LAPORAN PRAKTIKUM ALGORITMA

DAN PEMROGRAMAN 1

MODUL 13

REPEAT-UNTIL

Disusun oleh:

RIZKY TABRIZ DEANOVA

109082500177

S1IF-13-07

Asisten Praktikum

Adithana Dharma Putra

Apri Pandu Wicaksono

PROGRAM STUDI S1 INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

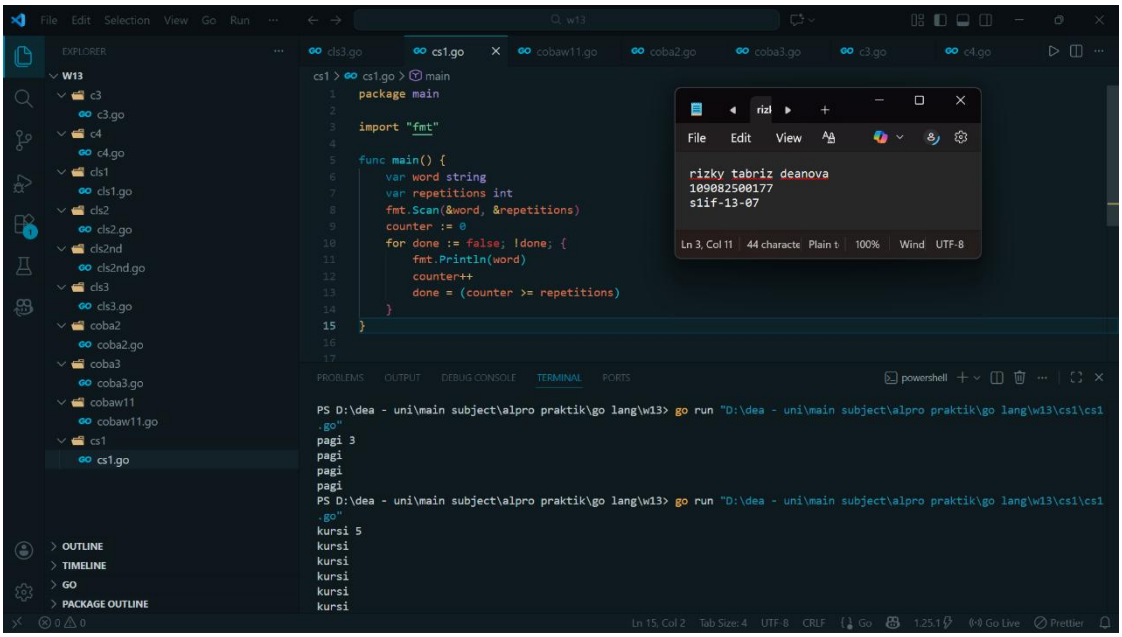
LATIHAN KELAS – GUIDED

1. Guided 1

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)
    }
}
```

Screenshoot program



Deskripsi program

`package main` sebagai penanda bahwa program ini merupakan program *executable* atau program yang dapat dieksekusi (dijalankan).

`import ("fmt")` yang digunakan untuk memasukkan library `"fmt"` untuk dipakai menjalankan program nantinya.

`func main () {}` yang berperan sebagai penanda tempat program mulai berjalan, seluruh kode ataupun instruksi yang terdapat di dalam kurung `{}` setelah `func main ()` nantinya akan dieksekusi atau dijalankan secara sistematis (berurutan).
`var` dalam program adalah sebagai wadah utama untuk penyimpanan data sebelum nanti dibagi lagi sesuai tipe data.
`fmt.Scan` yang dalam bahasa pemrograman Go digunakan untuk menjeda hasil output sejenak guna membaca input user, yang kemudian disimpan di dalam variabel.
`fmt.Println` yang dalam bahasa pemrograman Go digunakan sebagai instruksi untuk mencetak atau menampilkan output di garis baru.
`for` berfungsi sebagai perulangan, yang merupakan proses mengulang-ulang eksekusi blok kode di dalam perintah `for` tanpa henti selama kondisi yang diperintahkan terpenuhi, dalam blok kode `for` juga biasanya terdapat variabel penanda kapan perulangan akan diberhentikan.

Yang dilakukan program di dalam gambar di atas adalah **Mencetak Input Kata Sebanyak Perulangan** yang mana terlihat melalui kode dari **baris 10 hingga baris 13**.

`for done := false; !done; {` → Deklarasi nilai atau kondisi variabel done sebagai false meriksa apakah kondisi *value* dalam variabel j lebih dari 1, jika lebih dari 1 atau ya, maka operasi dalam perulangan for akan dijalankan

`fmt.Println(word)` → Mencetak nilai j yang sekarang dilanjutkan dengan alfabet x. Permisalahan output: 10 x

`counter++` → Pembaruan *value* atau nilai variabel j setelah dilakukan operasi perulangan yang akan berulang hingg kondisi tidak sesuai dengan kondisi dalam perulangan

`done = (counter >= repetitions)` → Pembaruan *value* atau nilai variabel j setelah dilakukan operasi perulangan yang akan berulang hingg kondisi tidak sesuai dengan kondisi dalam perulangan

Runtutan Eksekusi:

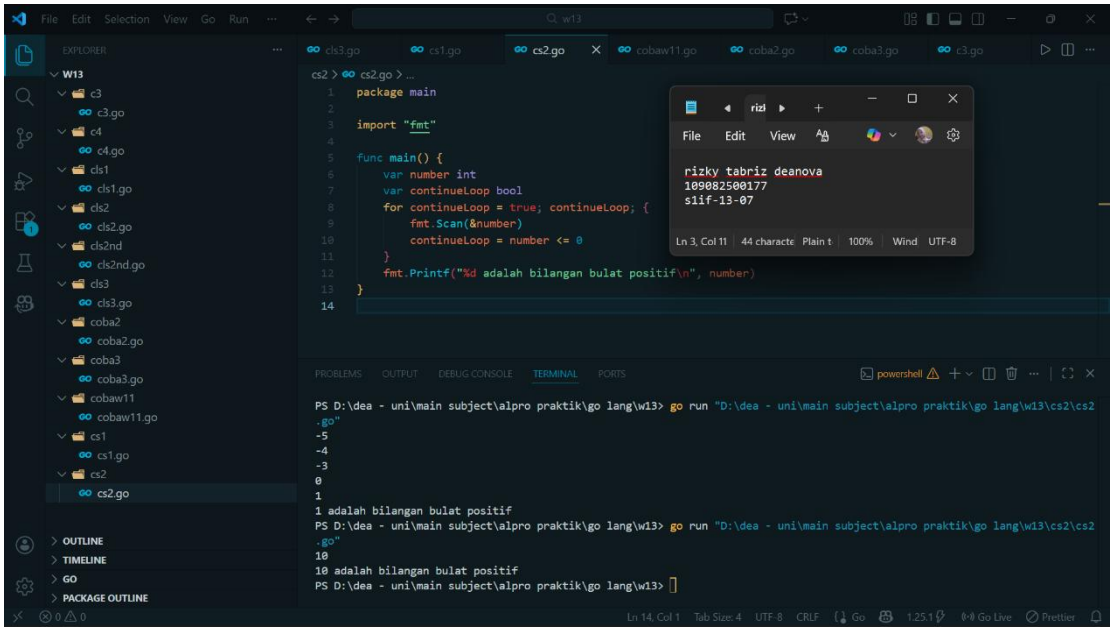
1. Menunggu user melakukan input kata serta angka yang akan dibaca lalu dimasukkan ke dalam variabel word dan repetitions secara berurut
2. Deklarasi nilai variabel counter sebanyak 0
3. Melakukan perulangan dengan mendeklarasikan variabel done yang bernilai false dan perulangan akan selalu dilakukan selagi nilai done false atau tidak true
4. Mencetak *value* atau nilai di dalam variabel word di garis baru
5. Selama pencetakan dilakukan, nilai variabel counter akan bertambah sebanyak 1
6. Memeriksa *value* atau nilai dalam variabel done dengan ketentuan nilai variabel counter lebih dari atau sama dengan nilai variabel repetitions. Jika sesuai dengan ketentuan, nilai variabel done akan berubah menjadi true. Dalam baris ini, jika tidak sesuai, nilai variabel done akan tetap menjadi false dan proses pencetekan akan diulang kembali.

2. Guided 2

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)
}
```

Screenshoot program



Deskripsi program

`package main` sebagai penanda bahwa program ini merupakan program *executable* atau program yang dapat dieksekusi (dijalankan).

`import ("fmt")` yang digunakan untuk memasukkan library "`fmt`" untuk dipakai menjalankan program nantinya.

`func main () {}` yang berperan sebagai penanda tempat program mulai berjalan, seluruh kode ataupun instruksi yang terdapat di dalam kurung `{}` setelah `func main ()` nantinya akan dieksekusi atau dijalankan secara sistematis (berurutan).

`var` dalam program adalah sebagai wadah utama untuk penyimpanan data sebelum nanti dibagi lagi sesuai tipe data.
`fmt.Scan` yang dalam bahasa pemrograman Go digunakan untuk menjeda hasil output sejenak guna membaca input user, yang kemudian disimpan di dalam variabel.
`fmt.Printf` bertugas untuk mencetak hasil input yang terformat
`for` berfungsi sebagai perulangan, yang merupakan proses mengulang-ulang eksekusi blok kode di dalam perintah `for` tanpa henti selama kondisi yang diperintahkan terpenuhi, dalam blok kode `for` juga biasanya terdapat variabel penanda kapan perulangan akan diberhentikan.

Yang dilakukan program di dalam gambar di atas adalah **Memeriksa Input Bilangan Positif Atau Tidak** yang mana terlihat melalui kode dari **baris 8 hingga baris 10**.

`for continueLoop = true; continueLoop; {` → Memulai perulangan dengan deklarasi nilai kondisi variabel continueLoop sebagai true dan akan terus berulang selama nilai kondisi variabel continueLoop adalah true

`fmt.Scan(&number)` → Menunggu input baru dari user dan memasukkan input terbaru sebagai nilai terbaru dalam variabel number

`continueLoop = number <= 0` → Memeriksa kondisi variabel continueLoop dengan ketentuan nilai variabel number adalah kurang dari atau sama dengan 0, bila memenuhi maka perulangan akan tetap diulang kembali dan nilai variabel continueLoop akan tetap mejadi true, jika tidak sesuai maka nilai variabel continueLoop akan berubah menjadi false dan nilai false ini akan menghentikan perulangan

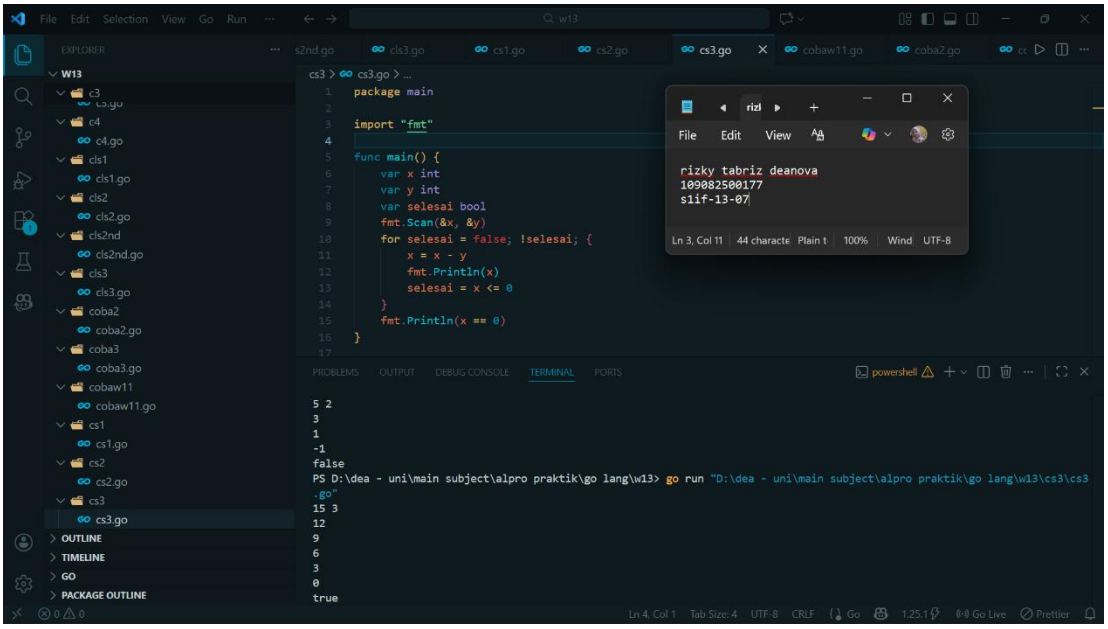
Runtutan Eksekusi:
1. Melakukan perulangan dengan deklarasi nilai variabel continueLoop true dan perulangan akan terus dilakukan selama nilai kondisi variabel continueLoop adalah true
2. Menunggu user melakukan input angka (integer) yang akan dimasukkan ke dalam variabel number
3. Memeriksa kondisi variabel continueLoop dengan ketentuan nilai variabel number adalah kurang dari atau sama dengan 0, bila memenuhi maka perulangan akan tetap diulang kembali dan nilai variabel continueLoop akan tetap mejadi true, jika tidak sesuai maka nilai variabel continueLoop akan berubah menjadi false dan nilai false ini akan menghentikan perulangan
4. Jika nilai variabel continueLoop berubah menjadi false, akan dilanjutkan ke baris *print out output*
5. Mencetak nilai terformat integer dari variabel number yang merupakan bilangan lebih dari 0 dengan keterangan lanjutan “adalah bilangan bulat positif”

3. Guided 3

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)
}
```

Screenshoot program



Deskripsi program

`package main` sebagai penanda bahwa program ini merupakan program *executable* atau program yang dapat dieksekusi (dijalankan).

`import ("fmt")` yang digunakan untuk memasukkan library `"fmt"` untuk dipakai menjalankan program nantinya.

`func main () {}` yang berperan sebagai penanda tempat program mulai berjalan, seluruh kode ataupun instruksi yang terdapat di dalam kurung `{}` setelah `func main ()` nantinya akan dieksekusi atau dijalankan secara sistematis (berurutan).
`var` dalam program adalah sebagai wadah utama untuk penyimpanan data sebelum nanti dibagi lagi sesuai tipe data.
`fmt.Scan` yang dalam bahasa pemrograman Go digunakan untuk menjeda hasil output sejenak guna membaca input user, yang kemudian disimpan di dalam variabel.
`fmt.Println` yang dalam bahasa pemrograman Go digunakan sebagai instruksi untuk mencetak atau menampilkan output di garis baru.
`for` berfungsi sebagai perulangan, yang merupakan proses mengulang-ulang eksekusi blok kode di dalam perintah `for` tanpa henti selama kondisi yang diperintahkan terpenuhi, dalam blok kode `for` juga biasanya terdapat variabel penanda kapan perulangan akan diberhentikan.

Yang dilakukan program di dalam gambar di atas adalah **Memeriksa Nilai X Kelipatan Nilai Y Atau Tidak** yang mana terlihat melalui kode dari **baris 10 hingga baris 13**.

`for selesai = false; !selesai; {` → Memulai perulangan dengan memberikan nilai kondisi variabel selesai adalah false dan perulangan akan terus berlanjut hingga kondisi nilai dari variabel selesai merupakan nilai negasinya atau true

`x = x - y` → Melakuan operasi pengurangan antara variabel x dengan y dan hasilnya akan dimasukkan ke dalam variabel x sebagai nilai terbaru dari variabel x

`fmt.Println(x)` → Mencetak nilai variabel x di baris baru

`selesai = x <= 0s1 = s2` → Melakukan pemeriksaan kondisi variabel x kurang dari atau sama dengan 0 yang mana hasilnya berupa true atau false dan akan dimasukkan ke dalam variabel selesai sebagai nilai

Runtutan Eksekusi:
1. Menunggu user melakukan input integer ke dalam variabel x dan y
2. Memulai perulangan dengan deklarasi atau memasukkan false sebagai nilai dari variabel selesai yang mana perulangan ini akan dilakukan terus-menerus selama kondisi nilai variabel selesai adalah false dan akan berhenti saat kondisi nilai variabel selesai adalah negasinya atau true
3. Melakukan operasi pengurangan antara variabel x dan y yang mana hasilnya akan dimasukkan ke dalam variabel x sebagai nilai baru
4. Mencetak nilai dari variabel x dalam baris baru
5. Kondisi variabel selesai akan berubah menjadi true jika nilai variabel x adalah kurang dari atau sama dengan 0. Jika memenuhi kondisi di dalam ketentuan baris ini, maka nilai variabel selesai akan menjadi true dan perulangan akan selesai

- Mencetak nilai dari operasi pemeriksaan kondisi variabel x sama dengan 0 atau tidak. Jika sama dengan 0 maka true, jika tidak sama dengan 0 maka false

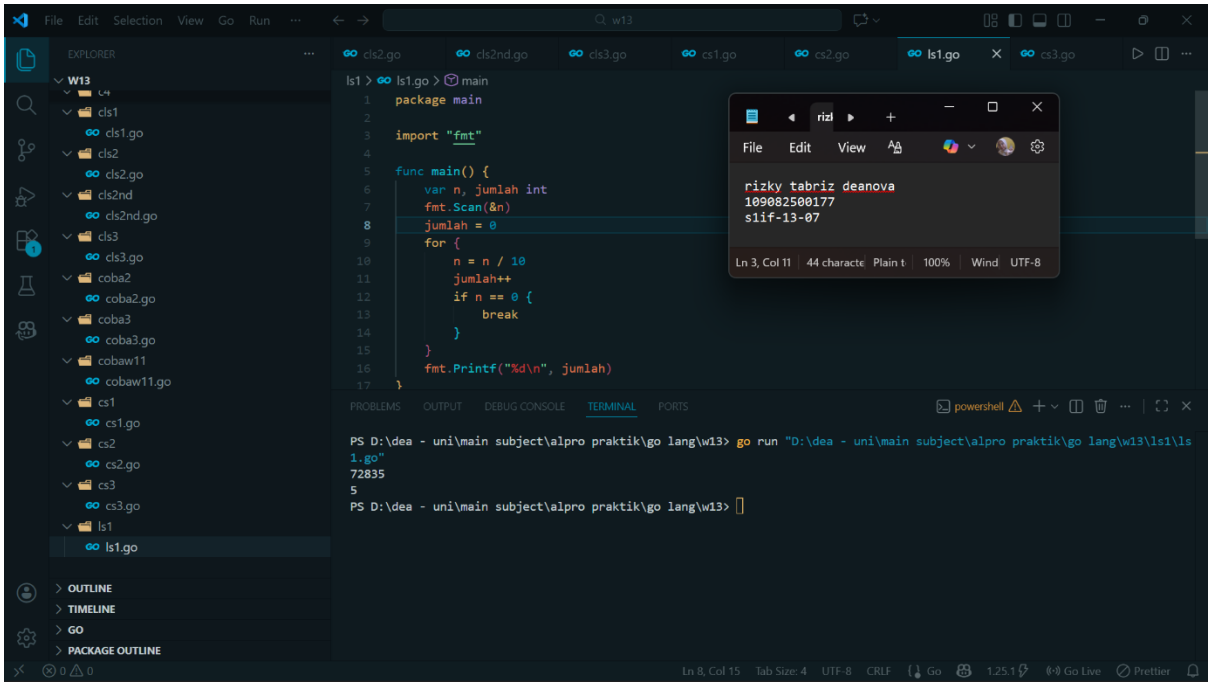
TUGAS

1. Tugas 1

Source code

```
package main
import "fmt"
func main() {
    var n, jumlah int
    fmt.Scan(&n)
    jumlah = 0
    for {
        n = n / 10
        jumlah++
        if n == 0 {
            break
        }
    }
    fmt.Printf("%d\n", jumlah)
}
```

Screenshoot program



Deskripsi program

`package main` sebagai penanda bahwa program ini merupakan program *executable* atau program yang dapat dieksekusi (dijalankan).

`import ("fmt")` yang digunakan untuk memasukkan library `"fmt"` untuk dipakai menjalankan program nantinya.

`func main () {}` yang berperan sebagai penanda tempat program mulai berjalan, seluruh kode ataupun instruksi yang terdapat di dalam kurung `{}` setelah `func main ()` nantinya akan dieksekusi atau dijalankan secara sistematis (berurutan).

`var` dalam program adalah sebagai wadah utama untuk penyimpanan data sebelum nanti dibagi lagi sesuai tipe data.

`fmt.Scan` yang dalam bahasa pemrograman Go digunakan untuk menjeda hasil output sejenak guna membaca input user, yang kemudian disimpan di dalam variabel.

`fmt.Printf` yang dalam bahasa pemrograman Go digunakan sebagai instruksi untuk mencetak atau menampilkan output yang terformat.

`for` berfungsi sebagai perulangan, yang merupakan proses mengulang-ulang eksekusi blok kode di dalam perintah `for` tanpa henti selama kondisi yang diperintahkan terpenuhi, dalam blok kode `for` juga biasanya terdapat variabel penanda kapan perulangan akan diberhentikan.

`if` berfungsi sebagai pengecekan kesesuaian kondisi suatu variabel dengan kondisi di dalam baris `if`.

`break` adalah *command* untuk menghentikan operasi `if` jika suatu variabel yang tercantum dalam operasi tersebut sesuai dengan kondisi di dalam operasi yang diperiksa.

Yang dilakukan program di dalam gambar di atas adalah **Mencetak Jumlah Bilangan Dalam Input** yang mana terlihat melalui kode dari **baris 9 hingga baris 16**.

`for {` → Dimulainya perulangan dengan operasi dan juga kondisi yang terdapat di dalamnya

`n = n / 10` → Melakukan operasi pembagian antara nilai dalam variabel n dengan 10 yang mana hasilnya akan dimasukkan ke dalam variabel n sebagai nilai terbaru

`jumlah++` → Menambah nilai variabel jumlah sebanyak 1 (increment)

`if n == 0 {` → Memeriksa kondisi apakah nilai variabel n sama dengan 0, jika sesuai maka akan dilanjutkan ke operasi selanjutnya

`break` → Menghentikan operasi for jikalau input yang diberikan user sesuai dengan kondisi di dalam operasi if

`fmt.Printf("%d \n", jumlah)` → Mencetak jumlah bilangan dalam input dengan output terformat, yaitu integer yang mana tersimpan di dalam variabel jumlah

Runtutan Eksekusi:

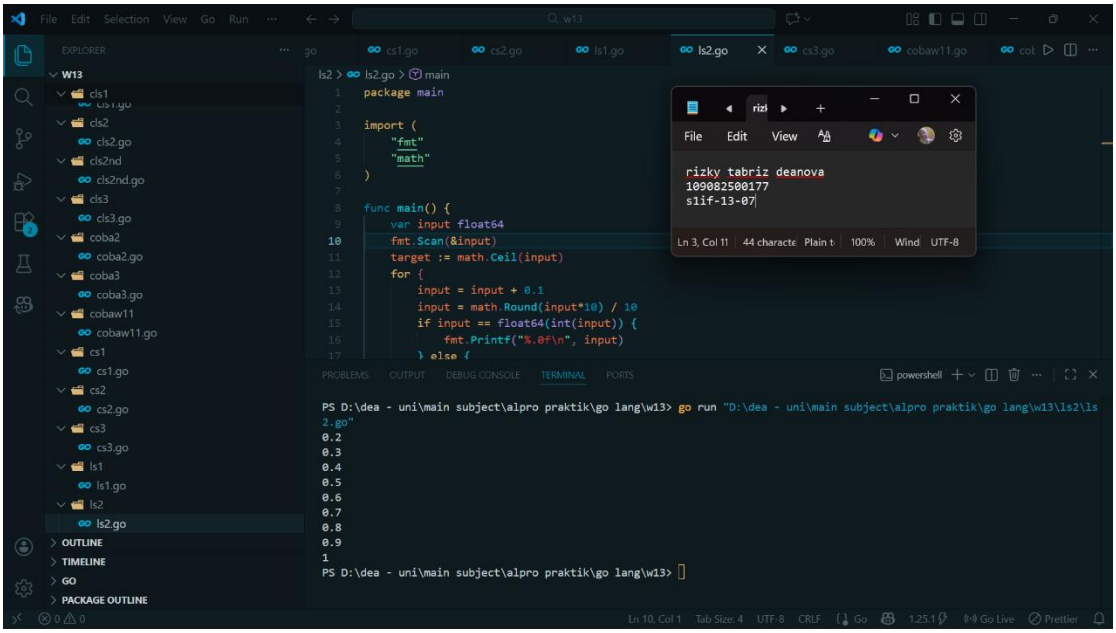
1. Memulai perulangan dengan melakukan pembagian antara variabel n dengan angka 10 yang mana hasilnya akan dimasukkan ke dalam variabel n sebagai nilai terbaru
2. Menambah nilai variabel jumlah sebanyak 1 selama perulangan dilakukan
3. Memeriksa kondisi variabel n sama dengan 0 atau tidak, jika sesuai maka akan dilanjutkan ke break
4. Dalam break, operasi perulangan dihentikan
5. Mencetak output terformat integer variabel jumlah yang merupakan hasil program atau jumlah bilangan input

2. Tugas 2

Source code

```
package main
import (
    "fmt"
    "math"
)
func main() {
    var input float64
    fmt.Scan(&input)
    target := math.Ceil(input)
    for {
        input = input + 0.1
        input = math.Round(input*10) / 10
        if input == float64(int(input)) {
            fmt.Printf("%.0f\n", input)
        } else {
            fmt.Printf("%.1f\n", input)
        }
        if input >= target {
            break
        }
    }
}
```

Screenshoot program



Deskripsi program

`package main` sebagai penanda bahwa program ini merupakan program *executable* atau program yang dapat dieksekusi (dijalankan).

`import ("fmt" "math")` yang digunakan untuk memasukkan library `"fmt"` serta `"math"` untuk dipakai menjalankan program nantinya, library `"math"` di sini berfungsi untuk melakukan operasi hitung matematika yang lebih kompleks atau memerlukan library tersebut untuk menjalankannya.

`func main () {}` yang berperan sebagai penanda tempat program mulai berjalan, seluruh kode ataupun instruksi yang terdapat di dalam kurung `{}` setelah `func main ()` nantinya akan dieksekusi atau dijalankan secara sistematis (berurutan).

`var` dalam program adalah sebagai wadah utama untuk penyimpanan data sebelum nanti dibagi lagi sesuai tipe data.

`fmt.Scan` yang dalam bahasa pemrograman Go digunakan untuk menjeda hasil output sejenak guna membaca input user, yang kemudian disimpan di dalam variabel.

`fmt.Printf` yang dalam bahasa pemrograman Go digunakan sebagai instruksi untuk mencetak atau menampilkan output yang terformat.

`for` berfungsi sebagai perulangan, yang merupakan proses mengulang-ulang eksekusi blok kode di dalam perintah `for` tanpa henti selama kondisi yang diperintahkan terpenuhi, dalam blok kode `for` juga biasanya terdapat variabel penanda kapan perulangan akan diberhentikan.

`break` dalam program digunakan untuk menghentikan program jika sesuai dengan kondisi tertentu yang berkaitan dengan break itu sendiri.

`math.Ceil` digunakan untuk membulatkan bilangan desimal (float) selalu ke atas menuju bilangan bulat terdekat.

`math.Round` digunakan untuk membulatkan bilangan desimal ke bilangan bulat terdekat.

Yang dilakukan program di dalam gambar di atas adalah **Perulangan Operasi Tambah Hingga Input Mencapai Bilangan Bulat Ke atas Terdekat** yang mana terlihat melalui kode dari **baris 11 hingga baris 21**.

- Program akan diulang secara terus-menerus hingga nilai variabel input lebih dari atau sama dengan nilai variabel target yang merupakan pembulatan ke atas dari nilai input

`target := math.Ceil(input)` → Deklarasi variabel target beserta nilainya yang merupakan pembulatan ke atas nilai input

`input = input + 0.1` → Menambah nilai input sebanyak 0.1 dan hasilnya akan dimasukkan ke dalam variabel input sebagai nilai terbaru dari variabel input

`input = math.Round(input*10) / 10` → Melakukan pembulatan nilai hasil perkalian input dengan angka 10 ke bilangan bulat terdekat yang kemudian dibagi 10 untuk tetap menjaga nilai sebagai 1 desimal, kemudian hasilnya akan dimasukkan ke dalam variabel input sebagai nilai variabel input terbaru

`if input == float64(int(input)) {` → Memeriksa kondisi nilai input apakah sama dengan bilangan bulat atau tidak, jika sesuai maka program selanjutnya akan dijalankan

`fmt.Printf("%.0f\n", input)` → Mencetak nilai input jika merupakan bilangan bulat, tetapi membuang .0 di belakangnya

`fmt.Printf("%.1f\n", input)` → Mencetak nilai variabel input dengan hanya 1 digit di belakang koma

`if input >= target {` → Memeriksa kondisi nilai input lebih dari atau sama dengan nilai variabel target atau tidak, jika ya maka akan dilanjutkan ke baris program selanjutnya

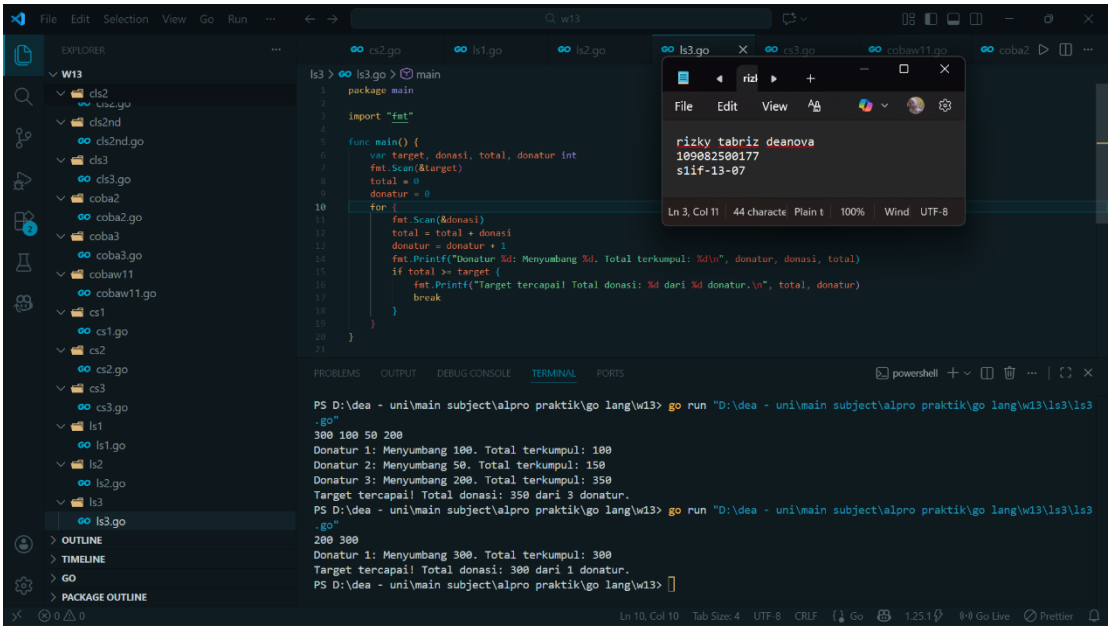
`break` → Menghentikan perogram

3. Tugas 3

Source code

```
package main
import "fmt"
func main() {
    var target, donasi, total, donatur int
    fmt.Scan(&target)
    total = 0
    donatur = 0
    for {
        fmt.Scan(&donasi)
        total = total + donasi
        donatur = donatur + 1
        fmt.Printf("Donatur %d: Menyumbang %d. Total
terkumpul: %d\n", donatur, donasi, total)
        if total >= target {
            fmt.Printf("Target tercapai! Total donasi: %d
dari %d donatur.\n", total, donatur)
            break
        }
    }
}
```

Screenshoot program



Deskripsi program

`package main` sebagai penanda bahwa program ini merupakan program *executable* atau program yang dapat dieksekusi (dijalankan).

`import ("fmt")` yang digunakan untuk memasukkan library "`fmt`" serta "`math`" untuk dipakai menjalankan program nantinya, library "`math`" di sini berfungsi untuk melakukan operasi hitung matematika yang lebih kompleks atau memerlukan library tersebut untuk menjalankannya.

`func main () {}` yang berperan sebagai penanda tempat program mulai berjalan, seluruh kode ataupun instruksi yang terdapat di dalam kurung `{}` setelah `func main ()` nantinya akan dieksekusi atau dijalankan secara sistematis (berurutan).
`var` dalam program adalah sebagai wadah utama untuk penyimpanan data sebelum nanti dibagi lagi sesuai tipe data.
`fmt.Scan` yang dalam bahasa pemrograman Go digunakan untuk menjeda hasil output sejenak guna membaca input user, yang kemudian disimpan di dalam variabel.
`fmt.Printf` yang dalam bahasa pemrograman Go digunakan sebagai instruksi untuk mencetak atau menampilkan output di garis baru.
`for` berfungsi sebagai perulangan, yang merupakan proses mengulang-ulang eksekusi blok kode di dalam perintah `for` tanpa henti selama kondisi yang diperintahkan terpenuhi, dalam blok kode `for` juga biasanya terdapat variabel penanda kapan perulangan akan diberhentikan.
`break` dalam program digunakan untuk menghentikan program jika sesuai dengan kondisi tertentu yang berkaitan dengan break itu sendiri.

Yang dilakukan program di dalam gambar di atas adalah **Mencetak Hasil Penjumlahan Donasi Serta Donatur** yang mana terlihat melalui kode dari **baris 10 hingga baris 17**.

- Perulangan akan terus dijalankan sampai jumlah donasi mencapai target

`for` → Memulai perulangan

`fmt.Scan(&donasi)` → Memasukkan input user ke dalam variabel donasi

`total = total + donasi` → Menjumlahkan nilai variabel total dengan donasi dan hasilnya akan dimasukkan ke dalam variabel total sebagai nilai terbaru

`donatur = donatur + 1` → Menambah nilai variabel donatur sebanyak 1 selama program terulang

`fmt.Printf("Donatur %d: Menyumbang %d. Total terkumpul: %d\n", donatur, donasi, total)` → Mencetak "Donatur (nilai donatur): Menyumbang (nilai donasi). Total terkumpul: (nilai total)" dengan output terformat integer dari variabel donatur, donasi, dan total

`if total >= target {` → Memeriksa kondisi nilai variabel total lebih dari atau sama dengan target

`fmt.Printf("Target tercapai! Total donasi: %d dari %d donatur.\n", total, donatur)` → Mencetak "Target tercapai! Total donasi: (nilai total) dari (nilai donatur)" dengan output terformat integer dari variabel total dan donatur

`break` → Menghentikan program