MODUL PRAKTIKUM 12 - WHILE-LOOP

ALGORITMA DAN PEMROGRAMAN 1

S1 INFORMATIKA

Telkom University

Fakultas Informatika
School of Computing
Telkom University

informatics lab

LEMBAR PENGESAHAN

Saya yang bertanda tangan di bawah ini:

Nama	: Prasti Eko Yunanto, S.T., M.Kom.
NIP	: 19890017
Koordinator Mata Kuliah	: Algoritma dan Pemrograman 1
Prodi	: S1 Informatika

Menerangkan dengan sesungguhnya bahwa modul ini digunakan untuk pelaksanaan praktikum di Semester Ganjil Tahun Ajaran 2024/2025 di Laboratorium Informatika, Fakultas Informatika, Universitas Telkom.

Bandung, 17 Agustus 2024

Mengesahkan,	Mengetahui,
Koordinator Mata Kuliah	Kaprodi S1 Informatika
Algoritma Pemrograman 1	
Prasti Eko Yunanto, S.T., M.Kom.	Dr. Erwin Budi Setiawan, S.Si., M.T.
NIP. 19890017	NIP. 00760045

MODUL 12. WHILE-LOOP

12.1 Paradigma Perulangan

Perulangan merupakan salah satu struktur kontrol yang memungkinkan suatu instruksi yang sama dilakukan berulang kali dalam waktu atau jumlah yang lama. Tanpa instruksi perulangan, maka suatu instruksi akan ditulis dalam jumlah yang sangat banyak. Sebelumnya pada modul ke-5 dan 6 telah dipelajari instruksi perulangan dengan for-loop. Instruksi for-loop memungkinkan kita melakukan berulangan sebanyak n iterasi, akan tetapi pada banyak kasus yang melibatkan perulangan, **tidak semua perulangan diketahui jumlah iterasinya di awal**. Perulangan seperti ini disebut juga dengan istilah **perulangan dengan kondisi**.

Contoh perulangan jenis ini di kehidupan dunia nyata adalah sebagia berikut:

- *"Menulis teks tertentu selama tinta pena masih ada".*
  Terdapat kondisi "tinta pena masih ada" sebagai syarat perulangan.
- *"Saya makan suap demi suap selama saya masih lapar".*
  Terdapat kondisi "saya masih lapar" sebagai syarat perulangan.

Penting! Pastikan bahwa instruksi perulangan yang digunakan pasti bisa membuat proses perulangan berhenti, apabila tidak maka program akan terus berjalan mengulangi instruksi tanpa akan pernah berhenti.

Terdapat dua bentuk perulangan dengan kondisi ini, yaitu **while-loop** dan **repeat-until**. Masing-masing kita akan bahas di modul yang berbeda.

12.2 Karakterisik While-Loop

Struktur kontrol perulangan menggunakan while-loop memiliki bentuk yang hampir serupa dengan penulisan if-then pada percabangan, yaitu memiliki **kondisi** dan **aksi**. Hal yang membedakan adalah aksi akan dilakukan secara berulang-ulang selama kondisi bernilai true.

1) **Kondisi**, merupakan nilai atau operasi tipe data yang menghasilkan tipe data boolean. Kondisi ini merupakan syarat terjadinya perulangan. Artinya perulangan terjadi apabila kondisi bernilai true.
2) **Aksi**, merupakan kumpulan instruksi yang akan dieksekusi secara berulang-ulang selama kondisi bernilai true. *Salah satu instruksi dari aksi harus bisa membuat kondisi yang awalnya bernilai true manjadi false, tujuannya adalah untuk membuat perulangan berhenti.*

Pada penulisan notasinya secara umum bahasa pemrograman menggunakan kata kunci **while**, tetapi khusus di bahasa pemrograman Go, kata kunci yang digunakan adalah **for**. Walaupun berbeda dari kata kunci yang digunakan, **secara struktur penulisannya tetap sama**, sehingga tetap mudah untuk membedakan instruksi **for** yang digunakan adalah **for-loop** atau **while-loop**.

Berikut adalah notasi dari while-loop:

Notasi dalam pseudocode	Notasi dalam bahasa Go

while kondisi **do**
    // aksi
endwhile
	for kondisi {
	    // aksi
	}

12.3 Implementasi menggunakan Go

Contoh persoalan yang melibatkan perulangan dengan kondisi ini adalah:

Program yang akan menampilkan hasil penjumlahan dua bilangan yang diterima dari masukan pengguna secara terus menerus. Proses ini akan berlangsung terus selama hasil penjumlahan adalah genap.

```
1   // filename: whileloop1.go
2   package main
3   import "fmt"
4
5   func main() {
6       var a int
7       var b int
8       fmt.Scan(&a, &b)
9       for (a + b) % 2 == 0 {
10          fmt.Println("Hasil penjumlahan",a+b)
11          fmt.Scan(&a, &b)
12      }
13      fmt.Println("Program selesai")
14  }
C:\users\go\src\hello>go build whileloop1.go
C:\users\go\src\hello>whileloop1
10 6
Hasil penjumlahan 16
5 5
Hasil penjumlahan 10
3 9
Hasil penjumlahan 12
9 2
Program selesai
C:\users\go\src\hello>whileloop1
4 5
Program selesai
```

Pada contoh di atas, instruksi di baris ke-10 dan 11 akan berhenti dieksekusi berulang ketika kondisi di baris ke-9 bernilai false, yaitu ketika a dan b bernilai 9 dan 2 (9 + 2 = 11, yaitu ganjil).

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	`    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 )`

12.4 Contoh Soal Modul 12

1) Buatlah program yang digunakan untuk menampilkan deret bilangan Faktorial dari sautu bilangan.

   Masukan terdiri dari sebuah bilangan bulat non negatif n.

   Keluaran berupa deret bilangan dari Faktorial n. Perhatikan contoh masukan dan keluaran yang diberikan.

   Contoh masukan dan keluaran:

No	Masukan	Keluaran
1	0	1
2	5	5 x 4 x 3 x 2 x 1
3	10	10 x 9 x 8 x 7 x 6 x 5 x 4 x 3 x 2 x 1
4	1	1

   Jawaban

```go
package main
import "fmt"
func main() {
    var n, j int
    fmt.Scan(&n)
    j = n
    for j > 1 {
        fmt.Print(j, " x ")
        j = j - 1
    }
    fmt.Println(1)
```

```
12 }
```

```
gppras@SR8 GO % go build Demo_Soal.go
gppras@SR8 GO % ./Demo_Soal
0
1
gppras@SR8 GO % ./Demo_Soal
5
5 x 4 x 3 x 2 x 1
gppras@SR8 GO % ./Demo_Soal
10
10 x 9 x 8 x 7 x 6 x 5 x 4 x 3 x 2 x 1
gppras@SR8 GO % ./Demo_Soal
1
1
```

2) Buatlah program Go yang digunakan untuk login ke dalam suatu aplikasi. Asumsi token untuk yang valid adalah "12345abcde".

Masukan terdiri dari suatu token. Selama token yang diberikan salah, maka program akan meminta token secara terus menerus hingga token yang diberikan benar.

Keluaran adalah teks yang menyatakan "Selamat Anda berhasil login".

Contoh masukan dan keluaran:

No	Masukan	Keluaran
1	Qwe12312 231234 13213 123lijwe 12345abcde	Selamat Anda berhasil login
2	12345abcde	Selamat Anda berhasil login

Jawaban

```go
package main
import "fmt"
func main() {
    var token string
    fmt.Scan(&token)
    for token != "12345abcde" {
        fmt.Scan(&token)
    }
    fmt.Println("Selamat Anda berhasil login")
}
```

```
gppras@SR8 GO % go build Demo_Soal.go
gppras@SR8 GO % ./Demo_Soal
Qwe12312
231234
13213
123lijwe
12345abcde
Selamat Anda berhasil login
gppras@SR8 GO % ./Demo_Soal
12345abcde
Selamat Anda berhasil login
```

3) Buatlah program dalam bahasa Go yang digunakan untuk mencetak N bilangan pertama dalam deret Fibonacci.

Masukan terdiri dari bilangan bulat positif N dengan nilai besar atau sama dengan 2.

Keluaran terdiri dari sejumlah N bilangan yang menyatakan N deret bilangan Fibonacci yang pertama.

Contoh masukan dan keluaran:

No	Masukan	Keluaran
1	5	0 1 1 2 3
2	2	0 1
3	10	0 1 1 2 3 5 8 13 21 34

Jawaban

```
1  package main
2  import "fmt"
3  func main() {
4      var N, s1, s2, j, temp int
5      fmt.Scan(&N)
6      s1 = 0
7      s2 = 1
8      j = 0
9      for j < N {
10         fmt.Print(s1," ")
11         temp = s1 + s2
12         s1 = s2
13         s2 = temp
14         j = j + 1
15     }
16 }
```

```
gppras@SR8 GO % go build Demo_Soal.go
gppras@SR8 GO % ./Demo_Soal
5
0 1 1 2 3
gppras@SR8 GO % ./Demo_Soal
2
0 1
gppras@SR8 GO % ./Demo_Soal
10
0 1 1 2 3 5 8 13 21 34
```