MODUL PRAKTIKUM 13 - REPEAT-UNTIL

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 13. REPEAT-UNTIL

13.1 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. Pada modul 12 sebelumnya telah dipelajari terkait penggunaan struktur kontrol perulangan dengan while-loop, selanjutnya perulangan juga dapat dilakukan menggunakan **repeat-until**.

Penggunaan repeat-until pada dasarnya sama dengan while-loop di mana perulangan berdasarkan kondisi. Perbedaan terletak pada kondisi yang digunakan, pada while-loop kondisi yang harus didefinisikan adalah kondisi perulangannya, atau kapan perulangan itu terjadi, sedangkan pada repeat-until kondisi yang harus didefinisikan merupakan kondisi berhenti, atau kapan perulangan tersebut harus dihentikan.

Kondisi perulangan dan kondisi berhenti memiliki keterhubungan sifat komplemen, sehingga apabila kita mengetahui kondisi perulangannya, maka cukup dengan menambahkan operator negasi atau not untuk mengubah menjadi kondisi berhenti. Hal ini berlaku juga sebaliknya, komplemen dari kondisi berhenti adalah kondisi perulangan.

Pahami beberapa contoh yang diberikan berikut ini:

- Statement while-loop: *"Menulis teks tertentu **selama** tinta pena masih ada"*.
  Statement repeat-until: *"Menulis teks tertentu **sampai** tinta pena habis"*.
  Komplemen dari kondisi "tinta pena masih ada" adalah "tinta pena habis".
- Statement while-loop: *"Saya makan suap demi suap **selama** saya masih lapar"*.
  Statement repeat-until: *"Saya makan suap demi suap **sampai** saya merasa kenyang"*.
  Komplemen dari kondisi "saya masih lapar" adalah. "*saya merasa kenyang*".

13.2 Karakteristik Repeat-Until

Komponen dari repeat-until sama dengan while-loop, yaitu terdapat kondisi dan aksi, hanya struktur penulisannya saja yang berbeda.

1) **Aksi**, merupakan kumpulan instruksi yang akan dilakukan perulangan. Aksi minimal dijalankan sekali, baru dilakukan pengecekan kondisi berhenti setelahnya. Apabila kondisi bernilai true, maka perulangan dihentikan.
2) **Kondisi/berhenti**, merupakan kondisi berhenti dari perulangan, harus bernilai false selama perulangan dilakukan.

Notasi repeat-until memiliki banyak sekali keragaman kata kunci di dalam bahasa pemrograman. Penggunaan repeat-until sebenarnya berasal dari keluarga bahasa pemrograman Pascal. Pada keluarga bahasa pemrograman C/C++ digunakan do-while, sedangkan pada bahasa Go tidak ada instruksi eksplisit untuk repeat-until.

Notasi dalam pseudocode	Notasi dalam bahasa Go

repeat
    // aksi
until kondisi
	// versi dengan kata kunci break
	for kondisi = false; !kondisi; {
	    // aksi
	    kondisi = // update nilai kondisi
	}

13.3 Implementasi menggunakan Go

Sebagai contoh, misalnya terdapat suatu program yang digunakan untuk mengecek username dan password yang digunakan pengguna ketika login adalah "admin" dan "admin12345".

```
1   // filename: repeatuntil1.go
2   package main
3   import "fmt"
4
5   func main() {
6       var usr, pwd int
7       var kondisi bool
8       for kondisi = false; !kondisi; {
9           fmt.Scan(&usr, &pwd)
10          konsidi = usr == "admin" && pwd == "admin12345"
11      }
12      fmt.Println("Selamat, Anda berhasil login ")
13  }
14
```

```
C:\users\go\src\hello>go build repeatuntil1.go
C:\users\go\src\hello>repeatuntil1
user admin
admin admin
admin123 admin123
admin admin12345
Selamat, Anda berhasil login
C:\users\go\src\hello>repeatuntil1
admin admin12345
Selamat, Anda berhasil login
```

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	**until** f2 > maxF	selesai = f2 > maxF
12		}

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

Kode program dengan bahasa 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  }
```

13.4 Contoh Soal Modul 13

1) Buatlah program menggunakan bahasa Go yang menerima input kata dan mencetaknya sebanyak jumlah pengulangan yang diinginkan oleh pengguna. Program akan dihentikan ketika jumlah kata yang dicetak mencapai jumlah yang diinginkan oleh pengguna.

   Masukan berupa suatu kata dan jumlah pengulangan yang diinginkan oleh pengguna.

Keluaran berupa kata yang diinputkan pengguna dan dicetak sebanyaknya jumlah pengulangan yang diinginkan oleh pengguna.

Contoh masukan dan keluaran:

No	Masukan	Keluaran
1	pagi 3	pagi pagi pagi
2	kursi 5	kursi kursi kursi kursi kursi

Jawaban:

```go
package main
import "fmt"
func main() {
    var word string
    var repetitions int
    fmt.Scan(&word, &repetitions)
    counter := 0
    for done := false; !done; {
        fmt.Println(word)
        counter++
        done = (counter >= repetitions)
    }
}
```

```
gppras@SR8 GO % go build Demo_Soal.go
gppras@SR8 GO % ./Demo_Soal
pagi 3
pagi
pagi
pagi
gppras@SR8 GO % ./Demo_Soal
kursi 7
kursi
kursi
kursi
kursi
kursi
kursi
kursi
```

2) Buatlah program dalam bahasa Go yang meminta pengguna untuk memasukkan bilangan bulat positif. Program akan terus meminta input hingga pengguna memasukkan bilangan bulat positif.

Masukan berupa bilangan bulat positif, apabila bukan maka program akan terus meminta masukan hingga bilangan yang diberikan adalah bilangan bulat positif.

Keluaran berupa satu baris keluaran yang menunjukkan n bilangan adalah bilangan bulat positif.

Contoh masukan dan keluaran:

No	Masukan	Keluaran
1	-5 -2 -1 0 5	5 adalah bilangan bulat positif
2	17	17 adalah bilangan bulat positif

Jawaban:

```go
package main
import "fmt"
func main() {
	var number int
	var continueLoop bool
	for continueLoop = true; continueLoop; {
	    fmt.Scan(&number)
	    continueLoop = number <= 0
	}
	fmt.Printf("%d adalah bilangan bulat positif\n", number)
}
```

```
gppras@SR8 GO % go build Demo_Soal.go
gppras@SR8 GO % ./Demo_Soal
17
17 adalah bilangan bulat positif
gppras@SR8 GO % ./Demo_Soal
-5
-2
-1
0
5
5 adalah bilangan bulat positif
```

3) Buatlah program yang digunakan untuk melakukan pengecekan apakah suatu bilangan merupakan kelipatan dari bilangan lainnya.

Masukan terdiri dari dua buah bilangan bulat positif X dan Y.

Keluaran terdiri dari perulangan pengurangan kelipatan dengan hasil akhir boolean yang menyatakan apakah bilangan X merupakan kelipatan dari Y.

Contoh masukan dan keluaran:

No	Masukan	Keluaran
1	5 2	3 1 -1 false
2	15 3	12 9 6 3 0 true
3	25 5	20 15 10 5 0 true

Jawaban:

```
1  package main
2  import "fmt"
3  func main() {
4      var x int
5      var y int
6      var selesai bool
7      fmt.Scan(&x, &y)
8      for selesai = false; !selesai; {
9          x = x - y
10         fmt.Println(x)
11         selesai = x <= 0
12     }
13     fmt.Println(x == 0)
14 }
```

```
gppras@SR8 GO % go build Demo_Soal.go
gppras@SR8 GO % ./Demo_Soal
5 2
3
1
-1
false
gppras@SR8 GO % ./Demo_Soal
15 3
12
9
6
3
0
true
gppras@SR8 GO % ./Demo_Soal
25 5
20
15
10
5
0
true
```