# LAPORAN PRAKTIKUM

## Algoritma Pemrograman

MODUL 13

REPEAT-UNTIL


**Disusun Oleh:**

MUHAMAD RAFI ALFIANSYAH

109082500191

S1IF-13-04

PROGRAM STUDI S1 INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

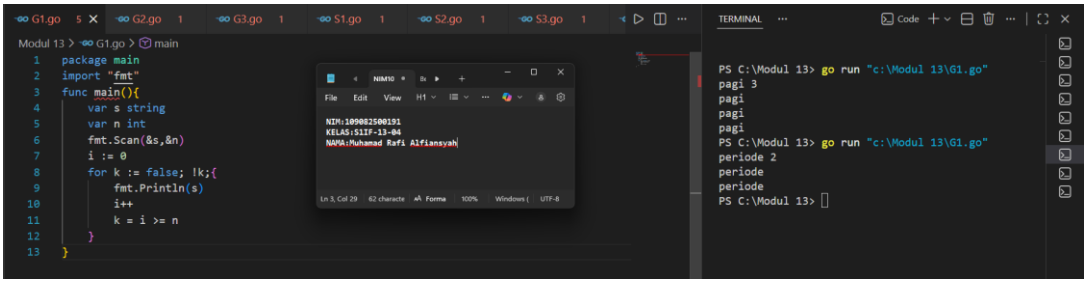
# LATIHAN KELAS – GUIDED

## 1. Guided 1

### Source Code

```
package main
import "fmt"
func main(){
	var s string
	var n int
	fmt.Scan(&s,&n)
	i := 0
	for k := false; !k;{
		fmt.Println(s)
		i++
		k = i >= n
	}
}
```

### Screenshoot Program



**Deskripsi Program**

Program ini digunakan untuk menampilkan sebuah kata sebanyak jumlah pengulangan yang diinginkan oleh pengguna. Program menerima dua buah masukan, yaitu sebuah kata dan sebuah bilangan bulat. Kata yang dimasukkan akan dicetak berulang kali sesuai dengan jumlah pengulangan yang diberikan oleh pengguna, dan proses akan berhenti ketika jumlah cetakan telah mencapai batas yang ditentukan.

Setelah menerima input, program menggunakan sebuah variabel penghitung bernama i yang diinisialisasi dengan nilai nol untuk menghitung berapa kali kata sudah dicetak. Selain itu, digunakan juga sebuah variabel boolean k yang berfungsi sebagai penanda kondisi berhenti pada perulangan. Proses pencetakan dilakukan di dalam struktur perulangan for yang dibuat menyerupai konsep repeat–until, sehingga perintah di dalam perulangan dijalankan terlebih dahulu sebelum kondisi penghentian diperiksa.

Pada setiap iterasi perulangan, program mencetak kata yang diinputkan menggunakan perintah fmt.Println(s). Setelah kata dicetak, nilai variabel i akan ditambah satu untuk menandakan bahwa satu kali pengulangan telah dilakukan. Selanjutnya, program memeriksa kondisi i >= n. Jika kondisi tersebut bernilai benar, maka variabel k akan diubah menjadi bernilai benar sehingga perulangan akan dihentikan. Jika kondisi belum terpenuhi, perulangan akan terus berjalan hingga jumlah pengulangan sesuai dengan input pengguna.

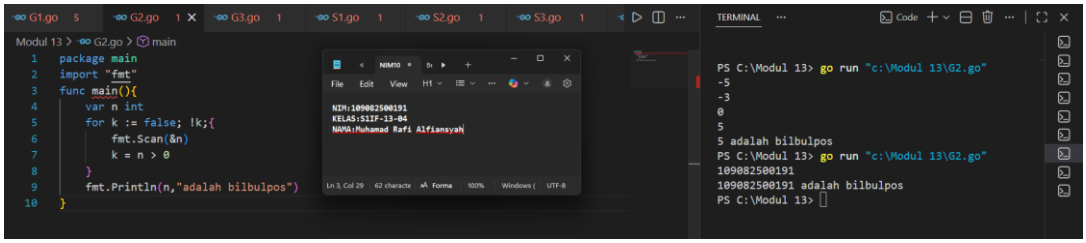
Dengan menggunakan cara ini, program dapat memastikan bahwa kata dicetak tepat sebanyak jumlah yang diminta oleh pengguna tanpa terjadi kelebihan atau kekurangan pengulangan. Struktur perulangan yang digunakan juga memastikan bahwa proses pencetakan minimal dijalankan satu kali, sehingga sesuai dengan konsep perulangan repeat–until. Program ini sederhana, mudah dipahami, dan telah memenuhi ketentuan soal yang diberikan.

## 2. Guided 2

### Source Code

```
package main
import "fmt"
func main(){
	var n int
	for k := false; !k;{
		fmt.Scan(&n)
		k = n > 0
	}
	fmt.Println(n,"adalah bilbulpos")
}
```

### Screenshoot Program




### Deskripsi Program

Program ini digunakan untuk meminta pengguna memasukkan sebuah bilangan bulat positif. Program akan terus meminta input dari pengguna selama bilangan yang dimasukkan belum memenuhi syarat sebagai bilangan bulat positif. Proses input akan dihentikan secara otomatis ketika pengguna berhasil memasukkan bilangan bulat yang lebih besar dari nol.

Pada awal program, dideklarasikan sebuah variabel n bertipe integer untuk menyimpan nilai input dari pengguna. Proses pengulangan dilakukan menggunakan struktur perulangan for yang disusun menyerupai konsep repeat–until. Dalam perulangan ini, digunakan sebuah variabel boolean k sebagai penanda kondisi berhenti. Selama nilai k masih bernilai salah, program akan terus meminta pengguna memasukkan bilangan.

Di dalam perulangan, program membaca input bilangan menggunakan perintah fmt.Scan(&n). Setelah itu, program memeriksa apakah nilai n lebih besar dari nol dengan kondisi n > 0. Jika kondisi tersebut terpenuhi, maka variabel k akan diubah menjadi bernilai benar sehingga perulangan berhenti. Jika nilai yang dimasukkan masih nol atau bilangan negatif, maka perulangan akan terus berjalan dan pengguna diminta memasukkan ulang bilangan hingga memenuhi syarat.

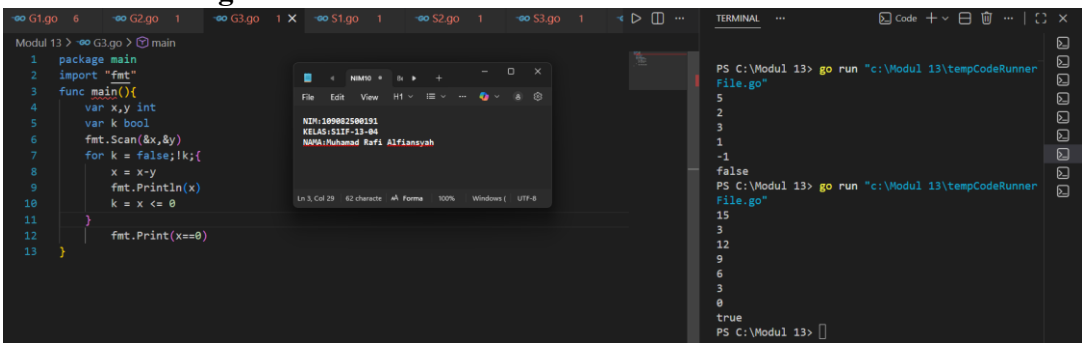
Setelah perulangan berhenti, program menampilkan satu baris keluaran yang menyatakan bahwa bilangan yang terakhir dimasukkan merupakan bilangan bulat positif. Dengan alur tersebut, program memastikan bahwa nilai yang diproses dan ditampilkan sebagai keluaran benar-benar merupakan bilangan bulat positif sesuai dengan ketentuan soal.

## 3. Guided 3

### Source Code

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

### ScreenshootProgram



**Deskripsi Program**

Program ini digunakan untuk melakukan pengecekan apakah suatu bilangan bulat positif merupakan kelipatan dari bilangan bulat positif lainnya dengan menggunakan metode pengurangan berulang. Program menerima dua buah input, yaitu bilangan bulat positif x dan y. Nilai x akan dikurangi secara terus-menerus dengan nilai y hingga memenuhi kondisi tertentu, kemudian program akan menentukan apakah x merupakan kelipatan dari y berdasarkan hasil akhir pengurangan tersebut.

Pada awal program, dideklarasikan variabel x dan y bertipe integer untuk menyimpan nilai masukan pengguna, serta sebuah variabel boolean k yang digunakan sebagai penanda kondisi berhenti perulangan. Setelah menerima input, program menjalankan struktur perulangan for yang disusun menyerupai konsep repeat–until, sehingga proses pengurangan dijalankan terlebih dahulu sebelum kondisi penghentian diperiksa.

Di dalam perulangan, nilai x dikurangi dengan y pada setiap iterasi, kemudian hasil pengurangan tersebut langsung ditampilkan ke layar menggunakan perintah fmt.Println(x). Setelah itu, program memeriksa kondisi x <= 0. Jika kondisi ini terpenuhi, maka variabel k diubah menjadi bernilai benar sehingga perulangan dihentikan. Proses ini menghasilkan deretan nilai pengurangan yang menunjukkan kelipatan atau sisa dari bilangan tersebut.

Setelah perulangan selesai, program menampilkan hasil akhir berupa nilai boolean dari ekspresi x == 0. Jika hasilnya bernilai true, maka bilangan awal x merupakan kelipatan dari y. Sebaliknya, jika bernilai false, maka bilangan tersebut bukan kelipatan dari y. Dengan demikian, program ini dapat menentukan hubungan kelipatan antara dua bilangan bulat positif tanpa menggunakan operator modulo.

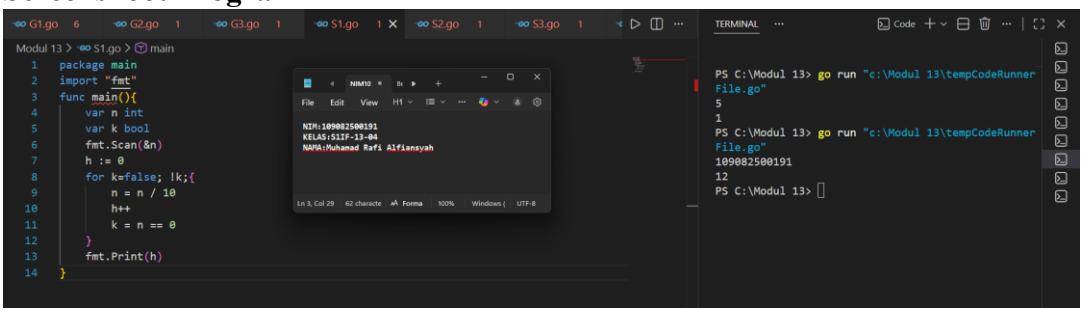
# TUGAS

## 1. Tugas 1

### Source Code

```
package main
import "fmt"
func main(){
	var n int
	var k bool
	fmt.Scan(&n)
	h := 0
	for k=false; !k;{
		n = n / 10
		h++
		k = n == 0
	}
	fmt.Print(h)
}
```

### Screenshoot Program




### Deskripsi Program

Program ini digunakan untuk menghitung banyaknya digit dari sebuah bilangan bulat positif yang dimasukkan oleh pengguna. Program akan menerima satu buah input berupa bilangan bulat positif, kemudian menghitung jumlah digit dari bilangan tersebut dan menampilkannya sebagai keluaran.

Pada awal program, dideklarasikan sebuah variabel n bertipe integer untuk menyimpan nilai bilangan yang diinputkan oleh pengguna. Selain itu, digunakan variabel h sebagai penghitung jumlah digit, yang diinisialisasi dengan nilai nol.

Program juga menggunakan sebuah variabel boolean k yang berfungsi sebagai penanda untuk menghentikan proses perulangan.

Proses perhitungan jumlah digit dilakukan menggunakan struktur perulangan for yang disusun menyerupai konsep repeat–until. Di dalam perulangan, nilai n dibagi dengan 10 pada setiap iterasi menggunakan operasi pembagian bilangan bulat. Pembagian ini bertujuan untuk menghilangkan satu digit terakhir dari bilangan tersebut. Setiap kali proses pembagian dilakukan, nilai h akan ditambah satu sebagai penanda bahwa satu digit telah dihitung.

Setelah pembagian dilakukan, program memeriksa kondisi n == 0. Jika kondisi tersebut terpenuhi, maka variabel k diubah menjadi bernilai benar sehingga perulangan dihentikan. Hal ini menandakan bahwa seluruh digit pada bilangan telah habis dibagi dan jumlah digit sudah sepenuhnya dihitung.

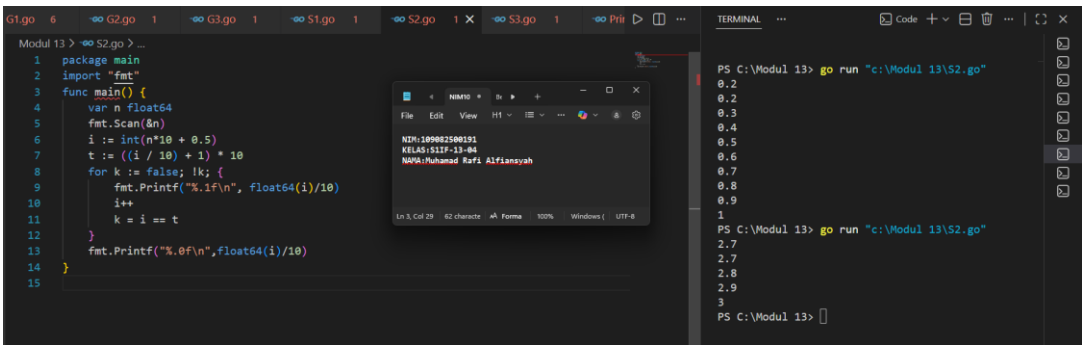
Setelah perulangan selesai, program menampilkan nilai h sebagai keluaran. Nilai ini menunjukkan banyaknya digit dari bilangan bulat positif yang dimasukkan oleh pengguna. Dengan alur tersebut, program dapat menghitung jumlah digit dengan benar untuk berbagai masukan bilangan bulat positif sesuai dengan ketentuan soal.

## 2. Tugas 2

### Source Code

```
package main
import "fmt"
func main() {
	var n float64
	fmt.Scan(&n)
	i := int(n*10 + 0.5)
	t := ((i / 10) + 1) * 10
	for k := false; !k; {
		fmt.Printf("%.1f\n", float64(i)/10)
		i++
		k = i == t
	}
	fmt.Printf("%.0f\n",float64(i)/10)
}
```

### Screenshoot Program



**Deskripsi Program**

Program ini digunakan untuk mendapatkan bilangan bulat optimal dari sebuah bilangan desimal yang dimasukkan oleh pengguna dengan cara melakukan penjumlahan secara bertahap hingga mencapai pembulatan ke atas dari bilangan tersebut. Program menerima satu buah masukan berupa bilangan desimal, kemudian menampilkan hasil penjumlahan setiap perulangan sampai nilai tersebut mencapai bilangan bulat terdekat ke atas.

Pada awal program, variabel n bertipe float64 digunakan untuk menyimpan nilai bilangan desimal hasil input pengguna. Untuk menghindari kesalahan presisi pada bilangan desimal, nilai n terlebih dahulu diubah ke dalam bentuk bilangan bulat dengan skala sepuluh melalui perhitungan i := int(n*10 + 0.5). Cara ini dilakukan agar proses penambahan dapat dikontrol menggunakan bilangan bulat sehingga lebih aman dan akurat. Selanjutnya, ditentukan nilai batas t yang merupakan bilangan bulat hasil pembulatan ke atas dari input pengguna, juga dalam skala sepuluh.

Proses perulangan dilakukan menggunakan struktur for yang disusun menyerupai konsep repeat–until. Di dalam perulangan, nilai i ditampilkan kembali dalam bentuk desimal dengan membaginya dengan 10 dan dicetak menggunakan format satu angka di belakang koma. Setelah itu, nilai i dinaikkan satu per satu yang merepresentasikan penambahan sebesar 0,1 pada nilai desimal aslinya. Perulangan akan terus berjalan hingga nilai i sama dengan batas t, yang menandakan bahwa bilangan telah mencapai pembulatan ke atas.

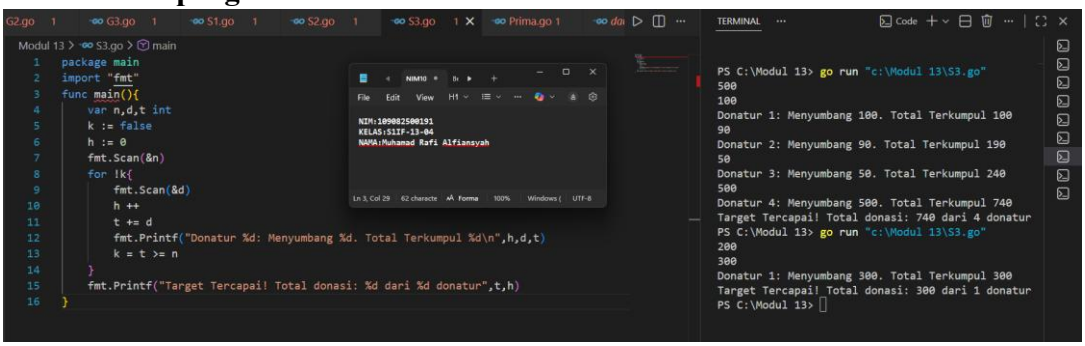
Setelah perulangan selesai, program mencetak satu baris terakhir berupa bilangan bulat hasil pembulatan ke atas dari bilangan desimal yang diinputkan. Dengan alur tersebut, program tidak hanya menampilkan hasil akhir pembulatan, tetapi juga menampilkan seluruh proses penjumlahan secara bertahap sesuai dengan ketentuan soal. Program ini memastikan hasil yang konsisten dan bebas dari kesalahan perhitungan akibat keterbatasan presisi bilangan desimal.

## 3. Tugas 3

### Source code

```
package main
import "fmt"
func main(){
	var n,d,t int
	k := false
	h := 0
	fmt.Scan(&n)
	for !k{
		fmt.Scan(&d)
		h ++
		t += d
		fmt.Printf("Donatur %d: Menyumbang %d. Total
Terkumpul %d\n",h,d,t)
		k = t >= n
	}
	fmt.Printf("Target Tercapai! Total donasi: %d dari %d
donatur",t,h)
}
```

### Screenshoot program



**Deskripsi Program**

Program ini digunakan untuk mensimulasikan proses pengumpulan donasi hingga mencapai target tertentu. Program menerima sebuah bilangan bulat sebagai target donasi, kemudian secara berulang meminta input berupa jumlah sumbangan dari setiap donatur. Proses akan terus berjalan sampai total donasi yang terkumpul telah mencapai atau melebihi target yang ditentukan.

Pada awal program, dideklarasikan beberapa variabel bertipe integer, yaitu n untuk menyimpan target donasi, d untuk menyimpan jumlah donasi dari setiap donatur, t untuk menyimpan total donasi yang telah terkumpul, dan h untuk menghitung jumlah donatur yang telah menyumbang. Selain itu, digunakan variabel boolean k sebagai penanda kondisi berhenti pada perulangan. Nilai target donasi dibaca terlebih dahulu dari input pengguna menggunakan fmt.Scan(&n).

Proses pengumpulan donasi dilakukan menggunakan struktur perulangan for yang disusun menyerupai konsep repeat–until. Selama kondisi berhenti belum terpenuhi, program akan meminta input jumlah donasi dari seorang donatur. Setiap kali input diterima, nilai penghitung donatur h akan bertambah satu, dan jumlah donasi tersebut akan dijumlahkan ke total donasi t. Setelah itu, program menampilkan informasi donatur keberapa, jumlah sumbangan yang diberikan, serta total donasi yang telah terkumpul hingga saat itu.

Setelah menampilkan informasi donasi, program memeriksa kondisi t >= n. Jika total donasi sudah mencapai atau melebihi target, maka variabel k diubah menjadi bernilai benar sehingga perulangan dihentikan. Hal ini menandakan bahwa target donasi telah tercapai dan tidak diperlukan input tambahan.

Setelah perulangan selesai, program menampilkan pesan akhir yang menyatakan bahwa target donasi telah tercapai, beserta informasi total donasi yang terkumpul dan jumlah donatur yang berpartisipasi. Dengan alur tersebut, program dapat memantau proses pengumpulan donasi secara bertahap dan memastikan bahwa proses berhenti tepat ketika target donasi tercapai.