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

DISUSUN OLEH:

DAFFA TSAQIFNA FAUZTSANY

103112400032

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:

1. Paradigma Perulangan

- Perulangan adalah struktur kontrol yang memungkinkan instruksi yang sama dilakukan secara berulang berdasarkan suatu kondisi.
- **Repeat-Until** berbeda dengan **while-loop** karena:
    - While-loop mengevaluasi kondisi di awal, sedangkan repeat-until mengevaluasi kondisi di akhir.
    - Repeat-until menjamin aksi minimal dilakukan sekali sebelum pengecekan kondisi.

2. Karakteristik Repeat-Until

- **Aksi**: Instruksi yang dilakukan berulang. Dilakukan minimal satu kali.
- **Kondisi berhenti**: Perulangan akan berhenti jika kondisi bernilai **true**. Kondisi ini adalah komplemen dari kondisi perulangan.

3. Notasi Repeat-Until

- **Pseudocode**:

```
repeat
   // aksi
until kondisi
```

- Pada bahasa **Go**, konsep repeat-until direalisasikan dengan penggunaan **for-loop** dengan kondisi yang diperiksa di akhir iterasi.

4. Implementasi

- Modul ini memberikan beberapa contoh implementasi repeat-until dalam bahasa Go, seperti:
    - **Validasi login**: Memverifikasi username dan password.
    - **Deret Fibonacci**: Menghitung dan mencetak bilangan Fibonacci hingga batas tertentu.
    - **Penerimaan input positif**: Memastikan input pengguna adalah bilangan positif.
    - **Cek kelipatan bilangan**: Mengevaluasi apakah satu bilangan merupakan kelipatan dari bilangan lainnya.

5. Keuntungan Repeat-Until

- Aksi minimal dijalankan satu kali tanpa memandang kondisi awal.
- Lebih mudah digunakan jika aksi perlu dijalankan sebelum kondisi dievaluasi.

Contoh Implementasi Pseudocode:

- **Login Sederhana:**

```
repeat
```

```
    input username, password
until username = "admin" and password = "admin12345"
print "Login berhasil"
```

- Pengulangan Deret Fibonacci:

```
repeat
    hitung bilangan Fibonacci berikutnya
    cetak bilangan tersebut
until bilangan Fibonacci > batas maksimum
```

CONTOH SOAL

Conso 1:

```
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 D:\test bs> go run  'd:\test bs\lab shit\10th meet\conso10\conso10-1.go'
pagi 3
pagi
pagi
pagi
PS D:\test bs> go run  'd:\test bs\lab shit\10th meet\conso10\conso10-1.go'
kursi 5
kursi
kursi
kursi
kursi
kursi
```

Deskripsi Program:

Program ini berfungsi untuk mencetak sebuah kata (string) beberapa kali sesuai dengan jumlah yang diminta oleh pengguna.

1. **Deklarasi variabel:**

```
var word string
var repetitions int
```

- word menyimpan kata yang akan dicetak.
- repetitions menyimpan jumlah berapa kali kata tersebut akan dicetak.

2. Input:

```
fmt.Scan(&word, &repetitions)
```

- Program menerima dua input dari pengguna: sebuah kata (word) dan jumlah pengulangan (repetitions).

3. Inisialisasi Counter:

```
counter := 0
```

- counter digunakan untuk menghitung berapa kali kata sudah dicetak.

4. Looping dengan Kondisi done:

```
for done := false; !done; {
    fmt.Println(word)
    counter++
    done = (counter >= repetitions)
}
```

- **done := false**: Variabel done digunakan untuk mengendalikan apakah loop harus berhenti.
- **!done**: Loop terus berjalan selama done bernilai false.
- **fmt.Println(word)**: Mencetak kata yang dimasukkan oleh pengguna.
- **counter++**: Setiap kali kata dicetak, counter akan bertambah 1.
- **done = (counter >= repetitions)**: Setelah setiap pengulangan, program memeriksa apakah jumlah pengulangan sudah mencapai nilai yang diinginkan. Jika ya, done akan menjadi true, dan loop akan berhenti.

Contoh: Jika pengguna memasukkan

```
hello 3 (input)

hello (output)

hello (output)

hello (output)
```

2.cosol2

```
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 D:\test bs> go run  'd:\test bs\lab shit\10th meet\conso10\conso10-2.go'
-5
-2
-1
0
5
5 adalah bilangan bulat positif
PS D:\test bs> go run  'd:\test bs\lab shit\10th meet\conso10\conso10-2.go'
17
17 adalah bilangan bulat positif
```

Deskripsi Program:
Program ini akan meminta input dari pengguna dan terus melakukan loop sampai pengguna memasukkan angka positif. Setelah itu, program akan mencetak bahwa angka tersebut adalah bilangan bulat positif.

1. **Deklarasi variabel:**

```
var number int
var continueLoop bool
```

- number digunakan untuk menyimpan input bilangan dari pengguna.
- continueLoop adalah variabel boolean yang mengendalikan apakah loop harus terus berjalan atau tidak.

2. Looping dengan Kondisi continueLoop:

```
for continueLoop = true; continueLoop; {
    fmt.Scan(&number)
    continueLoop = number <= 0
}
```

- Loop akan terus berjalan selama continueLoop bernilai true.

- Di dalam loop, program menerima input dari pengguna menggunakan fmt.Scan(&number).
- Setelah itu, kondisi continueLoop = number <= 0 akan mengecek apakah number yang dimasukkan lebih kecil atau sama dengan 0. Jika number kurang dari atau sama dengan 0, maka continueLoop akan tetap bernilai true dan loop akan berlanjut. Jika number lebih besar dari 0, maka continueLoop akan bernilai false dan loop berhenti.

3. **Output::**

```
fmt.Printf("%d adalah bilangan bulat positif\n", number)
```

- Setelah loop berhenti (ketika pengguna memasukkan angka positif), program mencetak pesan yang menyatakan bahwa angka yang dimasukkan adalah bilangan bulat positif.

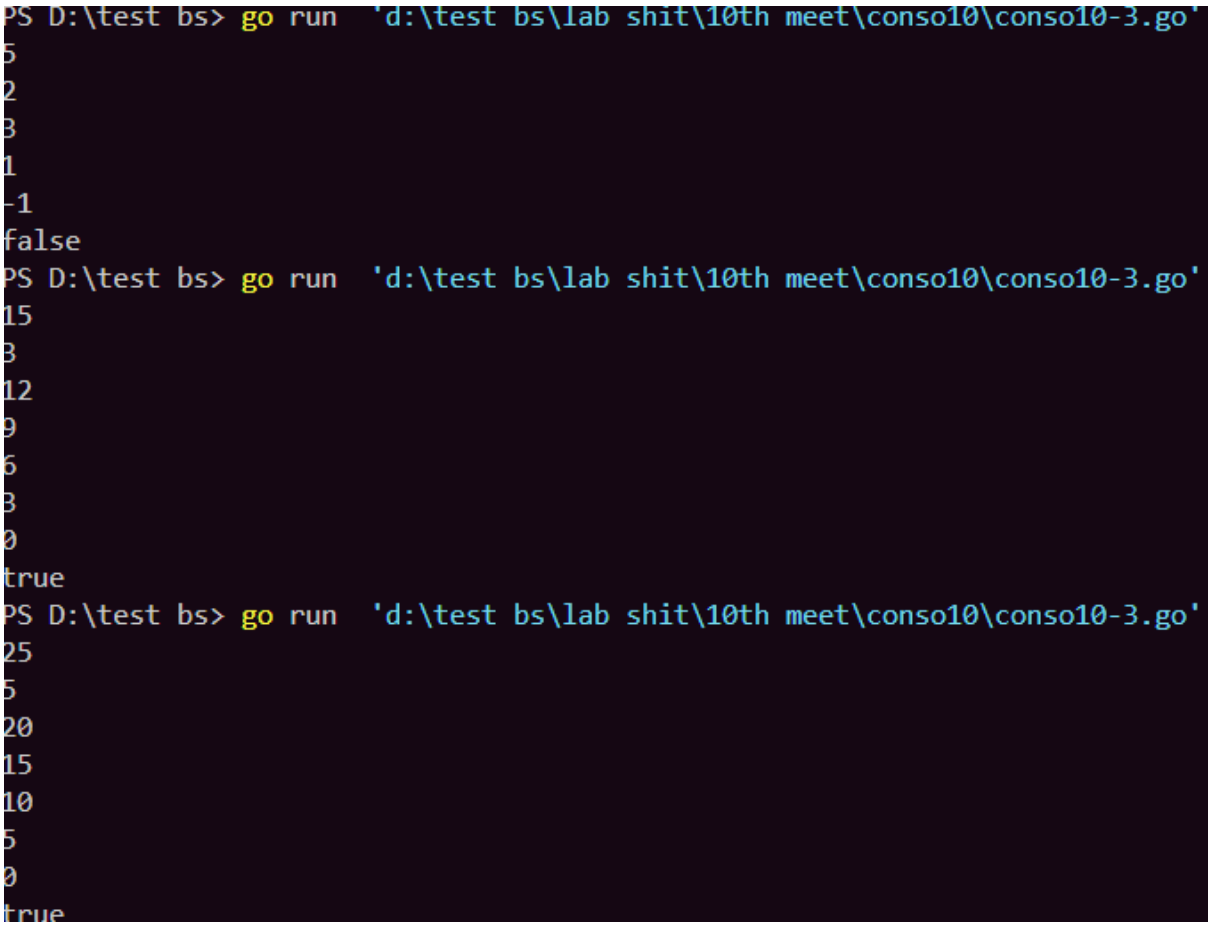
Contoh: Jika pengguna memasukkan

```
-3 (input)
-5 (input)
7 (input)
7 adalah bilangan bulat positif  (output)
```

3.cosol3

```
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:



Deskripsi Program:

Program ini akan mengurangi nilai x dengan y berulang kali, mencetak hasil setelah setiap pengurangan, dan berhenti ketika x menjadi 0 atau kurang dari 0. Di akhir program, akan mencetak apakah nilai x akhirnya menjadi 0 atau tidak.

1. **Deklarasi variabel:**

```
var x int
var y int
var selesai bool
```

- x dan y adalah variabel integer yang masing-masing akan menyimpan input dari pengguna.
- selesai adalah variabel boolean yang digunakan untuk menentukan apakah loop harus berhenti.

2. Input Pengguna:

```
fmt.Scan(&x, &y)
```

- Program meminta pengguna untuk memasukkan dua nilai: `x` (angka yang akan dikurangi) dan `y` (angka pengurang).

3. Looping dengan Kondisi selesai:

```
for selesai = false; !selesai; {
   x = x - y
   fmt.Println(x)
   selesai = x <= 0
}
```

- Loop berjalan selama selesai bernilai false.
- Dalam setiap iterasi, x dikurangi dengan y, dan hasilnya dicetak menggunakan fmt.Println(x).
- Setelah setiap pengurangan, selesai diubah untuk mengecek apakah x sudah kurang dari atau sama dengan 0. Jika x sudah <= 0, maka loop berhenti.

4. Output Akhir:

```
fmt.Println(x == 0)
```

- Setelah loop berhenti, program akan mencetak apakah nilai x sama dengan 0 atau tidak (true jika 0, false jika tidak).

Contoh: Jika pengguna memasukkan

```
10 3 (input)
7 (output)
4 (output)
1 (output)
false (output)
```

SOAL LATIHAN

1. Latihan1

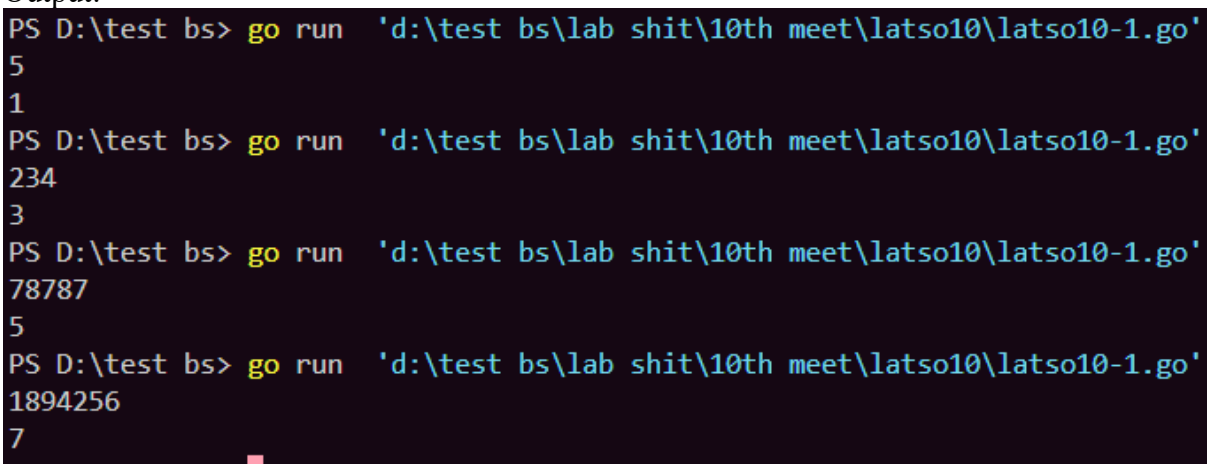
Source Code:

```
package main

import "fmt"

func main() {
   var x, y int
   fmt.Scan(&x)
   for x%10 != 0 {
      x = x / 10
      y++
   }
   fmt.Print(y)
}
```

Output:



Deskripsi Program:

Program ini menghitung jumlah digit dari bilangan bulat positif x. Setiap digit dihitung dengan membagi x secara berulang-ulang dengan 10 hingga nilai x menjadi nol.

1. **Deklarasi variabel:**

```
var x, y int
```

- x adalah bilangan input dari pengguna.
- y adalah penghitung jumlah digit pada bilangan x.

2. Input Nilai x:

```
fmt.Scan(&x)
```

- Pengguna diminta memasukkan bilangan bulat positif (x).

3. Looping untuk Menghitung Digit::

```
for x > 0 {
   x = x / 10
   y++
```

```
}
```

- Loop terus berjalan selama x lebih besar dari nol.
- Di setiap iterasi:
    - Nilai x dibagi 10 (x = x / 10) untuk menghilangkan digit terakhir.
    - Variabel penghitung y ditambah 1 (y++).

4. Output Hasil:

```
fmt.Print(y)
```

- Setelah loop selesai, nilai `y` (jumlah digit) dicetak.

Contoh: Jika pengguna memasukkan

```
5 (input)

1 (output)
```

2. Latihan2

Source Code:

```
package main

import "fmt"

func main() {
   var x, y float32
   var z int
   fmt.Scan(&x)
   z = int(x)
   y = float32(z) + 1
   for x != float32(y) {
      x = float32(int(x*10)+1) / 10
      fmt.Println(x)
   }
}
```

Output:

```
PS D:\test bs> go run  'd:\test bs\lab shit\10th meet\latso10\latso10-2.go'
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
PS D:\test bs> go run  'd:\test bs\lab shit\10th meet\latso10\latso10-2.go'
2.7
2.8
2.9
3
```

Deskripsi Program:

Program ini bertujuan untuk menaikkan nilai desimal dari bilangan pecahan (float32) secara bertahap hingga mencapai bilangan bulat berikutnya. Program mencetak setiap perubahan nilai x selama proses berlangsung.

1. **Deklarasi variabel:**

```
var x, y float32
var z int
```

- `x`: Nilai awal berupa bilangan pecahan (float32) yang akan diproses.
- `z`: Bagian bulat dari `x` (dibulatkan ke bawah).
- `y`: Bilangan bulat berikutnya setelah bagian bulat dari `x`.

2. Input:

```
fmt.Scan(&x)
```

- Pengguna memasukkan nilai x sebagai bilangan pecahan.

3. Inisialisasi Variabel z dan y:

```
z = int(x)
y = float32(z) + 1
```

- z menyimpan bagian bilangan bulat dari x.
- y adalah bilangan bulat berikutnya setelah

4. Perulangan untuk Menaikkan Nilai x:

```
fmt.Scan(&x)
```

- Loop berjalan hingga nilai `x` sama dengan bilangan bulat `y`.
- Dalam setiap iterasi:
    - x dikalikan 10 untuk memindahkan desimal ke tempat yang lebih besar, kemudian ditambahkan 1.
    - Hasilnya dibagi 10 untuk mengembalikan desimal ke posisi semula.
- Nilai baru `x` dicetak dalam setiap iterasi

Contoh: Jika pengguna memasukkan

```
Input: 2.3(input)
2.4 (output)
2.5 (output)
2.6 (output)
2.7 (output)
2.8 (output)
2.9 (output)
3.0 (output)
```

3. Latihan3

Source Code:

```
package main

import "fmt"

func main() {
   var x, y, z, i int
   fmt.Scan(&x)
   for z < x {
      i++
      fmt.Scan(&y)
      z = z + y
      fmt.Printf("Donatur %d: Menyumbang %d. Total terkumpul: %d\n", i, y, z)
   }
   fmt.Printf("Target tercapai! Total donasi: %d dari %d donatur.", z, i)
}
```

Output:

```
PS D:\test bs> go run  'd:\test bs\lab shit\10th meet\latso10\latso10-3.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 D:\test bs> go run  'd:\test bs\lab shit\10th meet\latso10\latso10-3.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 D:\test bs> go run  'd:\test bs\lab shit\10th meet\latso10\latso10-3.go'
200
300
Donatur 1: Menyumbang 300. Total terkumpul: 300
Target tercapai! Total donasi: 300 dari 1 donatur.
```

Deskripsi Program:

Program ini berfungsi untuk mencatat donasi yang diberikan oleh beberapa donatur hingga target donasi tercapai. Setiap kali seorang donatur menyumbang, program mencatat jumlah

sumbangan tersebut, menambahkannya ke total yang terkumpul, dan menampilkan informasi mengenai donasi terkini. Program berhenti setelah target donasi tercapai dan memberikan laporan akhir.

1. **Deklarasi variabel:**

```
var x, y, z, i int
```

- x: Target donasi yang harus dicapai.
- y: Jumlah donasi yang diberikan oleh seorang donatur.
- z: Total donasi yang terkumpul sejauh ini.
- i: Jumlah donatur yang telah menyumbang.

2. Input Target Donasi:

```
fmt.Scan(&x)
```

- Program meminta pengguna memasukkan target donasi `x`.

3. Proses Donasi (Perulangan):

```
for z < x {
   i++
   fmt.Scan(&y)
   z = z + y
   fmt.Printf("Donatur %d: Menyumbang %d. Total terkumpul: %d\n", i, y, z)
}
```

- Selama `z` (total donasi) kurang dari `x` (target donasi):
    - Program meminta input donasi dari seorang donatur (y).
    - Menambah jumlah donatur (i).
    - Menambahkan nilai donasi tersebut ke total donasi (z).
    - Menampilkan informasi donatur dan total terkumpul sejauh ini.

4. Laporan Akhir:

```
fmt.Printf("Target tercapai! Total donasi: %d dari %d donatur.", z, i)
```

- Setelah target donasi tercapai, program mencetak total donasi terkumpul dan jumlah donatur yang berpartisipasi.

Contoh: Jika pengguna memasukkan

```
200 (input)

300 (input)

Donatur 1: Menyumbang 300. Total terkumpul: 300 (output)

Target tercapai! Total donasi: 300 dari 1 donatur. (output)
```