

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



**DISUSUN OLEH:
FEROS PEDROSA VALENTINO
103112400055
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

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. 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.

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.

Bentuk Repeat-Until

- Berbeda dengan while-loop, di mana <action> akan dieksekusi secara berulang SAMPAI <condition> bernilai true.
- Jumlah iterasi tidak dapat ditentukan, karena bergantung dengan perubahan nilai pada <condition>.
- Pada bahasa Go, tidak ada penulisan secara spesifik untuk repeat-until, jadi alternatifnya bisa menggunakan bentuk while-loop.

CONTOH SOAL

1. Coso1

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:\semester1_alpro1> go run "c:\semester1_alpro1\alpro1_week13\coso1\coso1.go"
pagi 3
pagi
pagi
pagi
PS C:\semester1_alpro1> go run "c:\semester1_alpro1\alpro1_week13\coso1\coso1.go"
kursi 5
kursi
kursi
kursi
kursi
kursi
```

Deskripsi program:

Program di atas ditulis dalam bahasa Go dan berfungsi untuk mencetak sebuah kata sebanyak jumlah yang ditentukan oleh pengguna. Pertama deklarasikan variabel `word` yang bertipe `string` untuk menyimpan kata yang akan dicetak dan variabel `repetitions` yang bertipe `integer` untuk menyimpan jumlah pengulangan. Program kemudian meminta input dari pengguna untuk kedua variabel tersebut. Setelah mendapatkan inputan, program menggunakan `repeat-until` loop yang akan terus mencetak kata yang diberikan hingga jumlah pengulangan yang diinginkan tercapai. Di dalam loop, setiap kali kata dicetak, variabel `counter` akan bertambah satu. Loop akan berhenti ketika nilai `counter` sama dengan atau lebih besar dari `repetitions`, yang ditandai dengan mengubah nilai variabel `done` menjadi `true`.

2. Coso2

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:\semester1_alpro1> go run "c:\semester1_alpro1\alpro1_week13\coso2\coso2.go"
-5
-2
-1
0
5
5 adalah bilangan bulat positif
PS C:\semester1_alpro1> go run "c:\semester1_alpro1\alpro1_week13\coso2\coso2.go"
17
17 adalah bilangan bulat positif
```

Deskripsi program:

Program di atas ditulis dalam bahasa Go dan berfungsi untuk meminta pengguna memasukkan sebuah bilangan bulat positif. Pertama deklarasikan variabel `number` yang bertipe integer untuk menyimpan input bilangan bulat dari pengguna dan variabel `continueLoop` bertipe boolean yang berfungsi sebagai pengontrol untuk loop. Program kemudian menggunakan `repeat-until` loop yang akan terus berjalan selama `continueLoop` bernilai `true`. Di dalam loop, program meminta pengguna untuk memasukkan sebuah bilangan bulat melalui fungsi `fmt.Scan`. Setelah menerima inputan, program memeriksa apakah bilangan yang dimasukkan kurang dari atau sama dengan nol. Jika ya, maka `continueLoop` akan tetap bernilai `true`, dan loop akan berlanjut, meminta input lagi. Namun, jika bilangan yang dimasukkan adalah positif, `continueLoop` akan diubah menjadi `false`, sehingga loop berhenti. Setelah keluar dari loop, program mencetak pesan yang menyatakan bahwa bilangan yang dimasukkan adalah bilangan bulat positif.

3. Coso3

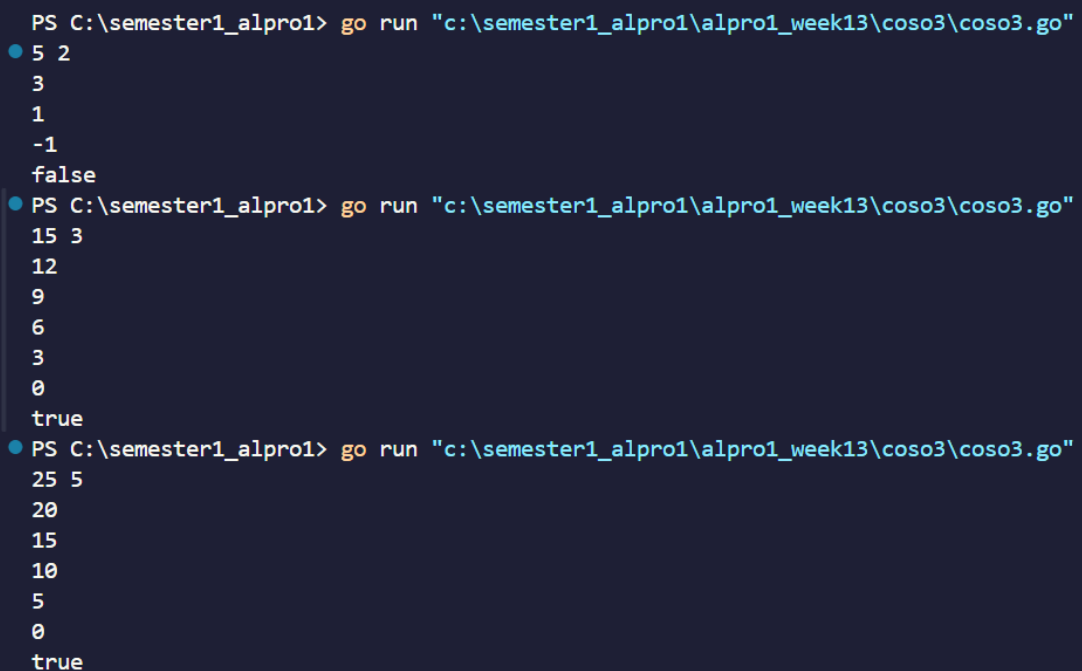
Source code:

```
package main

import "fmt"

func main() {
    var x int
    var 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:\semester1_alpro1> go run "c:\semester1_alpro1\alpro1_week13\coso3\coso3.go"
5 2
3
1
-1
false
PS C:\semester1_alpro1> go run "c:\semester1_alpro1\alpro1_week13\coso3\coso3.go"
15 3
12
9
6
3
0
true
PS C:\semester1_alpro1> go run "c:\semester1_alpro1\alpro1_week13\coso3\coso3.go"
25 5
20
15
10
5
0
true
```

Deskripsi program:

Program di atas ditulis dalam bahasa Go dan berfungsi untuk melakukan pengurangan berulang antara dua bilangan bulat yang diinputkan oleh pengguna. Pertama deklarasi variabel `x` dan `y` yang bertipe integer, serta variabel `selesai` yang bertipe boolean yang digunakan untuk mengontrol loop. Setelah itu, program meminta pengguna untuk memasukkan nilai untuk `x` dan `y` melalui fungsi `fmt.Scan`. Selanjutnya, program menggunakan `repeat-until` loop yang akan terus berjalan selama variabel `selesai` bernilai `false`. Di dalam loop, nilai `y` akan dikurangkan dari `x`, dan hasilnya akan dicetak ke layar. Setelah setiap pengurangan, program memeriksa apakah nilai `x` sudah kurang dari atau sama dengan nol; jika ya, variabel `selesai` akan diubah menjadi `true`, yang akan menghentikan loop. Setelah loop selesai, program akan mencetak hasil evaluasi apakah `x` sama dengan nol.

LATIHAN SOAL

1. Latihan soal 1

Source code:

```
package main

import "fmt"

func main() {
    var bilangan int
    fmt.Scan(&bilangan)
    jumlahDigit := 0

    for {
        jumlahDigit++
        bilangan /= 10
        if bilangan == 0 {
            break
        }
    }
    fmt.Print(jumlahDigit)
}
```

Output:

```
PS C:\semester1_alpro1> go run "c:\semester1_alpro1\alpro1_week13\latsol1\latsol1.go"
5
1
PS C:\semester1_alpro1> go run "c:\semester1_alpro1\alpro1_week13\latsol1\latsol1.go"
234
3
PS C:\semester1_alpro1> go run "c:\semester1_alpro1\alpro1_week13\latsol1\latsol1.go"
78787
5
PS C:\semester1_alpro1> go run "c:\semester1_alpro1\alpro1_week13\latsol1\latsol1.go"
1894256
7
```

Deskripsi program:

Program di atas ditulis dalam bahasa Go dan berfungsi untuk menghitung jumlah digit dari sebuah bilangan bulat yang diinputkan oleh pengguna. Pertama deklarasikan variabel bilangan yang bertipe integer dan meminta pengguna untuk memasukkan nilai melalui fungsi `fmt.Scan`. Selanjutnya, program menetapkan variabel `jumlahDigit` dengan nilai 0, yang akan digunakan untuk menghitung jumlah digit. Program kemudian memasuki sebuah loop tak terbatas di mana setiap iterasi akan menambah nilai `jumlahDigit` sebesar satu dan membagi bilangan dengan 10 untuk menghilangkan digit terakhir. Proses ini akan terus berlanjut hingga bilangan menjadi 0, pada titik ini program akan menghentikan loop dengan pernyataan `break`. Setelah loop selesai, program mencetak nilai `jumlahDigit`, yang merupakan total jumlah digit dari bilangan yang diinputkan.

2. Latihan soal 2

Source code:

```
package main

import "fmt"

func main() {
    var number, jumlah float64
    fmt.Scan(&number)
    jumlah = number + 0.1
    for jumlah <= float64(int(number)+1) {
        fmt.Printf("%.1f\n", jumlah)
        jumlah += 0.1
    }
}
```

Output:

```
PS C:\semester1_alpro1> go run "c:\semester1_alpro1\alpro1_week13\latsol2\latsol2.go"
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
```

Deskripsi program:

Program di atas ditulis dalam bahasa Go dan berfungsi untuk membaca inputan pengguna kemudian mencetak bilangan desimal, dengan penambahan 0.1 setiap iterasi, lalu program akan menampilkan serangkaian bilangan desimal yang dibulatkan ke atas sampai mencapai bilangan bulat yang lebih besar dari input awal. Pertama deklarasi variabel `number` bertipe `float64` yang berguna untuk menampung input dari pengguna dan variabel `jumlah` bertipe `float64` yang berguna untuk variabel iterasi. Setelah pengguna memasukkan bilangan desimal program akan menghitung nilai awal `jumlah` dengan menambahkan 0.1 pada angka input. Program kemudian menggunakan `repeat-until` loop untuk mencetak nilai `jumlah` dengan format desimal satu angka dibelakang koma. Iterasi berlangsung dengan menambahkan 0.1 ke `jumlah` disetiap langkah hingga nilainya melebihi bilangan bulat terbesar yang lebih kecil atau sama dengan input awal yang ditambah satu.

3. Latihan soal 3

Source code:

```
package main

import "fmt"

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

    for totalDonasi < target {
        fmt.Scan(&donasi)
        jumlahDonatur++
        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:

```
PS C:\semester1_alpro1> go run "c:\semester1_alpro1\alpro1_week13\latsol3\latsol3.go"
300
100
Donatur 1: Menyumbang 100. Total terkumpul: 100
50
Donatur 2: Menyumbang 50. Total terkumpul: 150
200
Donatur 3: Menyumbang 200. Total terkumpul: 350
Target tercapai! Total donasi: 350 dari 3 donatur.
PS C:\semester1_alpro1> go run "c:\semester1_alpro1\alpro1_week13\latsol3\latsol3.go"
500
150
Donatur 1: Menyumbang 150. Total terkumpul: 150
100
Donatur 2: Menyumbang 100. Total terkumpul: 250
50
Donatur 3: Menyumbang 50. Total terkumpul: 300
300
Donatur 4: Menyumbang 300. Total terkumpul: 600
Target tercapai! Total donasi: 600 dari 4 donatur.
PS C:\semester1_alpro1> go run "c:\semester1_alpro1\alpro1_week13\latsol3\latsol3.go"
200
300
Donatur 1: Menyumbang 300. Total terkumpul: 300
Target tercapai! Total donasi: 300 dari 1 donatur.
```

Deskripsi program:

Program di atas ditulis dalam bahasa Go dan digunakan untuk mengumpulkan donasi hingga mencapai target tertentu. Pertama deklarasi variabel target, donasi, totalDonasi, jumlahDonatur bertipe integer. Selanjutnya, pengguna diminta untuk memasukkan jumlah target donasi yang diinginkan. Setelah target ditentukan, program akan terus meminta input dari pengguna berupa jumlah donasi yang diberikan oleh setiap donatur. Setiap kali donatur memberikan sumbangan, program akan mencatat jumlah donatur dan total donasi yang telah terkumpul, serta menampilkan informasi tentang sumbangan yang diberikan dan total yang telah terkumpul hingga saat itu. Proses ini akan berlanjut hingga total donasi mencapai

atau melebihi target yang telah ditetapkan. Setelah target tercapai, program akan menampilkan pesan yang menyatakan bahwa target telah tercapai, beserta total donasi yang terkumpul dan jumlah donatur yang berpartisipasi.

DAFTAR PUSTAKA

Prasti Eko Yunanto, S.T., M.Kom. (2024). MODUL PRAKTIKUM 13 – REPEAT-UNTIL
ALGORITMA DAN PEMROGRAMAN 1 S1 INFORMATIKA (MODUL 13)