

LAPORAN PRAKTIKUM ALGORITMA

DAN PEMROGRAMAN 1

MODUL 10

ELSE IF



Disusun oleh:

JOSHUA NATHANIEL

109082530033

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 umur int
    var kk bool
    fmt.Print("Masukkan umur: ")
    fmt.Scan(&umur)
    fmt.Print("Apakah sudah memiliki kk? : ")
    fmt.Scan(&kk)
    if umur >= 17 && kk == true {
        fmt.Println("bisa membuat KTP")
    } else {
        fmt.Println("belum bisa membuat ktp")
    }
}
```

Screenshot program

The screenshot shows a Go development environment with three main panes:

- Code Editor:** Displays the source code for a Go program named `guided1.go`. The code prompts the user for their age and whether they have a KK (KTP Kartu Keluarga). It checks if the user is 17 or older and has a KK. If both conditions are met, it prints "Bisa membuat KTP". Otherwise, it prints "Belum bisa membuat KTP".
- Terminal:** Shows the command `go run "c:\Users\ASUS\Documents\LAPRAK MODUL 10\guided1.go"` being run in a Windows command prompt. The terminal output shows two runs: one where the user has a KK and one where they do not.
- Output Window:** A small window titled "NA" displays the program's output for each run, showing the user's name, NIM, and class information.

Deskripsi program

Program ini bekerja dengan cara menanyakan umur pengguna dan apakah mereka sudah memiliki KK, lalu memeriksa kedua syarat itu untuk menentukan apakah seseorang bisa membuat KTP atau tidak. Jika umur yang dimasukkan sudah 17 tahun atau lebih dan jawabannya menunjukkan bahwa pengguna sudah memiliki KK, maka program menampilkan pesan bahwa mereka bisa membuat KTP. Sebaliknya, kalau salah satu dari syarat tersebut belum terpenuhi, program langsung memberi tahu bahwa pengguna belum bisa membuat KTP.

2. Guided 2

Source Code

```
package main

import "fmt"
```

```
func main() {
    var a rune

    fmt.Print("Masukkan suatu huruf: ")

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

    if a == 'A' || a == 'I' || a == 'U' || a == 'E' || a ==
    'O' || a == 'a' || a == 'i' || a == 'u' || a == 'e' || a ==
    'o' {
        fmt.Println("Vokal")
    } else if (a >= 'a' && a <= 'z') || (a >= 'A' && a <=
    'Z') {
        fmt.Println("konsonan")
    } else {
        fmt.Println("bukan huruf")
    }
}
```

Screenshot program

The screenshot shows a code editor with two tabs: 'guided1.go' and 'guided2.go'. The 'guided2.go' tab contains the following Go code:

```

1 package main
2
3 import "fmt"
4
5 func main() {
6     var a rune
7
8     fmt.Println("Masukkan suatu huruf: ")
9     fmt.Scan(&a)
10
11    if a == 'A' || a == 'I' || a == 'U' || a == 'E' || a == 'O' || a == 'a' || a == 'i' || a == 'u' || a == 'e' || a == 'o' {
12        fmt.Println("Vokal")
13    } else if (a >= 'a' && a <= 'z') || (a >= 'A' && a <= 'Z') {
14        fmt.Println("konsonan")
15    } else {
16        fmt.Println("bukan huruf")
17    }
18 }

```

Below the code editor is a terminal window showing the execution of the program:

```

PS C:\Users\ASUS\Documents\LAPRAK MODUL 10> go run \guided2.go
Masukkan suatu huruf: a
Vokal
PS C:\Users\ASUS\Documents\LAPRAK MODUL 10> go run \guided2.go
Masukkan suatu huruf: f
konsonan
PS C:\Users\ASUS\Documents\LAPRAK MODUL 10> go run \guided2.go
Masukkan suatu huruf: 1
bukan huruf
PS C:\Users\ASUS\Documents\LAPRAK MODUL 10> go run \guided2.go
Masukkan suatu huruf: $
bukan huruf
PS C:\Users\ASUS\Documents\LAPRAK MODUL 10>

```

Deskripsi program

Program ini meminta pengguna memasukkan satu karakter, lalu program mengecek apakah karakter tersebut termasuk huruf vokal, huruf konsonan, atau bahkan bukan huruf sama sekali. Jika karakter yang dimasukkan adalah salah satu vokal seperti A, I, U, E, O (baik huruf besar maupun kecil), program akan menampilkan “Vokal”. Jika bukan vokal tetapi masih berada dalam rentang huruf alfabet, maka program menganggapnya sebagai “konsonan”. Namun, jika karakter itu tidak termasuk huruf sama sekali—misalnya angka atau simbol—program akan menampilkan “bukan huruf”.

3. Guided 3

Source Code

```

package main

import "fmt"

func main() {
    var angka, d1, d2, d3, d4 int
}

```

```
var urutan string

fmt.Println("Masukkan angka nya cuy: ")

fmt.Scan(&angka)

d1 = angka / 1000

d2 = (angka % 1000) / 100

d3 = (angka %100) / 10

d4 = angka % 10

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

    urutan = "terurut membesar"

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

    urutan = "terurut mengecil"

} else {

    urutan = "tidak terurut"

}

fmt.Println("digit pada bilangan" , angka, " ", urutan)

}
```

Screenshot program

The screenshot shows a Go development environment. At the top, there are three tabs: 'guided1.go 2', 'guided2.go 1', and 'guided3.go 1'. The main window displays the following Go code:

```
1 package main
2
3 import "fmt"
4
5 func main() {
6     var angka, d1, d2, d3, d4 int
7     var urutan string
8
9     fmt.Println("Masukkan angka nya cuy: ")
10    fmt.Scan(&angka)
11
12    d1 = angka / 1000
13    d2 = (angka % 1000) / 100
14    d3 = (angka % 100) / 10
15    d4 = angka % 10
16
17    if d1 < d2 && d2 < d3 && d3 < d4 {
18        urutan = "terurut membesar"
19    } else if d1 > d2 && d2 > d3 && d3 > d4 {
20        urutan = "terurut mengecil"
21    } else {
22        urutan = "tidak terurut"
23    }
24    fmt.Println("digit pada bilangan ", angka, ", urutan")
25 }
```

In the center, a terminal window shows the following command-line interaction:

```
PS C:\Users\ASUS\Documents\LAPRAK MODUL 10> go run .\guided3.go
Masukkan angka nya cuy: 2489
digit pada bilangan 2489 terurut membesar
PS C:\Users\ASUS\Documents\LAPRAK MODUL 10> go run .\guided3.go
Masukkan angka nya cuy: 3861
digit pada bilangan 3861 tidak terurut
PS C:\Users\ASUS\Documents\LAPRAK MODUL 10> go run .\guided3.go
Masukkan angka nya cuy: 9651
digit pada bilangan 9651 terurut mengecil
PS C:\Users\ASUS\Documents\LAPRAK MODUL 10> []
```

At the bottom, a status bar indicates: Ln 2, Col 17 55 character Plain t 100% Wind UTF-8.

Deskripsi program

Program ini mengambil sebuah angka empat digit, lalu memecahnya menjadi digit pertama (ribuan), kedua (ratusan), ketiga (puluhan), dan keempat (satuan). Setelah digit-digit itu dipisah, program mengecek apakah urutannya membesar, mengecil, atau acak. Kalau digit pertama lebih kecil dari digit kedua, digit kedua lebih kecil dari digit ketiga, dan seterusnya sampai digit keempat, maka angkanya dianggap “terurut membesar”. Kalau semuanya justru semakin kecil dari kiri ke kanan, hasilnya “terurut mengecil”. Tapi kalau polanya tidak konsisten—tidak sepenuhnya naik atau turun—program menyimpulkan bahwa angka tersebut “tidak terurut”.

TUGAS

1. Tugas 1

Source code

```
package main
```

```
import "fmt"

func main() {
    var beratParsel, kg, gram, biayaKg, biayaSisa, totalBiaya
    int

    fmt.Println("Berat parsel (gram): ")
    fmt.Scan(&beratParsel)

    kg = beratParsel / 1000
    gram = beratParsel % 1000

    biayaKg = kg * 10000

    if kg >= 10 {
        biayaSisa = 0
    } else if gram >= 500 {
        biayaSisa = gram * 5
    } else {
        biayaSisa = gram * 15
    }

    totalBiaya = biayaKg + biayaSisa

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

```
    fmt.Println("Detail biaya: Rp.", biayaKg, "+ Rp.",
biayaSisa)

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

}
```

Screenshot program

```

func main() {
    var beratParsel, kg, gram, biayaKg, biayaSisa, totalBiaya int
    fmt.Println("Berat parsel (gram): ")
    fmt.Scan(&beratParsel)
    kg = beratParsel / 1000
    gram = beratParsel % 1000
    biayaKg = kg * 10000
    if kg >= 10 {
        biayaSisa = 0
    } else if gram >= 500 {
        biayaSisa = gram * 5
    } else {
        biayaSisa = gram * 15
    }
    totalBiaya = biayaKg + biayaSisa
    fmt.Println("Detail berat:", kg, "kg +", gram, "gr")
    fmt.Println("Detail biaya: Rp.", biayaKg, "+ Rp.", biayaSisa)
    fmt.Println("Total biaya: Rp.", totalBiaya)
}

```

PS C:\Users\ASUS\Documents\LAPRAK MODUL 10> go run .\tugasi1.go
Berat parsel (gram): 8500
Detail berat: 8 kg + 500 gr
Detail biaya: Rp. 80000 + Rp. 2500
Total biaya: Rp. 82500
PS C:\Users\ASUS\Documents\LAPRAK MODUL 10> go run .\tugasi1.go
Berat parsel (gram): 9250
Detail berat: 9 kg + 250 gr
Detail biaya: Rp. 90000 + Rp. 3750
Total biaya: Rp. 93750
PS C:\Users\ASUS\Documents\LAPRAK MODUL 10> go run .\tugasi1.go
Berat parsel (gram): 11750
Detail berat: 11 kg + 750 gr
Detail biaya: Rp. 110000 + Rp. 0
Total biaya: Rp. 110000
PS C:\Users\ASUS\Documents\LAPRAK MODUL 10>

Deskripsi program

Program ini menghitung biaya pengiriman parsel berdasarkan berat yang dimasukkan dalam gram. program memecah berat tersebut menjadi kilogram dan gram. Setiap kilogram dikenai biaya tetap Rp10.000. Setelah itu, sisa gram dihitung biayanya dengan aturan: jika total kilogram sudah mencapai 10 kg atau lebih, sisa gram tidak dikenakan biaya sama sekali. Namun jika kurang dari 10 kg, sisa gram yang 500 gram ke atas dihitung Rp5 per gram, sedangkan sisa gram di bawah 500 dihitung Rp15 per gram. Hasil akhirnya adalah penjumlahan biaya per kilogram dan biaya dari sisa gram

2. Tugas 2

A. Jika nam diberikan adalah 80.1, apa keluaran dari program tersebut? Apakah eksekusi program tersebut sesuai spesifikasi soal?

Jawab : Jika input yang diberikan adalah 80.1, program tidak akan menghasilkan keluaran apa pun karena akan terjadi kesalahan saat pembacaan input; variabel beratParsel dideklarasikan sebagai integer sehingga tidak dapat menerima nilai desimal seperti 80.1. Akibatnya, program tidak akan berjalan sesuai spesifikasi jika soal mengharuskan mampu memproses nilai berat dengan koma, karena program ini hanya dirancang untuk menerima dan menghitung berat dalam bilangan bulat.

B. Apa saja kesalahan dari program tersebut? Mengapa demikian? Jelaskan alur

program seharusnya!

Jawab: Berikut penjelasannya dalam satu paragraf yang jelas dan bahasa manusiawi:

Kesalahan utama dari program tersebut adalah penggunaan tipe data int untuk membaca berat parsel, sehingga input desimal seperti 80.1 tidak bisa diproses dan akan menyebabkan error saat membaca data. Selain itu, perhitungan biaya sisa gram juga menjadi tidak akurat jika berat harus mendukung angka desimal, karena pembagian dan modulus hanya valid untuk bilangan bulat. Seharusnya program menggunakan tipe data float untuk menerima berat dalam gram, kemudian mengubahnya menjadi kilogram dan sisa gram dengan perhitungan yang benar, lalu menerapkan aturan biaya berdasarkan bobot tersebut. Alur program yang benar adalah: membaca berat dalam bentuk desimal, mengonversinya menjadi kg dan gram, menentukan biaya per kilogram, menghitung biaya tambahan sesuai aturan soal, kemudian menjumlahkannya dan menampilkan hasil akhir tanpa terjadi kesalahan tipe data.

C. Perbaiki program tersebut! Ujilah dengan masukan: 93.5; 70.6; dan 49.5.

Seharusnya keluaran yang diperoleh adalah 'A', 'B', dan 'D'.

Jawab:

Source code

```
package main

import "fmt"

func main() {

    var nam float64

    var nmk string


    fmt.Println("Nilai akhir mata kuliah: ")

    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 if nam <= 40 {

nmk = "E"

}

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

}

```

Screenshot program

The screenshot shows a Go development environment with the following details:

- Code Editor:** A code editor window titled "tugas2.go" containing the Go code provided in the previous snippet.
- Terminal:** A terminal window showing the execution of the program. It displays the input "Nilai Kuliah: 70,6" and the output "Nilai Akhir Mata Kuliah: 70,6".
- Output Window:** A window titled "Output" showing the command "go run .\vtugas2.go" and the resulting output "Nilai Akhir Mata Kuliah: 70,6".
- File Explorer:** A sidebar showing the project structure with files like "guided1.go", "guided2.go", "guided3.go", "tugas1.go", "tugas2.go", and "tugas3.go".

DESKRIPSI

Program ini digunakan untuk menentukan nilai huruf berdasarkan nilai akhir mata kuliah yang dimasukkan pengguna. Nilai akhir tersebut dibaca sebagai angka desimal, lalu program membandingkannya dengan beberapa rentang nilai untuk menentukan nilai huruf yang sesuai, mulai dari A untuk nilai di atas 80 hingga E jika nilainya 40 atau kurang. Setiap kondisi dicek secara berurutan menggunakan if dan else if, sehingga hanya satu nilai huruf yang akan dipilih. Setelah rentang yang cocok ditemukan, program menampilkan nilai huruf tersebut sebagai hasil akhirnya.

3. Tugas 3_1

Source code

```
package main

import "fmt"

func main() {
    var n, i int
    fmt.Print("masukkan bilangan: ")
    fmt.Scan(&n)

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

Screenshot program

```
1 package main
2 import "fmt"
3
4 func main() {
5     var n, i int
6     fmt.Print("masukkan bilangan: ")
7     fmt.Scan(&n)
8
9     fmt.Print("faktor: ")
10    for i = 1; i <= n; i++ {
11        if n % i == 0 {
12            fmt.Print(" ", i)
13        }
14    }
15 }
```

NAMA: JOSHUA NATHANIEL
NIM: 109082530033
KELAS: SIIF - 13 - 07

Deskripsi program

Program ini bekerja dengan cara meminta memasukkan sebuah angka, lalu program mencoba mencari angka-angka mana saja yang bisa membagi angka tersebut tanpa sisa. Untuk melakukannya, program memeriksa semua angka mulai dari 1 sampai angka yang dimasukkan. Setiap kali ditemukan angka yang dapat membagi habis angka awal, angka itu dianggap sebagai faktor dan langsung ditampilkan. Hasil akhirnya adalah daftar faktor dari angka tersebut, dituliskan berurutan dari kecil ke besar.

4. Tugas 3_2

Source code

```
package main

import "fmt"

func main() {
    var n, i, faktor int
    var prima bool
    fmt.Print("masukkan bilangan: ")
    fmt.Scan(&n)
    faktor = 0

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

    fmt.Println()

    fmt.Print("Prima: ")
    if faktor == 2 {
        prima = true
    }
    fmt.Print(prima)
}
```

Screenshoot program

The screenshot shows a code editor interface with several tabs at the top. The active tab contains Go code for finding factors and checking primality. The code uses standard library imports like `fmt`. It prompts the user for a number, iterates through all numbers from 1 to n to find factors, and then checks if the number has exactly two factors (1 and itself). The output window shows sample runs for the numbers 12 and 7.

```
1 package main
2 import "fmt"
3
4 func main() {
5     var n, i, faktor int
6     var prima bool
7     fmt.Println("MASUKAN BILANGAN : ")
8     fmt.Scan(&n)
9     faktor = 0
10
11    fmt.Println("faktor: ")
12    for i = 1; i <= n; i++ {
13        if n % i == 0 {
14            fmt.Println(" ", i)
15            faktor++
16        }
17    }
18
19    fmt.Println()
20
21    fmt.Println("Prima: ")
22    if faktor == 2 {
23        prima = true
24    }
25    fmt.Println(prima)
26 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL FORTS

```
PS C:\Users\ASUS\Documents\LAPRAK MODUL 10> go run .\tugas3_2.go
MASUKAN BILANGAN : 12
faktor: 1 2 3 4 6 12
Prima: false
PS C:\Users\ASUS\Documents\LAPRAK MODUL 10> go run .\tugas3_2.go
MASUKAN BILANGAN : 7
faktor: 1 7
Prima: true
PS C:\Users\ASUS\Documents\LAPRAK MODUL 10>
```

Deskripsi program

Program mencari faktor-faktornya dengan memeriksa setiap angka dari 1 sampai angka tersebut; setiap angka yang bisa dibagi habis nilai n dicetak sebagai faktor, dan jumlah faktornya dihitung. Setelah semua faktor ditemukan, program menentukan apakah bilangan tersebut prima atau tidak—yaitu bilangan yang hanya memiliki dua faktor, yaitu 1 dan dirinya sendiri. Jika jumlah faktornya tepat dua, variabel prima diisi nilai true, dan jika tidak, nilainya tetap false. Di akhir, program menampilkan daftar faktor bilangan tersebut dan apakah bilangan itu termasuk bilangan prima.