

LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 1
MODUL 13
“REPEAT-UNTIL”



DISUSUN OLEH:
ANGGUN WAHYU WIDIYANA
103112480280
S1 IF-12-01
DOSEN:
Yohani Setiya Rafika Nur, M. Kom.

PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2024/2025

DASAR TEORI

Paradigma Percabangan

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*".

Karakteristik While-Loop

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

- a) **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.
-

- b) **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 psedeucode	Notasi dalam bahasa Go
repeat // aksi Until kondisi	// versi dengan kata kunci break for kondisi = false; !kondisi; { // aksi kondisi = // update nilai kondisi }

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        kondisi = usr == "admin" && pwd == "admin12345"
11    }
12    fmt.Println("Selamat, Anda berhasil login ")
13 }
```

```
C:\users\go\src\hello>go build repeatuntil1.go
C:\users\go\src\hello>repeatuntil
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	fmt. Println ("Bilangan berikutnya:"
10	output ("Bilangan	,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 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 }

```

CONTOH SOAL

Latihan1

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 sebanyak 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

Source Code:

```
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)
    }
}
```

Output:

```
PS C:\Users\User\Documents\MODUL 13> go run "c:\Users\User\Documents\MODUL 13\Contoh Soal\Cosol1.go"
pagi 3
pagi
pagi
PS C:\Users\User\Documents\MODUL 13> go run "c:\Users\User\Documents\MODUL 13\Contoh Soal\Cosol1.go"
kursi 5
kursi
kursi
kursi
kursi
PS C:\Users\User\Documents\MODUL 13> go run "c:\Users\User\Documents\MODUL 13\Contoh Soal\Cosol1.go"
```

Deskripsi Program:

Program Go ini bertujuan untuk menerima dua input dari pengguna: sebuah kata (string) dan jumlah pengulangan (integer) yang diinginkan. Setelah menerima input, program akan menggunakan sebuah loop untuk mencetak kata tersebut berulang kali hingga jumlah pengulangan yang ditentukan tercapai. Proses ini dilakukan dengan mendeklarasikan sebuah variabel penghitung yang akan bertambah setiap kali kata dicetak, dan loop akan berhenti ketika penghitung mencapai jumlah pengulangan yang diminta.

Latihan2

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

Source Code:

```
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)
}
```

Output:

```
PS C:\Users\User\Documents\MODUL 13> go run "c:\Users\User\Documents\MODUL 13\Contoh Soal\Coso12.go"
-5
-2
-1
0
5
5 adalah bilangan bulat positif
PS C:\Users\User\Documents\MODUL 13> go run "c:\Users\User\Documents\MODUL 13\Contoh Soal\Coso12.go"
17
17 adalah bilangan bulat positif
PS C:\Users\User\Documents\MODUL 13> go run "c:\Users\User\Documents\MODUL 13\Contoh Soal\Coso12.go"
```

Deskripsi Program:

Program Go ini untuk meminta pengguna memasukkan bilangan bulat positif. Program akan terus meminta input hingga pengguna memberikan bilangan bulat yang memenuhi syarat tersebut. Jika input yang diberikan bukan bilangan bulat positif, program akan menampilkan prompt untuk memasukkan angka kembali. Setelah pengguna akhirnya memasukkan bilangan bulat positif, program akan mencetak output yang menyatakan bahwa angka tersebut adalah bilangan bulat positif. Dengan demikian, program ini memastikan bahwa hanya bilangan bulat positif yang diterima dan dikonfirmasi kepada pengguna.

Latihan3

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

Source Code:

```
package main

import "fmt"

func main() {
    var x, y int
    var selesai bool
    fmt.Scan(&x, &y)
    for selesai = false; !selesai; {
        x = x-y
        fmt.Println(x)
        selesai = x <= 0
    }
    fmt.Println(x==0)
}
```

Output:

```
PS C:\Users\User\Documents\MODUL 13> go run "c:\Users\User\Documents\MODUL 13\Contoh Soal\Cosol3.go"
5
2
3
1
-1
false
PS C:\Users\User\Documents\MODUL 13> go run "c:\Users\User\Documents\MODUL 13\Contoh Soal\Cosol3.go"
15
3
12
9
6
3
0
true
PS C:\Users\User\Documents\MODUL 13> go run "c:\Users\User\Documents\MODUL 13\Contoh Soal\Cosol3.go"
25
5
20
15
10
5
0
true
PS C:\Users\User\Documents\MODUL 13> go run "c:\Users\User\Documents\MODUL 13\Contoh Soal\Cosol3.go"
```

Deskripsi Program:

Program Go ini berfungsi untuk memeriksa apakah suatu bilangan bulat positif X merupakan kelipatan dari bilangan bulat positif Y. Setelah menerima input dua bilangan tersebut, program akan melakukan pengurangan berulang antara X dan Y hingga X menjadi nol atau negatif. Selama proses pengurangan, program mencetak nilai X yang dihasilkan dari setiap langkah pengurangan. Setelah mencapai kondisi terminasi, program akan mengevaluasi apakah X sama dengan nol, yang menandakan bahwa X adalah kelipatan dari Y, dan mencetak hasil boolean yang menunjukkan status tersebut.

SOAL LATIHAN

Latihan1

Buatlah program yang digunakan untuk menghitung banyaknya digit dari suatu bilangan.

Masukan berupa bilangan bulat positif.

Keluaran berupa bilangan bulat yang menyatakan banyaknya digit dari bilangan yang diberikan pada masukan.

Contoh masukan dan keluaran

No	Masukan	Keluaran
1	5	1
2	234	3
3	78787	5
4	1894256	7

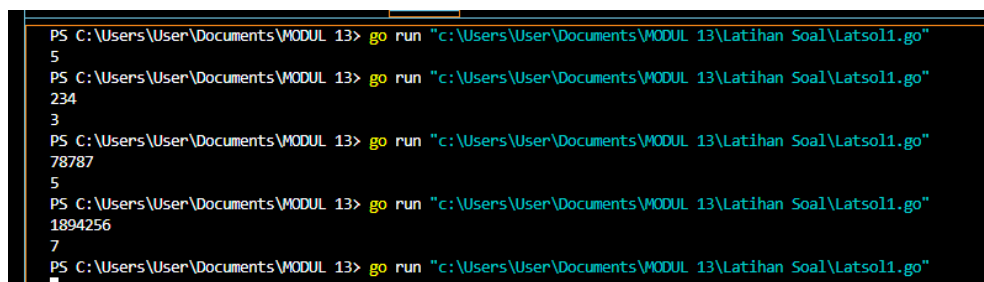
Source Code:

```
package main
import "fmt"
func main(){
    var bilangan int
    fmt.Scan(&bilangan)

    digit := 0
    if bilangan == 0 {
        digit = 1
    } else {
        for bilangan > 0 {
            bilangan /= 10
            digit++
        }
    }

    fmt.Println(digit)
}
```

Output:



```
PS C:\Users\User\Documents\MODUL 13> go run "c:\Users\User\Documents\MODUL 13\Latihan Soal\Latsol1.go"
5
PS C:\Users\User\Documents\MODUL 13> go run "c:\Users\User\Documents\MODUL 13\Latihan Soal\Latsol1.go"
234
3
PS C:\Users\User\Documents\MODUL 13> go run "c:\Users\User\Documents\MODUL 13\Latihan Soal\Latsol1.go"
78787
5
PS C:\Users\User\Documents\MODUL 13> go run "c:\Users\User\Documents\MODUL 13\Latihan Soal\Latsol1.go"
1894256
7
PS C:\Users\User\Documents\MODUL 13> go run "c:\Users\User\Documents\MODUL 13\Latihan Soal\Latsol1.go"
```

Deskripsi Program:

Program Go ini berfungsi untuk menghitung banyaknya digit dari suatu bilangan bulat positif. Setelah meminta pengguna untuk memasukkan bilangan, program akan menghitung jumlah digit dengan membagi bilangan tersebut secara berulang dengan 10 hingga nilainya menjadi nol, sambil menghitung berapa kali pembagian dilakukan. Hasil akhir, yang merupakan jumlah digit, akan dicetak ke layar.

Latihan2

Buatlah program yang digunakan untuk mendapatkan bilangan bulat optimal dari bilangan yang telah diinputkan. Melakukan penjumlahan tiap perulangan mencapai pembulatan keatas dari bilangan yang diinputkan.

Masukan berupa bilangan desimal.

Keluaran terdiri dari bilangan hasil penjumlahan tiap perulangannya sampai pembulatan keatas dari bilangan yang diinputkan.

Contoh masukan dan keluaran

No	Masukan	Keluaran
1	0.2	0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
2	2.7	2.8 2.9 3

Source Code:

```
package main
import (
    "fmt"
    "math"
)
func main(){
    var bilangan float64
    fmt.Scan(&bilangan)
    pembulatan := math.Ceil(bilangan)
    i := bilangan
    for i <= pembulatan {
        fmt.Printf("%.1f\n", i)
        i += 0.1
    }
}
```

Output:

```
PS C:\Users\User\Documents\MODUL 13> go run "c:\Users\User\Documents\MODUL 13\Latihan Soal\Latso12.go"
0.2
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
PS C:\Users\User\Documents\MODUL 13> go run "c:\Users\User\Documents\MODUL 13\Latihan Soal\Latso12.go"
2.7
2.7
2.8
2.9
PS C:\Users\User\Documents\MODUL 13> go run "c:\Users\User\Documents\MODUL 13\Latihan Soal\Latso12.go"
```

Deskripsi Program:

Program Go berfungsi untuk menghasilkan serangkaian bilangan desimal yang dimulai dari angka desimal yang diinputkan oleh pengguna hingga mencapai pembulatan ke atas dari angka tersebut. Setelah pengguna memasukkan bilangan desimal, program menghitung pembulatan ke atas menggunakan fungsi `math.Ceil` dan menyimpannya dalam variabel pembulatan. Program kemudian memulai dari nilai bilangan yang diinputkan dan menggunakan perulangan untuk menambahkan 0.1 pada setiap iterasi. Setiap nilai yang dihasilkan dicetak ke layar hingga nilai tersebut melebihi pembulatan yang telah dihitung. Dengan demikian, program ini memberikan keluaran yang menunjukkan semua bilangan desimal dari input hingga mencapai angka bulat terdekat yang lebih besar.

(Untuk inputan 2.7-nya seharusnya sampai 3.0, tetapi mohon maaf laptop saya kadang emang nge-lag, jadi keluarannya kurang benar sedikit)

Latihan3

Sebuah organisasi amal sedang mengumpulkan donasi untuk mendukung kegiatan sosial mereka. Setiap donatur dapat memberikan sumbangan dalam jumlah tertentu. Program ini akan terus meminta input dari pengguna untuk jumlah donasi hingga total donasi mencapai atau melebihi target yang telah ditentukan.

Masukan pada baris pertama berupa bilangan bulat yang merupakan target donasi yang harus dicapai. Masukan pada baris berikut dan seterusnya merupakan bilangan bulat yang menyatakan donasi oleh setiap donatur, masukan terus diterima hingga target tercapai.

Keluaran berupa bilangan hasil total penjumlahan tiap perulangannya serta jumlah donatur.

Contoh masukan dan keluaran:

No	Mas	Keluaran
1	300	Donatur 1: Menyumbang 100. Total terkumpul: 100
	100	Donatur 2: Menyumbang 50. Total terkumpul: 150
	50	Donatur 3: Menyumbang 200. Total terkumpul: 350
	200	Target tercapai! Total donasi: 350 dari 3 donatur.
2	500	Donatur 1: Menyumbang 150. Total terkumpul: 150
	150	Donatur 2: Menyumbang 100. Total terkumpul: 250
	100	Donatur 3: Menyumbang 50. Total terkumpul: 300
	50	Donatur 4: Menyumbang 300. Total terkumpul: 600
	300	Target tercapai! Total donasi: 600 dari 4 donatur.
3	200	Donatur 1: Menyumbang 300. Total terkumpul: 300
	300	Target tercapai! Total donasi: 300 dari 1 donatur

Source Code:

```
package main
import "fmt"
func main(){
    var target, donasi, totalDonasi, jumlahDonatur int
    fmt.Scan(&target)

    totalDonasi = 0
    jumlahDonatur = 0

    for totalDonasi < target {
        jumlahDonatur++
        fmt.Printf("Donatur %d: ", jumlahDonatur)
        fmt.Scan(&donasi)

        totalDonasi += donasi
        fmt.Printf("Donatur %d: Menyumbang %d. Total
        terkumpul: %d\n", jumlahDonatur, donasi, totalDonasi)
    }
    fmt.Printf("Target tercapai! Total donasi: %d dari %d
    donatur\n", totalDonasi, jumlahDonatur)
}
```

Output:

```
300
Donatur 1: 100
Donatur 1: Menyumbang 100. Total terkumpul: 100
Donatur 2: 50
Donatur 2: Menyumbang 50. Total terkumpul: 150
Donatur 3: 200
Donatur 3: Menyumbang 200. Total terkumpul: 350
Target tercapai! Total donasi: 350 dari 3 donatur
PS C:\Users\User\Documents\MODUL 13> go run "c:\Users\User\Documents\MODUL 13\Latihan Soal\Latsol3.go"
500
Donatur 1: 150
Donatur 1: Menyumbang 150. Total terkumpul: 150
Donatur 2: 100
Donatur 2: Menyumbang 100. Total terkumpul: 250
Donatur 3: 50
Donatur 3: Menyumbang 50. Total terkumpul: 300
Donatur 4: 300
Donatur 4: Menyumbang 300. Total terkumpul: 600
Target tercapai! Total donasi: 600 dari 4 donatur
PS C:\Users\User\Documents\MODUL 13> go run "c:\Users\User\Documents\MODUL 13\Latihan Soal\Latsol3.go"
200
Donatur 1: 300
Donatur 1: Menyumbang 300. Total terkumpul: 300
Target tercapai! Total donasi: 300 dari 1 donatur
PS C:\Users\User\Documents\MODUL 13> go run "c:\Users\User\Documents\MODUL 13\Latihan Soal\Latsol3.go"
```

Deskripsi Program:

Program Go ini untuk membantu organisasi amal dalam mengumpulkan donasi hingga mencapai target yang telah ditentukan. Setelah meminta pengguna untuk memasukkan target donasi, program akan menginisialisasi total donasi dan jumlah donatur. Selanjutnya, program akan terus meminta input dari pengguna untuk jumlah donasi dari setiap donatur hingga total donasi mencapai atau melebihi target. Setiap kali donatur memberikan sumbangan, program akan mencetak informasi tentang sumbangan tersebut beserta total terkumpul saat itu. Setelah target tercapai, program akan menampilkan total donasi yang berhasil dikumpulkan beserta jumlah donatur yang berkontribusi.

DAFTAR PUSTAKA

School of Computing. *Modul Praktikum 13 – Repeat-Until Algoritma dan Pemrograman 1 SI Informatika*.2024