

**LAPORAN PRAKTIKUM ALGORITMA  
DAN PEMROGRAMAN 1**

**MODUL 10**

**ELSE-IF**



**Disusun oleh:**

**NAUFAL BINTANG PRATAMA**

**109082500096**

**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

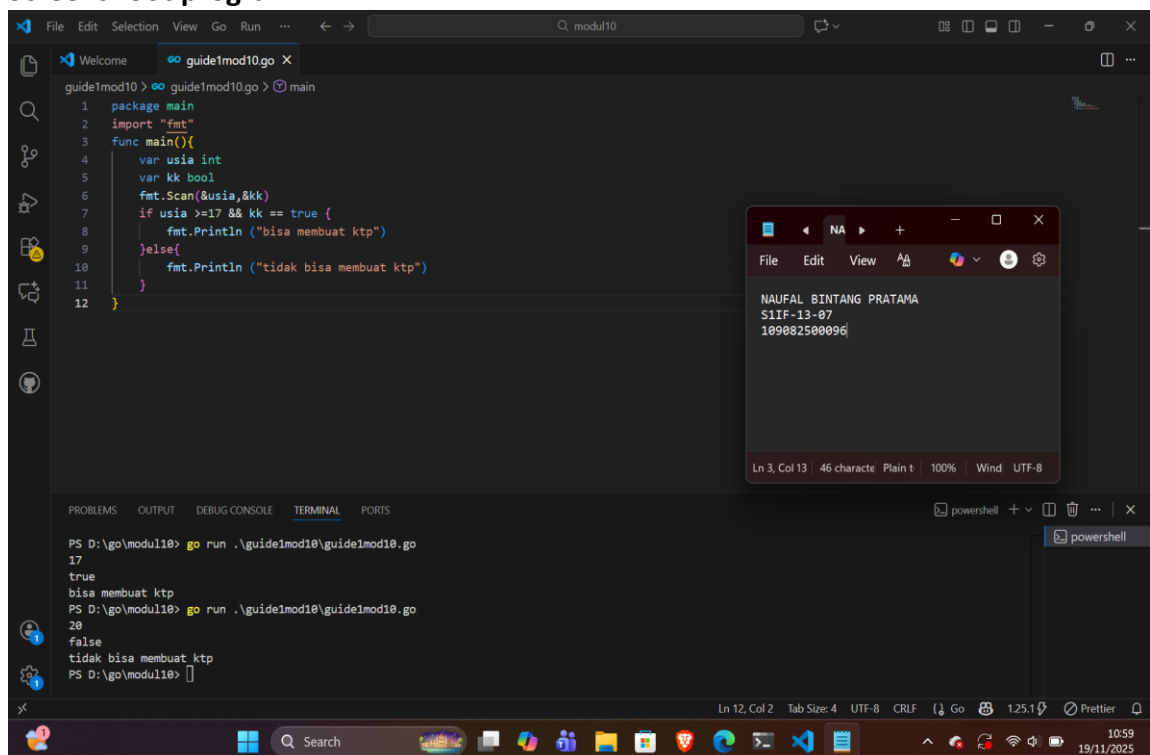
import "fmt"

func main(){
    var usia int
    var kk bool

    fmt.Scan(&usia,&kk)

    if usia >=17 && kk == true {
        fmt.Println ("bisa membuat ktp")
    }else{
        fmt.Println ("tidak bisa membuat ktp")
    }
}
```

#### Screenshoot program



#### Deskripsi program

Program ini digunakan untuk menentukan apakah seseorang **bisa membuat KTP** berdasarkan dua informasi yang dimasukkan pengguna, yaitu **usia** dan **status kepemilikan Kartu Keluarga (KK)**. Pengguna terlebih dahulu memasukkan angka usia dan nilai boolean (true atau false) untuk menunjukkan apakah mereka memiliki KK.

Setelah data dimasukkan, program membaca kedua input tersebut menggunakan `fmt.Scan`. Selanjutnya, program melakukan pengecekan menggunakan kondisi `if`. Seseorang dinyatakan **bisa membuat KTP** jika memenuhi dua syarat: berusia **minimal 17 tahun** dan memiliki **KK**. Jika kedua syarat terpenuhi, program menampilkan pesan *"bisa membuat ktp"*. Namun, jika salah satu syarat tidak terpenuhi, program akan menampilkan *"tidak bisa membuat ktp"*.

Secara sederhana, program ini berfungsi sebagai pengecekan kelayakan pembuatan KTP berdasarkan aturan dasar: cukup usia dan memiliki KK.

## 2. Guided 2

### Source Code

```
package main

import "fmt"

func main(){
    var x rune

    var huruf, vKecil, vBesar bool

    fmt.Scanf("%c",&x)

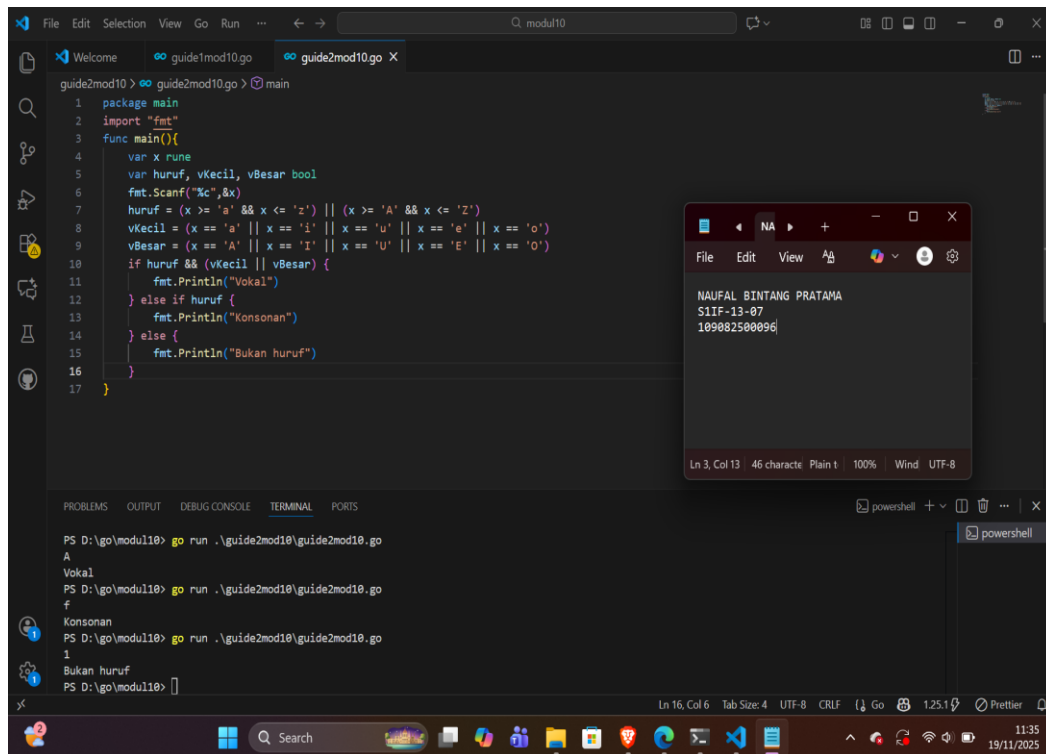
    huruf = (x >= 'a' && x <= 'z') || (x >= 'A' && x <= 'Z')

    vKecil = (x == 'a' || x == 'i' || x == 'u' || x == 'e'
    || x == 'o')

    vBesar = (x == 'A' || x == 'I' || x == 'U' || x == 'E'
    || x == 'O')

    if huruf && (vKecil || vBesar) {
        fmt.Println("Vokal")
    } else if huruf {
        fmt.Println("Konsonan")
    } else {
        fmt.Println("Bukan huruf")
    }
}
```

**Screenshoot program**



```
1 package main
2 import "fmt"
3 func main(){
4     var x rune
5     var huruf, vKecil, vBesar bool
6     fmt.Scanf("%c",&x)
7     huruf = (x >= 'a' && x <= 'z') || (x >= 'A' && x <= 'Z')
8     vKecil = (x == 'a' || x == 'i' || x == 'u' || x == 'e' || x == 'o')
9     vBesar = (x == 'A' || x == 'I' || x == 'U' || x == 'E' || x == 'O')
10    if huruf && (vKecil || vBesar) {
11        fmt.Println("Vokal")
12    } else if huruf {
13        fmt.Println("Konsonan")
14    } else {
15        fmt.Println("Bukan huruf")
16    }
17 }
```

```
PS D:\go\modul10> go run .\guide2mod10\guide2mod10.go
A
Vokal
PS D:\go\modul10> go run .\guide2mod10\guide2mod10.go
f
Konsonan
PS D:\go\modul10> go run .\guide2mod10\guide2mod10.go
1
Bukan huruf
PS D:\go\modul10> 
```

### Deskripsi program

Program ini digunakan untuk menentukan apakah sebuah karakter yang dimasukkan oleh pengguna merupakan **huruf vokal**, **huruf konsonan**, atau **bukan huruf**. Program meminta satu karakter input, lalu karakter tersebut disimpan dalam variabel bertipe rune.

Setelah menerima input, program melakukan tiga pengecekan utama. Pertama, program mengecek apakah karakter tersebut termasuk huruf alfabet, baik huruf kecil maupun huruf besar. Kedua, program memeriksa apakah karakter tersebut merupakan huruf vokal kecil (a, i, u, e, o). Ketiga, program juga memeriksa kemungkinan huruf vokal besar (A, I, U, E, O).

Hasil dari pengecekan tersebut digunakan dalam percabangan if-else. Jika karakter termasuk huruf dan juga termasuk vokal, program menampilkan "**Vokal**". Jika karakter adalah huruf tetapi bukan vokal, maka ditampilkan "**Konsonan**". Namun, jika karakter yang dimasukkan bukan huruf sama sekali, program menampilkan "**Bukan huruf**".

Dengan demikian, program ini berfungsi untuk mengelompokkan satu karakter yang dimasukkan pengguna berdasarkan jenisnya secara sederhana dan jelas.

### 3. Guided 3 Source Code

```
package main

import "fmt"

func main() {
```

```

var bilangan, d1, d2, d3, d4 int

var teks string

fmt.Print("Bilangan: ")

fmt.Scan(&bilangan)

d1 = bilangan / 1000

d2 = (bilangan / 100) % 10

d3 = (bilangan / 10) % 10

d4 = bilangan % 10

if d1 < d2 && d2 < d3 && d3 < d4 {

    teks = "terurut membesar"

}else if d1 > d2 && d2 > d3 && d3 > d4{

    teks = "terurut mengecil"

}else{

    teks = "tidak terurut"

}

fmt.Println("Digit pada bilangan", bilangan, teks)

}

```

## Screenshoot program

The screenshot shows a Visual Studio Code editor with a Go project. The source code for `guide3mod10.go` is displayed, which implements a program to check if the digits of a number are in ascending or descending order. The code uses `fmt` for input/output and calculates the thousands, hundreds, tens, and units digits. It then compares these digits to determine the order.

The terminal output shows three test cases:

```

PS D:\go\modul10> go run .\guide3mod10\guide3mod10.go
Bilangan: 2489
Digit pada bilangan 2489 terurut membesar
PS D:\go\modul10> go run .\guide3mod10\guide3mod10.go
Bilangan: 3861
Digit pada bilangan 3861 tidak terurut
PS D:\go\modul10> go run .\guide3mod10\guide3mod10.go
Bilangan: 8641
Digit pada bilangan 8641 terurut mengecil
PS D:\go\modul10>

```

A small window titled "NAUFAL BINTANG PRATAMA S1IF-13-07 109082500096" is also visible in the background.

## Deskripsi program

Program ini digunakan untuk menentukan apakah susunan digit pada sebuah bilangan empat digit berada dalam keadaan **terurut membesar**, **terurut mengecil**, atau **tidak terurut**. Pengguna terlebih dahulu diminta memasukkan sebuah bilangan, yang kemudian disimpan dalam variabel `bilangan`.

Program memecah bilangan tersebut menjadi empat digit terpisah, yaitu ribuan (`d1`), ratusan (`d2`), puluhan (`d3`), dan satuan (`d4`). Pemisahan ini dilakukan dengan operasi pembagian dan modulo. Setelah semua digit diperoleh, program melakukan pengecekan urutan menggunakan struktur `if-else`.

Jika digit pertama lebih kecil dari digit kedua, digit kedua lebih kecil dari digit ketiga, dan digit ketiga lebih kecil dari digit keempat, maka bilangan dinyatakan "**terurut membesar**". Sebaliknya, jika digit pertama lebih besar dari digit kedua, dan kondisi itu berlanjut hingga digit terakhir, maka bilangan dinyatakan "**terurut mengecil**". Selain kedua kondisi tersebut, program memberikan hasil "**tidak terurut**".

Terakhir, program menampilkan hasil berupa digit bilangan yang dimasukkan beserta keterangannya. Dengan demikian, program ini berfungsi sebagai pengecek pola urutan digit pada bilangan empat digit.

## TUGAS

### 1. Tugas 1

#### Source code

```
package main

import "fmt"

func main() { var berat int
    fmt.Print("Berat parcel (gram): ")
    fmt.Scan(&berat)

    kg := berat / 1000
    sisa := berat % 1000

    fmt.Println("Detail berat :", kg, "kg +", sisa, "gr")

    biaya := kg * 10000

    if kg >= 10 {
        fmt.Println("Detail biaya: Rp", kg*10000)
        fmt.Println("Total biaya :", biaya)
        return
    }

    if sisa < 500 {
```

```

    fmt.Println("Detail biaya: Rp", kg*10000, "+ Rp", sisa*15)
    biaya += sisa * 15
} else {
    fmt.Println("Detail biaya: Rp", kg*10000, "+ Rp", sisa*5)
    biaya += sisa * 5
}

fmt.Println("Total biaya :", biaya)

}

```

### Screenshoot program

```

func main() {
    5
    15     biaya := kg * 10000
    16
    17     if kg >= 10 {
    18         fmt.Println("Detail biaya: Rp", kg*10000)
    19         fmt.Println("Total biaya :", biaya)
    20         return
    21     }
    22
    23     if sisa < 500 {
    24         fmt.Println("Detail biaya: Rp", kg*10000, "+ Rp", sisa*15)
    25         biaya += sisa * 15
    26     } else {
    27         fmt.Println("Detail biaya: Rp", kg*10000, "+ Rp", sisa*5)
    28         biaya += sisa * 5
    29     }
    30
    31     fmt.Println("Total biaya :", biaya)
    32 }

```

```

PS D:\go\modul10> go run .\lat1mod10\lat1mod10.go
Berat parcel (gram): 8500
Detail berat : 8 kg + 500 gr
Detail biaya: Rp 80000 + Rp 2500
Total biaya : 82500
PS D:\go\modul10> go run .\lat1mod10\lat1mod10.go
Berat parcel (gram): 9250
Detail berat : 9 kg + 250 gr
Detail biaya: Rp 90000 + Rp 3750
Total biaya : 93750

```

### Deskripsi program

Program ini digunakan untuk menghitung biaya pengiriman parcel berdasarkan berat yang dimasukkan dalam satuan gram. Berat tersebut dipecah menjadi dua bagian, yaitu jumlah kilogram dan sisa gram, lalu keduanya ditampilkan sebagai detail berat.

Setelah itu, program menghitung biaya dasar dari kilogram dengan tarif Rp10.000 per kg. Jika berat parcel mencapai 10 kg atau lebih, total biaya langsung ditampilkan karena tarif tersebut sudah mencakup seluruh berat tanpa tambahan per gram.

Untuk berat di bawah 10 kg, program menghitung biaya tambahan berdasarkan sisa gram. Jika sisa gram kurang dari 500 gram, tarif yang digunakan adalah Rp15 per gram. Sedangkan jika sisa gram 500 gram atau lebih, tarif berubah menjadi Rp5 per gram. Biaya tambahan ini dijumlahkan dengan biaya dasarnya, lalu total biaya pengiriman ditampilkan kepada pengguna.

## 2. Tugas 2

### Source code

Ini adalah kode program yang sudah diperbaiki

```
package main

import "fmt"

func main() {
    var nam float64
    var nmk string

    fmt.Print("Masukkan nilai akhir mata kuliah (NAM): ")
    fmt.Scan(&nam)

    if nam >= 80 {
        nmk = "A"
    } else if nam >= 72.5 {
        nmk = "AB"
    } else if nam >= 65 {
        nmk = "B"
    } else if nam >= 57.5 {
        nmk = "BC"
    } else if nam >= 50 {
        nmk = "C"
    } else if nam >= 40 {
        nmk = "D"
    } else {
        nmk = "E"
    }
}
```



```

    }

    fmt.Println("Nilai mutu mata kuliah (NMK):", nmk)
}

```

### Screenshoot program

Dibawah ini merupakan kode program yg sudah uji coba menggunakan program diatas yang benar dan berkaitan dengan soal C

The screenshot shows a Go IDE with the following source code in `lat2mod10.go`:

```

func main() {
    8
    9     fmt.Print("Masukkan nilai akhir mata kuliah (NAM): ")
    10    fmt.Scan(&nam)
    11
    12    if nam >= 80 {
    13        nmk = "A"
    14    } else if nam >= 72.5 {
    15        nmk = "AB"
    16    } else if nam >= 65 {
    17        nmk = "B"
    18    } else if nam >= 57.5 {
    19        nmk = "BC"
    20    } else if nam >= 50 {
    21        nmk = "C"
    22    } else if nam >= 40 {
    23        nmk = "D"
    24    } else {
    25        nmk = "E"
    26    }
    27
    28    fmt.Println("Nilai mutu mata kuliah (NMK):", nmk)
}

```

The terminal output shows the program being run multiple times with different input values and the corresponding letter grade output:

```

PS D:\go\modul10> go run .\lat2mod10\lat2mod10.go
Masukkan nilai akhir mata kuliah (NAM): 93.5
Nilai mutu mata kuliah (NMK): A
PS D:\go\modul10> go run .\lat2mod10\lat2mod10.go
Masukkan nilai akhir mata kuliah (NAM): 70.6
Nilai mutu mata kuliah (NMK): B
PS D:\go\modul10> go run .\lat2mod10\lat2mod10.go
Masukkan nilai akhir mata kuliah (NAM): 49.5
Nilai mutu mata kuliah (NMK): D
PS D:\go\modul10>

```

### Deskripsi program

Program ini dibuat untuk menentukan nilai huruf (NMK) dari sebuah nilai akhir mata kuliah (NAM) yang dimasukkan oleh pengguna. Nilai tersebut disimpan dalam variabel bertipe *float64* bernama **nam**, kemudian dibaca melalui fungsi `fmt.Scan`. Setelah nilai diperoleh, program melakukan pengecekan secara berurutan menggunakan struktur *if-else if* untuk mencocokkan nilai yang diberikan dengan rentang standar penilaian.

Proses pengecekan dimulai dari nilai terbesar. Jika **nam**  $\geq 80$ , maka nilai huruf yang diberikan adalah "A". Apabila tidak memenuhi, program melanjutkan ke batas berikutnya, yaitu  $\geq 72.5$  untuk nilai "AB", lalu  $\geq 65$  untuk "B",  $\geq 57.5$  untuk "BC", dan  $\geq 50$  untuk "C". Jika nilainya berada pada rentang  $\geq 40$ , maka hasilnya "D", sedangkan nilai di bawah itu secara otomatis diberi kategori "E". Hasil akhir dari proses klasifikasi tersebut disimpan pada variabel **nmk**.

Setelah mendapatkan nilai huruf, program menampilkannya ke layar menggunakan `fmt.Println`. Dengan pengelompokan rentang seperti ini, program dapat mengubah nilai numerik menjadi nilai huruf sesuai aturan penilaian yang telah ditetapkan.

Jawaban soal A,B,C sebagai berikut:

### a. Jika nilai NAM = 80.1, apa keluaran programnya? Apakah sesuai spesifikasi?

Pada program asli, input 80.1 langsung dibandingkan dengan interval bernilai 0–10. Karena kondisi pertama yang dicek adalah `nam >= 7.5`, maka nilai 80.1 otomatis dianggap memenuhi syarat tersebut sehingga hasil yang ditampilkan adalah **"A"**.

Namun, hal ini **tidak sesuai spesifikasi tabel**, karena:

1. Nilai 80.1 seharusnya masih berada pada skala 0–100, bukan langsung diperlakukan sebagai nilai 7.5–8.0.
2. Program tidak melakukan normalisasi nilai, sehingga pembandingan interval menjadi tidak akurat.
3. Rentang untuk A pada tabel dimulai dari **7.6**, tetapi program memakai **7.5**, sehingga batas bawah tidak tepat.

Dengan demikian, output program kurang tepat dan tidak sepenuhnya mengikuti aturan penilaian yang diberikan.

### b. Kesalahan program dan alur yang seharusnya

Kesalahan yang ditemukan:

1. **Interval**                      **penilaian**                      **tidak**                      **sesuai**                      **tabel**  
Program menggunakan batas bawah yang berbeda dari standar (misal 7.5, padahal seharusnya 7.6).
2. **Setiap**                      **kondisi**                      **memakai**                      **if**                      **terpisah**  
Karena tidak memakai `else if`, beberapa kondisi bisa saja aktif sekaligus jika interval saling tumpang tindih.
3. **Tidak**                      **ada**                      **pembatas**                      **nilai**                      **maksimal**  
Nilai lebih dari 10 tetap dianggap valid.
4. **Tidak**                      **ada**                      **konversi**                      **dari**                      **skala**                      **100**                      **ke**                      **skala**                      **10**  
Padahal tabel skor memakai format 0–10.
5. **Tidak ada validasi nilai yang berada di bawah 0 atau di atas 100.**

## Alur program yang seharusnya:

1. Terima input nilai akhir dalam rentang 0–100.
2. Konversi nilai menjadi skala 0–10:  
 $skala = nilai / 10$ .
3. Bandingkan skala tersebut dengan interval tabel secara berurutan menggunakan `else if`.
4. Tentukan huruf mutu berdasarkan interval yang sesuai.
5. Tampilkan output akhir.

C. Perbaiki Program + Uji coba

Program yang sudah diperbaiki:

package main

```
import "fmt"
```

```
func main() {
```

```
    var nam float64
```

```
    var nmk string
```

```
    fmt.Print("Masukkan nilai akhir mata kuliah (NAM): ")
```

```
    fmt.Scan(&nam)
```

```
    if nam >= 80 {
```

```
        nmk = "A"
```

```
    } else if nam >= 72.5 {
```

```
        nmk = "AB"
```

```
    } else if nam >= 65 {
```

```
        nmk = "B"
```

```
    } else if nam >= 57.5 {
```

```
        nmk = "BC"
```

```
    } else if nam >= 50 {
```

```
        nmk = "C"
```

```
    } else if nam >= 40 {
```

```
        nmk = "D"
```

```
    } else {
```

```
        nmk = "E"
```

```
    }
```

```
    fmt.Println("Nilai mutu mata kuliah (NMK):", nmk)
```

```
}
```

Hasil uji coba:

Input	Output	Sesuai?
93.5	A	✓

70.5	C	✓
48.5	E	✓

### 3. Tugas 3

#### Source code

```
package main

import "fmt"

func main() {
    var b int
    fmt.Print("Bilangan: ")
    fmt.Scan(&b)

    fmt.Print("Faktor: ")

    jumlahFaktor := 0
    for i := 1; i <= b; i++ {
        if b%i == 0 {
            fmt.Print(i, " ")
            jumlahFaktor++
        }
    }

    fmt.Println()

    prime := (jumlahFaktor == 2)

    fmt.Print("Prima: ")
    if prime {
        fmt.Println("true")
    } else {
        fmt.Println("false")
    }
}
```

#### Screenshoot program

The screenshot shows a Go IDE with a file named `lat3mod10.go`. The code defines a `main` function that takes an integer `b` as input. It prints the input, then finds all factors of `b` by iterating from 1 to `b` and checking if `b % i == 0`. The factors are printed, and the total count is stored in `jumlahFaktor`. Finally, it checks if `jumlahFaktor == 2` to determine if `b` is a prime number and prints the result.

```
1 package main
2
3 import "fmt"
4
5 func main() {
6     var b int
7     fmt.Print("Bilangan: ")
8     fmt.Scan(&b)
9
10    fmt.Print("Faktor: ")
11
12    jumlahFaktor := 0
13    for i := 1; i <= b; i++ {
14        if b%i == 0 {
15            fmt.Print(i, " ")
16            jumlahFaktor++
17        }
18    }
19
20    fmt.Println()
21
22    prime := (jumlahFaktor == 2)
23
24    fmt.Println("Prima: ", prime)
25 }
```

The terminal output shows two test cases: `Bilangan: 7` resulting in `Prima: true`, and `Bilangan: 12` resulting in `Prima: false`.

## Deskripsi program

Program ini dibuat untuk menampilkan faktor-faktor dari suatu bilangan dan sekaligus menentukan apakah bilangan tersebut merupakan bilangan prima atau bukan. Nilai input disimpan dalam variabel **b** bertipe integer, kemudian dibaca menggunakan `fmt.Scan`.

Setelah itu, program mencetak semua faktor dari bilangan tersebut dengan menggunakan perulangan `for` dari 1 sampai **b**. Pada setiap iterasi, program mengecek apakah nilai **i** merupakan pembagi tepat dari **b** dengan kondisi `b % i == 0`. Jika benar, nilai tersebut dicetak sebagai faktor dan penghitung faktor (`jumlahFaktor`) ditambah satu.

Setelah semua faktor ditampilkan, program memeriksa apakah jumlah faktor sama dengan 2. Sebuah bilangan dikatakan prima apabila hanya memiliki tepat dua faktor, yaitu 1 dan bilangan itu sendiri. Kondisi ini disimpan dalam variabel **prime**.

Terakhir, program mencetak hasil pengecekan dengan menampilkan **"true"** jika bilangan adalah prima dan **"false"** jika bukan.