

**LAPORAN PRAKTIKUM**  
**ALGORITMA DAN PEMROGRAMAN 1**  
**MODUL 13**  
**“TIPE DATA & VARIABEL”**



**DISUSUN OLEH:**  
**MUHAMMAD ZAKY MUBAROK**  
**103112400073 S1**  
**IF-12-01**  
**DOSEN:**  
**Yohani Setiya Rafika Nur, M. Kom.**

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**  
**FAKULTAS INFORMATIKA**  
**TELKOM UNIVERSITY PURWOKERTO**

## DASAR TEORI

### Dasar Teori Repeat-Until

#### Paradigma Perulangan

Perulangan merupakan struktur kontrol yang memungkinkan suatu instruksi dilakukan berulang kali dalam waktu atau jumlah yang lama. Ini sangat berguna untuk menghindari penulisan instruksi yang sama berulang-ulang. Pada modul sebelumnya, kita telah mempelajari while-loop. Selanjutnya, kita akan mempelajari perulangan dengan repeat-until.

#### Perbedaan While-Loop dan Repeat-Until

- **While-Loop:** Perulangan berdasarkan kondisi perulangan. Perulangan terjadi selama kondisi bernilai true.
  - Contoh: "Menulis teks tertentu selama tinta pena masih ada."
- **Repeat-Until:** Perulangan berdasarkan kondisi berhenti. Perulangan terjadi sampai kondisi tertentu terpenuhi, yaitu ketika kondisi bernilai true.
  - Contoh: "Menulis teks tertentu sampai tinta pena habis."

#### Komplemen Kondisi

Kondisi perulangan dan kondisi berhenti memiliki keterhubungan sifat komplemen:

- **While-Loop:** Kondisi perulangan (kapan perulangan terjadi).
- **Repeat-Until:** Kondisi berhenti (kapan perulangan dihentikan).
  - Komplemen kondisi "tinta pena masih ada" adalah "tinta pena habis".
  - Komplemen kondisi "saya masih lapar" adalah "saya merasa kenyang".

#### Karakteristik Repeat-Until

Komponen dari repeat-until sama dengan while-loop, tetapi struktur penulisannya berbeda.

##### 1. Aksi:

- Kumpulan instruksi yang dijalankan dalam perulangan.
- Aksi minimal dijalankan sekali sebelum pengecekan kondisi berhenti.

##### 2. Kondisi Berhenti:

- Kondisi yang menyebabkan perulangan berhenti.
- Kondisi ini harus bernilai false selama perulangan berlangsung.

## Contoh Repeat-Until dalam Pseudocode

Berikut adalah pseudocode untuk repeat-until:

**repeat**

**aksi**

**until (kondisi\_berhenti)**

## Contoh Kasus Nyata

- **While-Loop:** "Saya makan suap demi suap selama saya masih lapar."
- **Repeat-Until:** "Saya makan suap demi suap sampai saya merasa kenyang."

## Implementasi dalam Bahasa Pemrograman

- **Pascal:** Menggunakan repeat-until.
- **C/C++:** Menggunakan do-while.
- **Go:** Tidak memiliki instruksi eksplisit untuk repeat-until, namun dapat disimulasikan menggunakan for.

## Simulasi Repeat-Until dalam Go

Meskipun Go tidak memiliki instruksi repeat-until secara eksplisit, kita bisa menggunakan for loop untuk mensimulasikannya:

**go**

**package main**

**import (**

**"fmt"**

**)**

**func main() {**

**var n, i int**

**i = 1**

**for {**

**fmt.Println(i)**

**i++**

```
    if  $i > n$  {  
        break  
    }  
}  
}
```

**Dalam contoh di atas, perulangan dilakukan sampai kondisi  $i > n$  terpenuhi, yang merupakan kondisi berhenti untuk simulasi repeat-until.**

## CONTOH SOAL

### 1. Latihan1

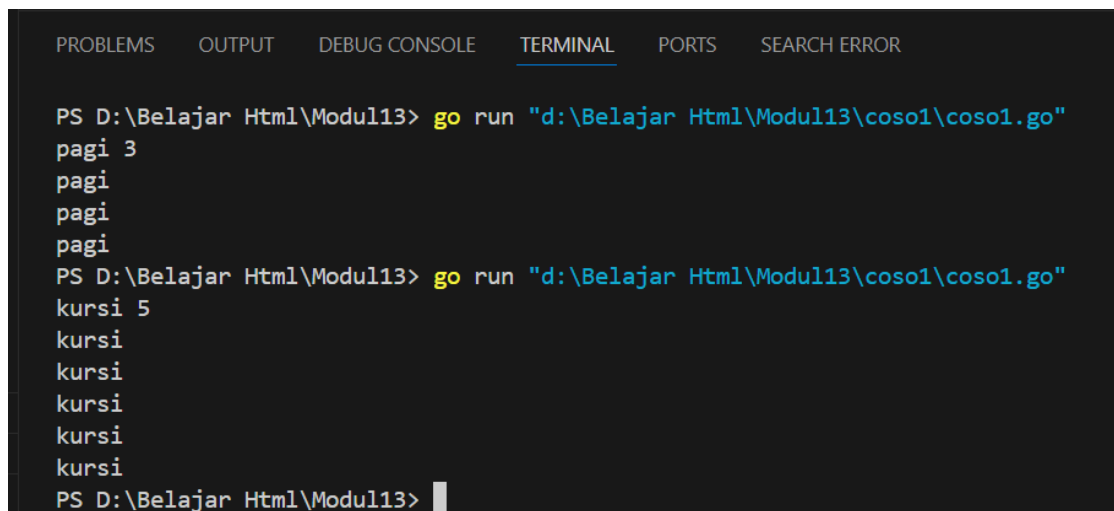
Source Code:

```
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:



The screenshot shows a terminal window with a dark background and light-colored text. At the top, there are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is active), 'PORTS', and 'SEARCH ERROR'. The terminal shows the command prompt 'PS D:\Belajar Html\Modul13>' followed by the command 'go run "d:\Belajar Html\Modul13\coso1\coso1.go"'. The output of the program is 'pagi 3' followed by three lines of 'pagi'. Then, the command is run again, and the output is 'kursi 5' followed by five lines of 'kursi'. The prompt is visible at the end of the last line.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SEARCH ERROR

PS D:\Belajar Html\Modul13> go run "d:\Belajar Html\Modul13\coso1\coso1.go"
pagi 3
pagi
pagi
pagi
PS D:\Belajar Html\Modul13> go run "d:\Belajar Html\Modul13\coso1\coso1.go"
kursi 5
kursi
kursi
kursi
kursi
kursi
PS D:\Belajar Html\Modul13> 
```

## **Deskripsi Program:**

### **1. Deklarasi Variabel:**

- **word:** Variabel untuk menyimpan kata yang diinputkan oleh pengguna.
- **repetitions:** Variabel untuk menyimpan jumlah pengulangan yang diinginkan oleh pengguna.
- **counter:** Variabel penghitung yang diinisialisasi dengan nilai 0 untuk menghitung jumlah pengulangan yang telah dilakukan.

### **2. Membaca Input:**

- Program membaca nilai word dan repetitions yang diinputkan oleh pengguna menggunakan `fmt.Scan(&word, &repetitions)`.

### **3. Inisialisasi Variabel:**

- counter diinisialisasi dengan nilai 0.
- done adalah kondisi boolean yang digunakan untuk menghentikan perulangan, diinisialisasi dengan nilai false.

### **4. Perulangan for:**

- Perulangan dijalankan dengan kondisi `for done := false; !done;`
- Di dalam perulangan:
  - Program mencetak nilai word.
  - counter ditambah satu setiap kali iterasi (`counter++`).
  - Kondisi done diubah menjadi true jika nilai counter lebih besar atau sama dengan repetitions (`done = counter >= repetitions`).

## **Contoh Penggunaan:**

Misalkan pengguna menginputkan nilai word sebagai "Halo" dan repetitions sebagai 3. Maka program akan menghasilkan output berikut:

Halo

Halo

Halo

## **Penjelasan:**

- Iterasi 1: word = Halo, counter = 1, done = false
- Iterasi 2: word = Halo, counter = 2, done = false
- Iterasi 3: word = Halo, counter = 3, done = true

## CONTOH SOAL

### 2. Latihan2

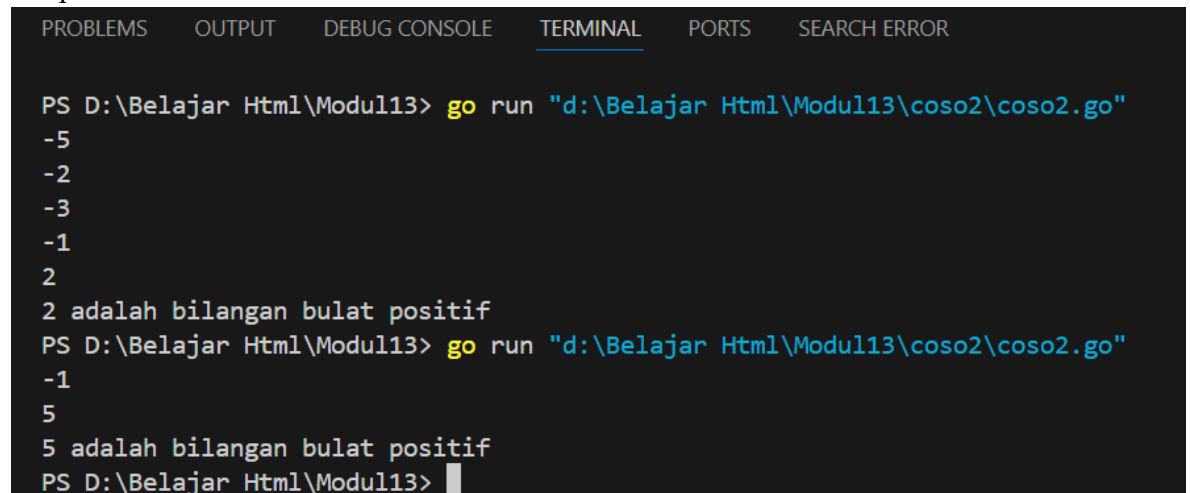
Source Code:

```
package main

import "fmt"

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

Output:

A screenshot of a terminal window showing the execution of a Go program. The terminal has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (selected), PORTS, and SEARCH ERROR. The command 'go run "d:\Belajar Html\Modul13\coso2\coso2.go"' is entered twice. The first run produces the output: -5, -2, -3, -1, 2, followed by the message '2 adalah bilangan bulat positif'. The second run produces the output: -1, 5, followed by the message '5 adalah bilangan bulat positif'. The prompt 'PS D:\Belajar Html\Modul13>' is visible at the end of the second run.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SEARCH ERROR

PS D:\Belajar Html\Modul13> go run "d:\Belajar Html\Modul13\coso2\coso2.go"
-5
-2
-3
-1
2
2 adalah bilangan bulat positif
PS D:\Belajar Html\Modul13> go run "d:\Belajar Html\Modul13\coso2\coso2.go"
-1
5
5 adalah bilangan bulat positif
PS D:\Belajar Html\Modul13> 
```

## **Deskripsi Program:**

### **1. Deklarasi Variabel:**

- **number:** Variabel untuk menyimpan bilangan bulat yang diinputkan oleh pengguna.
- **cotinueLoop:** Variabel boolean yang digunakan untuk mengontrol perulangan.

### **2. Inisialisasi Perulangan:**

- **cotinueLoop** diinisialisasi dengan nilai **true** untuk memulai perulangan.

### **3. Perulangan for:**

- Perulangan dijalankan selama **cotinueLoop** bernilai **true**.
- Di dalam perulangan:
  - Program membaca nilai **number** yang diinputkan oleh pengguna menggunakan **fmt.Scan(&number)**.
  - Kondisi **cotinueLoop** diatur menjadi **true** jika **number** kurang dari atau sama dengan 0 (**cotinueLoop = number <= 0**).
- Perulangan berhenti ketika pengguna memasukkan bilangan yang lebih besar dari 0 (kondisi **number > 0**).

### **4. Mencetak Pesan:**

- Setelah perulangan berhenti, program mencetak nilai **number** dengan pesan bahwa bilangan tersebut adalah bilangan bulat positif (**fmt.Printf("%d adalah bilangan bulat positif\n", number)**).

## **Contoh Penggunaan:**

Misalkan pengguna menginputkan nilai:

-3

0

5

Maka program akan menghasilkan output berikut:

**5 adalah bilangan bulat positif**

## **Penjelasan:**

- Iterasi 1: **number = -3, cotinueLoop = true**
- Iterasi 2: **number = 0, cotinueLoop = true**
- Iterasi 3: **number = 5, cotinueLoop = false**



## CONTOH SOAL

### 3. Latihan3

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 D:\Belajar Html\Modul13> go run "d:\Belajar Html\Modul13\coso3\coso3.go"
5 2
3
1
-1
false
PS D:\Belajar Html\Modul13> go run "d:\Belajar Html\Modul13\coso3\coso3.go"
15 3
12
9
6
3
0
true
```

## Deskripsi Program:

### 1. Deklarasi Variabel:

- **x dan y:** Variabel untuk menyimpan dua bilangan bulat yang diinputkan oleh pengguna.
- **selesai:** Variabel boolean yang digunakan untuk mengontrol perulangan.

### 2. Membaca Input:

- Program membaca nilai **x** dan **y** yang diinputkan oleh pengguna menggunakan `fmt.Scan(&x, &y)`.

### 3. Inisialisasi Variabel:

- **selesai** diinisialisasi dengan nilai `false` untuk memulai perulangan.

### 4. Perulangan for:

- Perulangan dijalankan dengan kondisi `for selesai = false; !selesai;`
- Di dalam perulangan:
  - `x = x - y`: Mengurangi nilai **x** dengan **y**.
  - `fmt.Println(x)`: Mencetak nilai **x** yang telah diperbarui.
  - `selesai = x <= 0`: Memeriksa apakah **x** sudah kurang dari atau sama dengan nol. Jika ya, perulangan berhenti (`selesai = true`).

### 5. Mencetak Hasil Akhir:

- Setelah perulangan selesai, program mencetak hasil perbandingan apakah **x** sama dengan nol (`fmt.Println(x == 0)`).

## Contoh Penggunaan:

Misalkan pengguna menginputkan nilai **x** sebagai 10 dan **y** sebagai 3. Maka program akan menghasilkan output berikut:

7

4

1

-2

false

## Penjelasan:

- Iterasi 1: **x** = 10, **y** = 3, **x** menjadi 7 -> Output: 7
- Iterasi 2: **x** = 7, **y** = 3, **x** menjadi 4 -> Output: 4
- Iterasi 3: **x** = 4, **y** = 3, **x** menjadi 1 -> Output: 1

- Iterasi 4:  $x = 1$ ,  $y = 3$ ,  $x$  menjadi  $-2$  -> Output:  $-2$
- Setelah keluar dari perulangan, program mencetak false karena  $x$  tidak sama dengan  $0$ .

## SOAL LATIHAN

### Statement perulangan

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 D:\Belajar Html\Modul13> go run "d:\Belajar Html\Modul13\latsol1\latsol1.go"
5
1
PS D:\Belajar Html\Modul13> go run "d:\Belajar Html\Modul13\latsol1\latsol1.go"
234
3
PS D:\Belajar Html\Modul13> go run "d:\Belajar Html\Modul13\latsol1\latsol1.go"
78787
5
PS D:\Belajar Html\Modul13> go run "d:\Belajar Html\Modul13\latsol1\latsol1.go"
1894256
7
PS D:\Belajar Html\Modul13> 
```

## **Deskripsikan Program :**

### **1. Deklarasi Variabel:**

- **bilangan:** Variabel untuk menyimpan bilangan bulat yang diinputkan oleh pengguna.
- **jumlahDigit:** Variabel untuk menyimpan jumlah digit dari bilangan, diinisialisasi dengan nilai 0.

### **2. Membaca Input:**

- Program membaca nilai bilangan yang diinputkan oleh pengguna menggunakan `fmt.Scan(&bilangan)`.

### **3. Inisialisasi Variabel:**

- **jumlahDigit** diinisialisasi dengan nilai 0.

### **4. Perulangan for:**

- Perulangan dijalankan tanpa kondisi awal (`for {}`) sebagai simulasi repeat-until.
- Di dalam perulangan:
  - **jumlahDigit++:** Menambah nilai jumlahDigit sebesar 1 setiap kali iterasi.
  - **bilangan /= 10:** Membagi nilai bilangan dengan 10 dan membuang bagian desimal untuk mengurangi satu digit dari bilangan.
  - **if bilangan == 0:** Mengecek apakah bilangan sudah menjadi 0. Jika ya, perulangan dihentikan menggunakan `break`.

### **5. Mencetak Jumlah Digit:**

- Setelah perulangan selesai, program mencetak nilai jumlahDigit yang merupakan jumlah digit dari bilangan awal (`fmt.Print(jumlahDigit)`).

#### **Contoh Penggunaan:**

Misalkan pengguna menginputkan nilai 12345. Maka program akan menghasilkan output berikut:

**5**

#### **Penjelasan:**

- Iterasi 1: **bilangan** = 12345, **jumlahDigit** = 1, **bilangan** menjadi 1234
- Iterasi 2: **bilangan** = 1234, **jumlahDigit** = 2, **bilangan** menjadi 123
- Iterasi 3: **bilangan** = 123, **jumlahDigit** = 3, **bilangan** menjadi 12
- Iterasi 4: **bilangan** = 12, **jumlahDigit** = 4, **bilangan** menjadi 1
- Iterasi 5: **bilangan** = 1, **jumlahDigit** = 5, **bilangan** menjadi 0 (perulangan berhenti)

## SOAL LATIHAN

### 2. Latihan2 Source

Code:

```
package main

import (
    "fmt"
    "math"
)

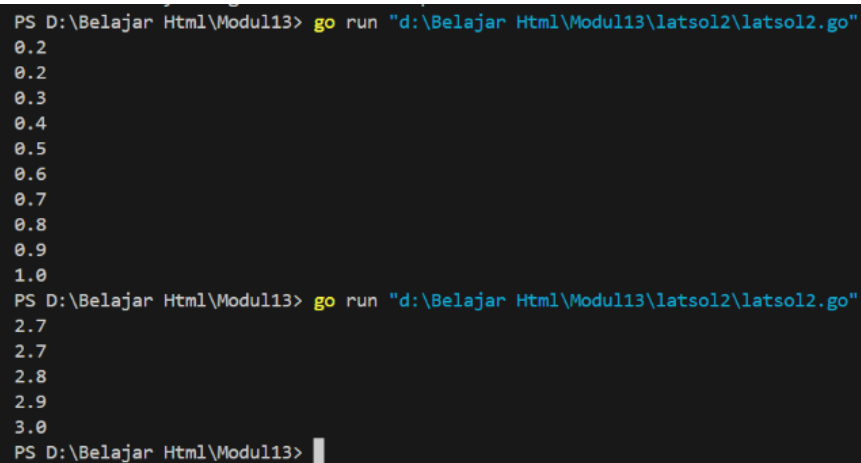
func main() {
    var bilangan float64
    var selesai bool

    fmt.Scan(&bilangan)

    saatIni := bilangan
    batasAtas := math.Ceil(bilangan)

    for selesai = false; !selesai; {
        fmt.Printf("%.1f\n", saatIni)
        saatIni += 0.1
        saatIni = math.Round(saatIni*10) / 10
        selesai = saatIni > batasAtas
    }
}
```

Output:



```
PS D:\Belajar Html\Modul13> go run "d:\Belajar Html\Modul13\latsol2\latsol2.go"
0.2
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
PS D:\Belajar Html\Modul13> go run "d:\Belajar Html\Modul13\latsol2\latsol2.go"
2.7
2.7
2.8
2.9
3.0
PS D:\Belajar Html\Modul13> █
```

## Deskripsi Program:

### 1. Deklarasi Variabel:

- **bilangan:** Variabel untuk menyimpan bilangan desimal yang diinputkan oleh pengguna.
- **selesai:** Variabel boolean yang digunakan untuk mengontrol perulangan.
- **saatIni:** Variabel untuk menyimpan nilai saat ini dari bilangan selama perulangan.
- **batasAtas:** Variabel untuk menyimpan nilai yang dibulatkan ke atas dari bilangan menggunakan `math.Ceil`.

### 2. Membaca Input:

- Program membaca nilai bilangan yang diinputkan oleh pengguna menggunakan `fmt.Scan(&bilangan)`.

### 3. Inisialisasi Variabel:

- **saatIni** diinisialisasi dengan nilai bilangan.
- **batasAtas** diinisialisasi dengan nilai `math.Ceil(bilangan)` untuk membulatkan nilai ke atas.

### 4. Perulangan for:

- Perulangan dijalankan dengan kondisi `selesai = false; !selesai;`
- Di dalam perulangan:
  - `fmt.Printf("%.1f\n", saatIni);` Mencetak nilai `saatIni` dengan satu angka desimal.
  - `saatIni += 0.1;` Menambah 0.1 ke nilai `saatIni`.
  - `saatIni = math.Round(saatIni*10) / 10;` Membulatkan `saatIni` ke satu angka desimal.
  - `selesai = saatIni > batasAtas;` Menghentikan perulangan jika `saatIni` melebihi `batasAtas`.

## Contoh Penggunaan:

Misalkan pengguna menginputkan nilai 0.2, maka program akan menghasilkan output berikut:

0.2

0.3

0.4

0.5

**0.6**

**0.7**

**0.8**

**0.9**

**1.0**

**Penjelasan:**

- **Iterasi 1: saatIni = 0.2, batasAtas = 1.0**
- **Iterasi 2: saatIni = 0.3, batasAtas = 1.0**
- **Iterasi 3: saatIni = 0.4, batasAtas = 1.0**
- **dan seterusnya hingga saatIni mencapai atau melebihi batasAtas.**



## SOAL LATIHAN

### Statement perulangan

#### 3. Latihan3

#### Source Code:

```
package main

import "fmt"

func main() {
    var target, donasi1, donasi2, donasi3, donasi4 int
    var total int
    var jumlahDonatur int
    var selesai bool

    fmt.Scan(&target)

    jumlahDonatur = 0
    total = 0

    for selesai = false; !selesai; {
        jumlahDonatur++

        if jumlahDonatur == 1 {
            fmt.Scan(&donasi1)
            total += donasi1
        } else if jumlahDonatur == 2 {
            fmt.Scan(&donasi2)
            total += donasi2
        } else if jumlahDonatur == 3 {
            fmt.Scan(&donasi3)
            total += donasi3
        } else if jumlahDonatur == 4 {
            fmt.Scan(&donasi4)
            total += donasi4
        }

        selesai = total >= target
    }

    total = 0

    if jumlahDonatur >= 1 {
        total += donasi1
    }
}
```

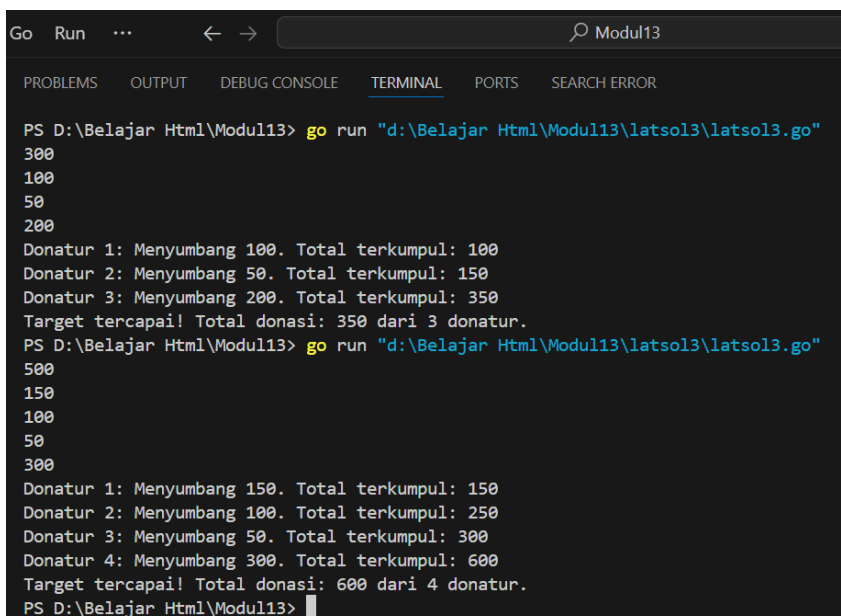
```

        fmt.Printf("Donatur 1: Menyumbang %d. Total terkumpul: %d\n", donasi1, total)
    }
    if jumlahDonatur >= 2 {
        total += donasi2
        fmt.Printf("Donatur 2: Menyumbang %d. Total terkumpul: %d\n", donasi2, total)
    }
    if jumlahDonatur >= 3 {
        total += donasi3
        fmt.Printf("Donatur 3: Menyumbang %d. Total terkumpul: %d\n", donasi3, total)
    }
    if jumlahDonatur >= 4 {
        total += donasi4
        fmt.Printf("Donatur 4: Menyumbang %d. Total terkumpul: %d\n", donasi4, total)
    }

    fmt.Printf("Target tercapai! Total donasi: %d dari %d donatur.\n", total,
jumlahDonatur)
}

```

Output:



```

Go Run ... < -> Modul13
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR

PS D:\Belajar Html\Modul13> go run "d:\Belajar Html\Modul13\latsol3\latsol3.go"
300
100
50
200
Donatur 1: Menyumbang 100. Total terkumpul: 100
Donatur 2: Menyumbang 50. Total terkumpul: 150
Donatur 3: Menyumbang 200. Total terkumpul: 350
Target tercapai! Total donasi: 350 dari 3 donatur.
PS D:\Belajar Html\Modul13> go run "d:\Belajar Html\Modul13\latsol3\latsol3.go"
500
150
100
50
300
Donatur 1: Menyumbang 150. Total terkumpul: 150
Donatur 2: Menyumbang 100. Total terkumpul: 250
Donatur 3: Menyumbang 50. Total terkumpul: 300
Donatur 4: Menyumbang 300. Total terkumpul: 600
Target tercapai! Total donasi: 600 dari 4 donatur.
PS D:\Belajar Html\Modul13>

```

## **Deskripsi Program:**

### **1. Deklarasi Variabel:**

- **target:** Variabel untuk menyimpan target jumlah donasi yang ingin dicapai.
- **donasi1, donasi2, donasi3, donasi4:** Variabel untuk menyimpan jumlah donasi dari masing-masing donatur.
- **total:** Variabel untuk menyimpan total donasi yang terkumpul.
- **jumlahDonatur:** Variabel untuk menghitung jumlah donatur yang telah memberikan donasi.
- **selesai:** Variabel boolean yang digunakan untuk mengontrol perulangan.

### **2. Membaca Input Target:**

- Program membaca nilai target yang diinputkan oleh pengguna menggunakan `fmt.Scan(&target)`.

### **3. Inisialisasi Variabel:**

- **jumlahDonatur** diinisialisasi dengan nilai 0.
- **total** diinisialisasi dengan nilai 0.

### **4. Perulangan for:**

- Perulangan dijalankan selama **selesai** bernilai false.
- Di dalam perulangan:
  - **jumlahDonatur** ditambah 1 pada setiap iterasi.
  - Program membaca input donasi berdasarkan jumlah donatur dan menambahkannya ke **total**.
  - Program memeriksa apakah **total** telah mencapai atau melebihi target. Jika ya, **selesai** diatur ke true.

### **5. Mencetak Hasil Donasi:**

- Setelah perulangan selesai, program mencetak jumlah donasi dari masing-masing donatur dan total donasi yang terkumpul menggunakan `fmt.Printf`.

## **Contoh Penggunaan:**

Misalkan pengguna menginputkan nilai berikut:

- Target donasi: 100
- Donasi donatur 1: 30

- **Donasi donatur 2: 40**
- **Donasi donatur 3: 20**
- **Donasi donatur 4: 15**

**Maka program akan menghasilkan output berikut:**

**Donatur 1: Menyumbang 30. Total terkumpul: 30**

**Donatur 2: Menyumbang 40. Total terkumpul: 70**

**Donatur 3: Menyumbang 20. Total terkumpul: 90**

**Donatur 4: Menyumbang 15. Total terkumpul: 105**

**Target tercapai! Total donasi: 105 dari 4 donatur.**

***DAFTAR PUSTAKA***

***MODUL PRAKTIKUM 13 - REPEAT-UNTIL***