MODUL 2. REVIEW STRUKTUR KONTROL

2.1 Struktur Program Go

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

- **package main** merupakan penanda bahwa file ini berisi program utama.
- **func main()** berisi kode utama dari sebuah program Go.

Komentar, bukan bagian dari kode program, dan dapat ditulis di mana saja di dalam program:

- Satu baris teks yang diawali dengan garis miring ganda ('//') s.d. akhir baris, atau.
- Beberapa baris teks yang dimulai dengan pasangan karakter '/*' dan diakhiri dengan '*/'.

```
// Setiap program utama dimulai dengan "package main"
package main

// Impor paket yang dibutuhkan, "fmt" berisi proses I/O standar
import “fmt”

// Kode program utama dalam "fungsi main"
func main() {
    …
}
```

Telkom University informatics lab

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

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

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

```
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

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:\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)

Fakultas Informatika
Telkom University
Informatics lab

- 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/S5Rt2qO2QEg

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 komputer yang Anda gunakan. kemudian salin contoh program di atas ke dalam folder C:\Users\userid\Go\hello\hello.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).

Fakultas Informatika
School of Computing
Telkom University
informatics lab

2.2 Tipe Data dan Instruksi Dasar

1) Data dan Variabel

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

- Nama variabel dimulai dengan huruf dan dapat dikuti 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	

Fakultas Informatika
School of Computing
Telkom University
informatics lab

- 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

Fakultas Informatika
School of Computing
Telkom University

Contoh:

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

- Bahasa Go menganut kesesuaian tipe data yang ketat. Tipe data yang berbeda tidak boleh dicampur dalam satu ekspresi, bahkan tipe data masih yang sejenis, misalnya masih sama-sama integer (**int** dan **int32**). Untuk menyesuaikan tipe data, ada beberapa cara yang dapat dilakukan:
  - ✓ Casting, **tipe(data),** mengubah tipe dari data yang diberikan ke tipe yang diinginkan.
  - ✓ Memanfaatkan fungsi **Sprint** dan **Sscan** 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`

Fakultas Informatika
School of Computing
Telkom University

- 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  }
```

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 = 3.14159265358979323846264 33
2  const MARKER = “AKHIR”
```

4) **Soal Latihan Modul 2A**

Fakultas Informatika
School of Computing
Telkom University
informatics lab

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  }
```

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: **2016** Kabisat: true
2	Tahun: **2000** Kabisat: true
3	Tahun: **2018** Kabisat: false

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

Fakultas Informatika
School of Computing
Telkom University
Informatics lab

4. Dibaca nilai temperatur dalam derajat Celsius. Nyatakan temperatur tersebut dalam Fahrenheit

$$Celsius = (Fahrenheit - 32) \times \frac{5}{9} \; Reamur = Celcius \times \frac{4}{5} \; Kelvin = (Fahrenheit + 459.67) \times \frac{5}{9}$$

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

Temperatur Celsius: **50** Derajat Fahrenheit: 122

   Lanjutkan program di atas, sehingga temperatur dinyatakan juga dalam derajat Reamur dan Kelvin.

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

Temperatur Celsius: **50** Derajat Reamur: 40 Derajat Fahrenheit: 122 Derajat Kelvin: 323

5. 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 buat 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 dipisahkan spasi).

   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).

No.	Masukan	Keluaran
1	66 97 103 117 115 SNO	Bagus TOP

Fakultas Informatika
Telkom University
informatics lab

Catatan: Gunakan fmt.Scanf("%c", &var) untuk pembacaan satu karakter dan fmt.Printf("%c", var) untuk penulisan satu karakter.

2.3 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

```
for inisialisasi; kondisi; update {
    // .. for-loop ala C
    // .. ke-3 bagian opsional, tetapi ";" tetap harus ada
}
for kondisi {
    // .. ulangi kode di sini selama kondisi terpenuhi
    // .. sama seperti “for ; kondisi; {“
}
for {
    // .. tanpa kondisi, berarti loop tanpa henti (perlu if-break)
}
for ndx, var := range slice/array {
    // .. iterator mengunjungi seluruh isi slice/array
    // .. pada setiap iterasi ndx diset indeks dan var diisi nilainya
}
```

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, ada kondisi 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 <- 0.0000001	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 {
6	y <- y1	y = y1

7
8
9
	`    y1 <- 0.5*y + 0.5*(x/y)`
	`endwhile`
	`output("sqrt(",x,")=",y)`
		`    y1 = 0.5*y + 0.5*(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
2
3
4
	`repeat`
	`    .. kode yang diulang`
	`until (kondisi)`
		`for selesai:=false; !selesai; {`
		`    .. kode yang diulang`
		`    selesai = kondisi`
		`}`

5
6
7
8
9
		`for selesai:=false; !selesai;`
		`selesai=kondisi {`
		`    .. kode yang diulang`
		`}`

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

	Notasi Algoritma	Penulisan dalam bahasa Go

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

```
maxF := 100
f0 := 0
f1 := 1
f2 := 1
fmt.Println("Bilangan pertama:", f1 )
for f2 <= maxF {
    f0 = f1
    f1 = f2
    f2 = f1 + f0
    fmt.Println("Bilangan berikutnya:", f1 )
}
```

3) **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
```

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

Fakultas Informatika School of Computing Telkom University informatics lab

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: **3** Bunga 1: **Kertas** Bunga 2: **Mawar** Bunga 3: **Tulip** Pita: Kertas – Mawar – Tulip -	N : **0** Pita :

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):

Fakultas Informatika
School of Computing

Bunga 1: **Kertas** Bunga 2: **Mawar** Bunga 3: **Tulip** Bunga 4: **SELESAI** Pita: Kertas – Mawar – Tulip - Bunga: 3	Bunga 1: **SELESAI** Pita : Bunga: 0

3. Setiap hari Pak Andi membawa banyak barang belanjaan dari pasar dengan mengendarai sepeda motor. Barang belanjaan tersebut dibawa dalam kantong terpal di kiri-kanan motor. Sepeda motor tidak akan oleng jika selisih berat barang di kedua kantong sisi tidak lebih dari 9 kg.

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

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

```
Masukan berat belanjaan di kedua kantong: 5.5 1.0
Masukan berat belanjaan di kedua kantong: 7.1 8.5
Masukan berat belanjaan di kedua kantong: 2 6
Masukan berat belanjaan di kedua kantong: 9 5.8
Proses selesai.
```

Pada modifikasi program tersebut, program akan menampilkan **true** jika selisih kedua isi kantong lebih dari atau sama dengan 9 kg. Program berhenti memproses apabila total berat isi kedua kantong melebihi 150 kg atau salah satu kantong beratnya negatif.

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

```
Masukan berat belanjaan di kedua kantong: 5 10
Sepeda motor pak Andi akan oleng: false
Masukan berat belanjaan di kedua kantong: 55.6 70.2
Sepeda motor pak Andi akan oleng: true
Masukan berat belanjaan di kedua kantong: 72.3 66.9
Sepeda motor pak Andi akan oleng: false
Masukan berat belanjaan di kedua kantong: 59.5 98.7
Proses selesai.
```

Telkom University
Informatics lab

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
```

$\sqrt{2}$ 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 =` **`10`** `Nilai akar 2 = 1.4062058441`
2	`Nilai K =` **`100`** `Nilai akar 2 = 1.4133387072`
3	`Nilai K =` **`1000`** `Nilai akar 2 = 1.4141252651`

Fakultas Informatika
School of Computing
Telkom University
informatics lab

2.4 Struktur Kontrol Percabangan

Untuk analisa kasus, bahasa Go mendukung dua bentuk percabangan, yaitu **if-else** dan **switch-case**.

1) **Bentuk If-Else**

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

	Notasi algoritma	Penulisan dalam bahasa Go

1
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 )
```

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**.

Fakultas Informatika
School of Computing
Telkom University
Informatics lab

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

Fakultas Informatika
School of Computing

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 }
```

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

3) **Soal Latihan Modul 2C**

1. PT POS membutuhkan aplikasi perhitungan biaya kirim berdasarkan berat parsel. Maka, **buatlah program BiayaPos untuk menghitung biaya pengiriman tersebut dengan ketentuan sebagai berikut!**

   Dari berat parsel (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 parsel (gram): **8500** Detail berat: 8 kg + 500 gr Detail biaya: Rp. 80000 + Rp. 2500 Total biaya: Rp. 82500
2	Contoh #2 Berat parsel (gram): **9250** Detail berat: 9 kg + 250 gr Detail biaya: Rp. 90000 + Rp. 3750 Total biaya: Rp. 93750
3	Contoh #3 Berat parsel (gram): **11750** Detail berat: 11 kg + 750 gr Detail biaya: Rp. 110000 + Rp. 3750 Total biaya: Rp. 110000

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:

a. Jika **nam** diberikan adalah 80.1, apa keluaran dari program tersebut? Apakah eksekusi program tersebut sesuai spesifikasi soal?

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

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

3. 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: **12** Faktor: 1 2 3 4 6 12	Bilangan: **7** Faktor: 1 7

   Bilangan bulat **b** > 0 merupakan bilangan prima **p** jika dan hanya jika memiliki persis dua faktor bilangan saja, yaitu 1 dan dirinya sendiri.

   Lanjutkan program sebelumnya. Setelah menerima masukan sebuah bilangan bulat **b** > 0. Program tersebut mencari dan menampilkan semua faktor bilangan tersebut. Kemudian, program menentukan apakah **b** merupakan bilangan prima.

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

Bilangan: **12** Faktor: 1 2 3 4 6 12 Prima: false	Bilangan: **7** Faktor: 1 7 Prima: true

Fakultas Informatika
School of Computing