebut merupakan program nilai akhir mata kuliah (NAM) [0..100] dan standar penilaian nilai mata kuliah (NMK). Pada program tersebut kita menginputkan nilai tertentu maka keluar output berupa standar penilaian nilai mata kuliah (NMK)

- `package main` → paket utama program golang
- `import "fmt"` → mengimpor "fmt"
- `func main() {` → merupakan fungsi utama
- `var nam float64` → deklarasi varuiabel nam bertipe data float64
- `var nmk string` → deklarasi varuiabel nmk bertipe data string
- `fmt.Print("Nilai akhir mata kuliah: ")` → menampilkan
- `fmt.Scanln(&nam)` Nilai akhir mata kuliah
- `if nam > 80 {` → percabangan if jika `nam > 80`
- `nmk = "A"` → nilainya A
- `} else if nam > 72.5 {` → percabangan if jika `nam > 72.5`
- `nmk = "AB"` → nilainya AB
- `} else if nam > 65 {` → percabangan if jika `nam > 65`
- `nmk = "B"` → nilainya B
- `} else if nam > 57.5 {` → percabangan if jika `nam > 57.5`
- `nmk = "BC"` → nilainya BC
- `} else if nam > 50 {` → percabangan if jika `nam > 50`
- `nmk = "C"` → nilainya C
- `} else if nam > 40 {` → percabangan if jika `nam > 50`
- `nmk = "D"` → nilainya D
- `} else {` → kalua tidak
- `nmk = "E"` → nilainya E
- `fmt.Println("Nilai mata kuliah: ", nmk)` → menampilkan Nilai mata kuliah

3. Diberikan sebuah persamaan sebagai berikut ini.

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
---	-----------------------------------

Source Code

```
package main

import (
    "fmt"
    "strings"
)

func main() {
    var factorsnum int
    fmt.Print("Bilangan: ")
    fmt.Scanln(&factorsnum)

    var factors []string
    for i := 1; i <= factorsnum; i++ {
        if factorsnum%i == 0 {
            factors = append(factors,
fmt.Sprintf("%d", i))
        }
    }
}
```

```

    }

    fmt.Println("Faktor:",
strings.Join(factors, " "))
}

```

Screenshot Output

```

PS C:\Semester3Go\Modul 2\2c 3 #1> go run soal2cno3output1.go
Bilangan: 12
Faktor: 1 2 3 4 6 12
PS C:\Semester3Go\Modul 2\2c 3 #1> go run soal2cno3output1.go
Bilangan: 7
Faktor: 1 7
PS C:\Semester3Go\Modul 2\2c 3 #1> 

```

*Tidak berjudul - Not...
File Edit Lihat
NIM : 2311102046
Nama : Faisal Khoiruddin
Kelas : IF-11-02
Ln 3, Col 17 | 100% | Windows (CRLF) | UTF-8

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

Source Code

```

package main

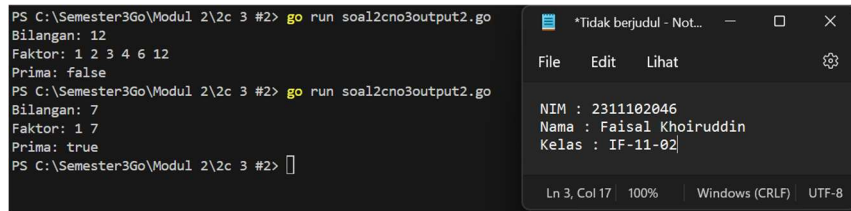
import (
    "fmt"
    "strings"
)

```



```
func main() {  
    var factorsnum int  
    fmt.Print("Bilangan: ")  
    fmt.Scanln(&factorsnum)  
  
    var factors []string  
    for i := 1; i <= factorsnum; i++ {  
        if factorsnum%i == 0 {  
            factors = append(factors,  
fmt.Sprintf("%d", i))  
        }  
    }  
  
    fmt.Println("Faktor:",  
strings.Join(factors, " "))  
  
    isPrime := true  
    if factorsnum <= 1 {  
        isPrime = false  
    } else {  
        for i := 2; i*i <= factorsnum; i++ {  
            if factorsnum%i == 0 {  
                isPrime = false  
                break  
            }  
        }  
    }  
  
    fmt.Println("Prima:", isPrime)  
}
```

Screenshot Output



```
PS C:\Semester3Go\Modul 2\2c 3 #2> go run soal2cno3output2.go
Bilangan: 12
Faktor: 1 2 3 4 6 12
Prima: false
PS C:\Semester3Go\Modul 2\2c 3 #2> go run soal2cno3output2.go
Bilangan: 7
Faktor: 1 7
Prima: true
PS C:\Semester3Go\Modul 2\2c 3 #2> []
```

The screenshot shows a Windows terminal window with a Go program being executed. The program takes an input number and outputs its factors and whether it is prime. For input 12, the factors are 1, 2, 3, 4, 6, and 12, and it is not prime. For input 7, the factors are 1 and 7, and it is prime. A Notepad window in the background contains student information: NIM: 2311102046, Nama: Faisal Khoiruddin, and Kelas: IF-11-02.

Deskripsi :

Program tersebut merupakan program menentukan factor dari bilangan yang diinputkan. Kemudian program tersebut dimodifikasi sehingga menampilkan operasi Boolean dengan output berupa true/false untuk menentukan bilangan prima.

- `package main` → paket utama program golang
- `import` → mengimpor
- `"fmt"` → mengimpor "fmt"
- `"strings"` → mengimpor "strings"
- `func main() {` → merupakan fungsi utama
- `var factorsnum int` → deklarasi factorsnum bertipe data integer
- `fmt.Print("Bilangan: ")` → menampilkan statement Bilangan: untuk memasukkan bilangan
- `fmt.Scanln(&factorsnum)` → membaca input pengguna dan menyimpan ke variabel actorsnum
- `var factors []string` → deklarasi variabel factors bertipe data string
- `for i := 1; i <= factorsnum; i++ {` → perulangan for `i := 1; i <= factorsnum; i++` (increment)
- `if factorsnum%i == 0 {` → percabangan jika `factorsnum%i == 0`
- `factors = append(factors, fmt.Sprintf("%d", i))` → `factors =`

`append(factors, fmt.Sprintf("%d", i))`

- `fmt.Println("Faktor:", strings.Join(factors, " "))` ➔ menampilkan faktor
- `isPrime := true` ➔ `isPrime` menjadi `true`
- `if factorsnum <= 1 {` ➔ jika `factorsnum <= 1`
- `isPrime = false` ➔ `isPrime` menjadi `false`
- `} else {` ➔ kalau tidak
- `for i := 2; i*i <= factorsnum; i++ {` ➔ melakukan perulangan `i := 2;`
`i*i <= factorsnum; i++`
- `if factorsnum%i == 0 {` ➔ jika `factorsnum%i == 0`
- `isPrime = false` ➔ `isPrime` sama dengan `false`
- `break` ➔ berhenti
- `fmt.Println("Prima:", isPrime)` ➔ menampilkan apakah bilangan merupakan bilangan prima atau bukan