

**LAPORAN PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL II  
REVIEW STRUKTUR KONTROL**



**Disusun Oleh :**

**Deshan Rafif Alfarisi / 2311102326**

**S1-IF-11-06**

**Dosen Pengampu :**

**Abednego Dwi Septiadi S.Kom., M.Kom.**

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**

**FAKULTAS INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**

**2024**

## I. DASAR TEORI

### a. Struktur Program Go

Dalam kerangka program yang ditulis dalam bahasa pemrograman Go, program utama selalu

mempunyai dua komponen berikut:

- i. **package main** merupakan penanda bahwa file ini berisi program utama.
- ii. **func main()** berisi kode utama dari sebuah program Go.  
  
Komentar, bukan bagian dari kode program, dan dapat ditulis di mana saja di dalam program:
- iii. Satu baris teks yang diawali dengan garis miring ganda (//) s.d. akhir baris, atau.
- iv. 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     fmt.Println(greetings)
7     fmt.Scanln(&a, &b)
8     fmt.Printf("%v + %v = %v\n", a, b, a+b)
9
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
75
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**

Beberapa bahasa pemrograman dirancang untuk diimplementasikan sebagai interpreter dan lainnya sebagai kompilator. Interpreter akan membaca setiap baris instruksi dan kemudian langsung mengeksekusinya, dengan hanya sedikit pemeriksaan apakah penulisan keseluruhan program sudah benar atau belum. Kompilator 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 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:\Userstgo\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. Video tutorial memulai program Go:

<https://youtu.be/S5Pt2qO2QEG>

## 2) Latihan

1. Selidiki bahasa-bahasa pemrograman berikut, apakah termasuk diinterpretasi, dikompilasi, dikompilasi (ke instruksi perantara) kemudian diinterpretasi:

- Pascal
- C dan C++
- java
- Python

2. Unduh kompilator Go di Computer yang Anda gunakan. kemudian salin content program di atas ke dalam folder C:\Users\tuser\id\Gothello\thello.go, yaitu buat folder Go dalam direktori home Anda, kemudian buat subfolder hello dan taruh file hello.go di dalamnya. Hidupkan terminal (cmd.exe), dan panggil go build di dalam folder hello tersebut. Periksa apakah hello.exe muncul di folder tersebut? Jika ya, coba eksekusi program tersebut, juga melalui terminal cmd.exe tersebut. (Jangan di klik melalui browser folder).

### 2.1 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	:fnt 1nt8 1nt32 // 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> -7)
real	float32 float64	32bit : -3.4E+38 .. 3.4E+38 64bit : -q.7E+388 .. 1.7E+388
boolean (atau logikal)	bool	false dan true
karakter	byte //uint8 rune //int32	tabel ASCII/UTF-8 tabel UTF-16

- 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
<code>+</code>	string integer dan real	konkatenasi 2 string operasi penjumlahan
<code>-*/</code>	integer dan real	operasi pengurangan, perkalian, dan pembagian
<code>%</code>	integer	operasi sisa pembagian integer (modulo)
<code>&amp;   ^ &amp;^</code>	integer	operasi <b>per-bit AND, OR, XOR, AND-NOT</b>
	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 apa saja	mendapatkan data dari lokasi memori dan mendapatkan lokasi dari variabel

Contoh:

Operasi	Hasil
"non suff:t" + "c:tt 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 dari data yang diberikan ke tipe yang diinginkan. Memanfaatkan fungsi `Sprintf` dan `Scan` dari paket `fmt`.

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

Lihat lampiran untuk contoh penggunaan.

Contoh:

Operasi	Hasil
2020.0° 19	will be an illegal expression error
int(2020.0) °a 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,"%äc",&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

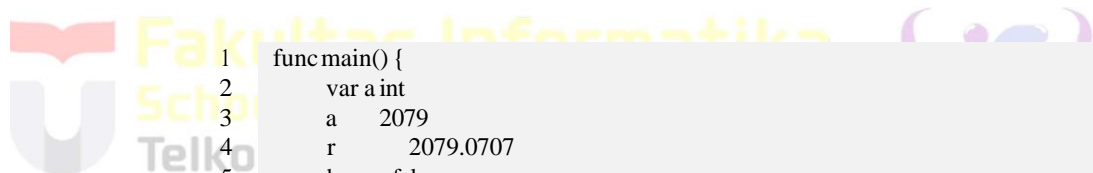
" (string kosong, string dengan panjang 0) untuk string

mi 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 = 2079
4     r = 2079.0707
5     b = false
6     c = x
7     s := "a string is a string"
8 }

```

### 1) 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, mengisif data ke lokasi memori (variabel)
input(v1, v2)	fmt.Scan( &v1, &v2 ) fmt.Scanln( &v1, &v2 ) fmt.Sprintf( "%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( "av %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 }
```

### 3) Konstanta Simbolik

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

```
1 const PI = 3.1415926535897932384626433
2 const PIARKER = "AKHIR"
```

## 2.2 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 inialisasi; 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 diisi 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 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). Inijuga 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
4	endwhile	}

Contoh penggunaan bentuk while-loop untuk menghitung y — adalah sebagai berikut:

	Notasi algoritma	Penulisan dalam bahasa Go
7	e <- 0.0000001	e = 0.0000001
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 {
		I

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)

#### 1) 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
7	repeat	for selesai:=false; !selesai; { kode yang diulang
2	kode yang diulang	selesai: kondisi
3	until (kondisi)	}
4		
5		for selesai:=false; !selesai; selesai=kondisi {
6		. kode yang diulang
7		}
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 berikutnya:", f1)	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 := ;
4  f2 := 7
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 }

```

## 2.3 StrukturKontrolPercabangan

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-7) then	if kondisi 1 {
70	.. kode untuk kondisi-7 true else if(kondisi-	. kode untuk kondisi 7 true
11	2) then	} else if kondisi_2 {
12	.. kode untuk kondisi-2 true dst. dst.	. kode untuk kondisi_2 true
13	else	dst. dst.
14	.. kode jika semua kondisi di atas false	} else {
75	endif	kode jika semua kondisi di atas
76		false
77		

Contoh bonversi(nilai,tubes,hehadiran) menjadiindebsnilai.

```

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
11 fmt.Printf( "Nilai %v dengan kehadiran %v%% dan buat tubes:%v, mendapat
    indeks %c\n", nilai, pctHadir, adaTubes, indeks )

```

## 2) Bentuk Swtch-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
--	------------------	---------------------------

7	depend on expresi nilai:	switch ekspresi { case
2	.. kode jika ekspresi bernilai 7	nilai_1:
3	nilai_2:	. kode jika ekspresi bernilai_7
4	.. kode jika ekspresi bernilai_2	case nilai 2:
5	dst. dst.	. kode jika ekspresi bernilai 2 dst. dst.
6	}	default:
7		kode jika tidak ada nilai yang cocok
8		dengan ekspresi
9		
70		
11	depend on (daftar variabel)	switch {
12	kondisi_1 :	case kondisi_1 :
13	.. kode jika ekspresi_1 true	. kode jika ekspresi: f_1 true
14	kondisi_2 :	case kondisi_2 :
15	.. kode jika ekspresi_2 true	. kode jika ekspresi: f_2 true
16	dst . dst .	dst . dst .
17	}	default:
18		jika tidak ada ekspresi yang bernilai
19		true
20		}

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

### 1. Soal Studi Case

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?

#### Sourcecode

```
package main

import "fmt"

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

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

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

    temp = satu
    satu = dua
    dua = tiga
    tiga = temp

    fmt.Println("Output akhir:")
    fmt.Println("satu =", satu)
    fmt.Println("dua =", dua)
    fmt.Println("tiga =", tiga)
}
```



## Screenshoot Output

```
go run /tmp/Q2iPhwLVtF.go
Masukan input string pertama: 3
Masukan input string kedua: 4
Masukan input string ketiga: 5
Output awal:
satu = 3
dua = 4
tiga = 5
Output akhir:
satu = 4
dua = 5
tiga = 3
```

## Deskripsi Program

Program ini bekerja dengan cara meminta pengguna memasukkan tiga data, kemudian menyimpannya dalam tiga variabel. Selanjutnya, program melakukan penukaran nilai antara variabel-variabel tersebut menggunakan variabel sementara. Hasil akhir dari penukaran ini adalah urutan ketiga data tersebut menjadi terbalik.

## 2. Soal Studi Case

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).

## Sourcecode

```
package main

import "fmt"

func main() {
    var tahun int

    fmt.Print("Masukkan tahun: ")
    fmt.Scanln(&tahun)

    if (tahun%400 == 0) || (tahun%4 == 0 && tahun%100 != 0) {
        fmt.Println(tahun, "adalah tahun kabisat")
    } else {
        fmt.Println(tahun, "bukan tahun kabisat")
    }
}
```

## Screenshoot Output

```
go run /tmp/4VMu2ZKCz2.go
Masukkan tahun: 2004
2004 adalah tahun kabisat
|
```

## Deskripsi Program

Program ini menerima input berupa tahun (bilangan bulat) dari pengguna, kemudian melakukan perhitungan berdasarkan aturan tahun kabisat yang umum digunakan. Aturan ini memperhitungkan kelipatan 4, 100, dan 400 untuk menentukan apakah suatu tahun memiliki hari tambahan (29 Februari) atau tidak. Hasil akhir berupa pesan yang menginformasikan apakah tahun tersebut adalah tahun kabisat atau bukan.

### 3. Soal Studi Case

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

## Sourcecode

```
package main

import (
    "fmt"
    "math"
)

func main() {
    var jariJari float64

    fmt.Print("Masukkan jari-jari bola: ")
    fmt.Scanln(&jariJari)

    volume := (4.0 / 3.0) * math.Pi * math.Pow(jariJari, 3)
    luasPermukaan := 4 * math.Pi * math.Pow(jariJari, 2)

    fmt.Printf("Volume bola: %.2f\n", volume)
    fmt.Printf("Luas permukaan bola: %.2f\n", luasPermukaan)
}
```

## Screenshoot Output

```
go run /tmp/ZhdruyQST3.go
Masukkan jari-jari bola: 5
Volume bola: 523.60
Luas permukaan bola: 314.16
|
```

**Deskripsi Program**

Pengguna diminta untuk memasukkan nilai jari-jari bola. Setelah itu, program akan melakukan perhitungan matematis berdasarkan rumus volume dan luas permukaan bola, kemudian menampilkan hasil perhitungan tersebut. Program ini menggunakan konstanta pi ( $\pi$ ) yang lebih akurat dan format output yang rapi untuk memudahkan pembacaan hasil.

### III. UNGUIDED

#### 1. Soal Studi Case

Dibaca nilai temperatur dalam derajat Celsius. Nyatakan temperatur tersebut dalam Fahrenheit. Lanjutkan program di atas, sehingga temperatur dinyatakan juga dalam derajat Reamur dan Kelvin.

#### Sourcecode

```
package main

import "fmt"

func main() {
    var celsius float64

    fmt.Print("Masukkan suhu dalam derajat Celsius: ")
    fmt.Scanln(&celsius)

    // Konversi ke Fahrenheit
    fahrenheit := (celsius * 9 / 5) + 32

    // Konversi ke Reamur
    reamur := (4 / 5) * celsius

    // Konversi ke Kelvin
    kelvin := celsius + 273.15

    fmt.Printf("%.2f derajat Celsius setara dengan:\n", celsius)
    fmt.Printf("%.2f derajat Fahrenheit\n", fahrenheit)
    fmt.Printf("%.2f derajat Reamur\n", reamur)
    fmt.Printf("%.2f Kelvin\n", kelvin)
}
```

#### Screenshoot Output

```
~ go run /tmp/EJkDdJ7ISB.go
Masukkan suhu dalam derajat Celsius: 5
5.00 derajat Celsius setara dengan:
41.00 derajat Fahrenheit
0.00 derajat Reamur
278.15 Kelvin
```

#### Deskripsi Program

Pengguna diminta untuk memasukkan nilai suhu dalam derajat Celsius. Program kemudian akan menghitung dan menampilkan nilai suhu yang setara dalam Fahrenheit, reamur, dan kelvin.

## 2. Soal Studi Case

Tipe karakter sebenarnya hanya apa yang tampak dalam tampilan. Di dalamnya tersimpan dalam bentuk biner 8 bit (byte) atau 32 bit (rune) saja. Buat program ASCII yang akan membaca 5 buah data integer dan mencetaknya dalam format karakter. Kemudian membaca 3 buah data karakter dan mencetak 3 buah karakter setelah karakter tersebut (menurut tabel ASCII)

Masukan terdiri dari dua baris. Baris pertama berisi 5 buah data integer. Data integer mempunyai nilai antara 32 s.d. 127. Baris kedua berisi 3 buah karakter yang berdampingan satu dengan yang lain (tanpa dipisahkanspasi).

Keluaran juga terdiri dari dua baris. Baris pertama berisi 5 buah representasi, karakter dari data yang diberikan, yang berdampingan satu dengan lain, tanpa dipisahkan spasi. Baris kedua berisi 3 buah karakter juga tidak dipisahkan oleh spasi). Gunakan `fmt.Scanf("%c", &var)` untuk pembacaan satu karakter dan `fmt.Printf("%c", var)` untuk penulisan satu karakter.

### Sourcecode

```
package main

import "fmt"

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

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

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

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

```

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

```

### Screenshoot Output

```

go run /tmp/OYuvIOg8E5.go
Masukkan 5 buah data integer (32-127):
66
97
103
117
115
Masukkan 3 buah karakter:
S
N
Karakter dari data integer: Bagus
3 karakter setelahnya: V

```

### Deskripsi Program

Program ini menerima input berupa sejumlah angka integer yang merepresentasikan kode ASCII dan beberapa karakter. Angka-angka integer tersebut kemudian dikonversi menjadi karakter yang sesuai dan ditampilkan. Selain itu, program juga menerima input berupa beberapa karakter dan menampilkan karakter-karakter berikutnya berdasarkan urutan dalam tabel ASCII.

### 3. Soal Studi Case

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.

## Sourcecode

```
package main

import "fmt"

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

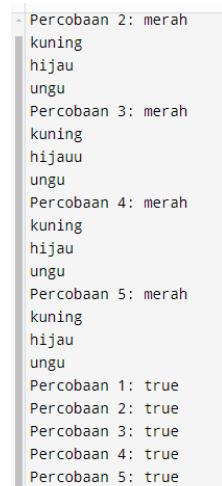
    fmt.Println("Masukkan 5 buah data integer (32-127):")
    for i := 0; i < 5; i++ {
        fmt.Scanf("%d", &numbers[i])
    }

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

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

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

## Screenshoot Output



```
Percobaan 2: merah
kuning
hijau
ungu
Percobaan 3: merah
kuning
hijauu
ungu
Percobaan 4: merah
kuning
hijau
ungu
Percobaan 5: merah
kuning
hijau
ungu
Percobaan 1: true
Percobaan 2: true
Percobaan 3: true
Percobaan 4: true
Percobaan 5: true
```

## Deskripsi Program

Program ini menerima input berupa warna cairan pada empat tabung reaksi untuk beberapa kali percobaan. Kemudian, program akan mengecek apakah urutan warna pada setiap percobaan sesuai dengan kriteria keberhasilan yang telah ditentukan, yaitu merah, kuning, hijau, dan ungu secara berurutan. Hasil pengecekan akan ditampilkan dalam bentuk boolean (true atau false), menunjukkan apakah percobaan tersebut berhasil atau gagal.

## 4. Soal Studi Case

Suatu pita(string) berisi kumpulan nama-namabunga yang dipisahkan oleh spasi dan '-', contoh pita: ilustrasi. Lanjutkan seperti berikut ini.

Rta: matar-meatl-tu1p-teratal-Ramboja-angrek.

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.

## Sourcecode

```
package main

import "fmt"

func main() {
    var N int
    var namaBunga string

    fmt.Print("Masukkan jumlah nama bunga: ")
    fmt.Scanln(&N)

    for i := 0; i < N; i++ {
        fmt.Printf("Masukkan nama bunga ke-%d: ", i+1)
        var bunga string
        fmt.Scanln(&bunga)
        namaBunga += bunga + " "
    }

    // Hapus spasi terakhir
    namaBunga = namaBunga[:len(namaBunga)-1]

    fmt.Println("Semua nama bunga:", namaBunga)
}
```



## Screenshoot Output

```
^ go run /tmp/rUTXrJkpNY.go
Masukkan jumlah nama bunga: 3
Masukkan nama bunga ke-1: kertas
Masukkan nama bunga ke-2: mawar
Masukkan nama bunga ke-3: tulip
Semua nama bunga: kertas mawar tulip
|
```

## Deskripsi Program

Pengguna diminta untuk memasukkan jumlah nama bunga yang ingin ditambahkan, kemudian secara berurutan memasukkan setiap nama bunga. Setiap nama bunga yang diinputkan akan ditambahkan ke akhir string yang sudah ada, dipisahkan oleh spasi. Dengan kata lain, program ini berfungsi seperti sebuah keranjang yang secara bertahap diisi dengan nama-nama bunga hingga terisi penuh sesuai dengan jumlah yang ditentukan oleh pengguna. Hasil akhirnya adalah sebuah string yang berisi semua nama bunga yang telah diinputkan.

### 5. Soal Studi Case

Setiap hari Pak Andi membawa banyakbarang belanjaan dari pasar denganmengendarai sepeda motor. Barang belanjaan tersebut dibawa dalam kantong terpal di kiri-kanan motor. Sepeda motor tidak akan olengjika selisih berat barang di kedua kantongsisitidak lebih dari 9 kg. Buatlah program Pak Andi yang menerima input dua buah bilangan real positif yang menyatakan berat total masing-masing isi kantongterpal. Program akan terus meminta input bilangantersebut hingga salah satu kantong terpal berisi 9 kg atau lebih.

## Sourcecode

```
package main

import "fmt"

func main() {
    var beratKiri, beratKanan float64

    for {
        fmt.Print("Masukkan berat isi kantong kiri (kg): ")
        fmt.Scanln(&beratKiri)

        fmt.Print("Masukkan berat isi kantong kanan (kg): ")
        fmt.Scanln(&beratKanan)

        if beratKiri >= 9 || beratKanan >= 9 {
            fmt.Println("Salah satu kantong sudah kelebihan beban!")
            break
        } else if beratKiri-beratKanan > 9 || beratKanan-beratKiri > 9 {
            fmt.Println("Selisih berat kedua kantong terlalu besar!")
        }
    }
}
```

```

    } else {
        fmt.Println("Beban seimbang. Lanjutkan belanja.")
    }
}
}

```

### Screenshoot Output

```

^ go run /tmp/cppBIPae73.go
Masukkan berat isi kantong kiri (kg): 1
Masukkan berat isi kantong kanan (kg): 5
Beban seimbang. Lanjutkan belanja.
Masukkan berat isi kantong kiri (kg): 2
Masukkan berat isi kantong kanan (kg): 7
Beban seimbang. Lanjutkan belanja.
Masukkan berat isi kantong kiri (kg): 6
Masukkan berat isi kantong kanan (kg): 9
Salah satu kantong sudah kelebihan beban!

```

### Deskripsi Program

Program ini akan terus meminta Pak Andi untuk memasukkan berat isi dari dua kantong belanjanya. Program akan memeriksa apakah selisih berat antara kedua kantong melebihi batas yang ditentukan (9 kg dalam kasus ini). Jika selisih berat terlalu besar atau salah satu kantong sudah kelebihan beban, program akan memberikan peringatan. Program ini akan terus berjalan hingga Pak Andi mencapai keseimbangan beban yang aman untuk sepeda motornya.

### 6. Soal Studi Case

Diberikan persamaan  $f(K) = (4k + 2)^2 / ((4k + 1) * (4k + 3))$ . Buatlah sebuah program yang menerima input sebuah bilangan sebagai K, kemudian menghitung dan menampilkan nilai  $f(K)$  sesuai persamaan di atas.

### Sourcecode

```

package main

import (
    "fmt"
    "math"
)

func main() {
    var K float64

    fmt.Print("Masukkan nilai K: ")
    fmt.Scanln(&K)

    // Hitung nilai f(K)
    fk := math.Pow((4*K+2), 2) / ((4*K+1) * (4*K+3))

    fmt.Printf("Nilai f(%g) adalah %g\n", K, fk)
}

```

```
}
```

## Screenshot Output

```
go run /tmp/ydaRtoAFaP.go
Masukkan nilai K: 100
Nilai f(100) adalah 1.0000061880039355
|
```

## Deskripsi Program

Pengguna diminta untuk memasukkan nilai input (K) yang kemudian akan digunakan dalam perhitungan fungsi tersebut. Fungsi yang digunakan dalam program ini adalah  $f(K) = (4k + 2)^2 / ((4k + 1) * (4k + 3))$ . Setelah nilai K dimasukkan, program akan melakukan perhitungan sesuai dengan rumus fungsi tersebut dan menampilkan hasil akhir berupa nilai f(K).