LAPORAN PRAKTIKUM ALGORITMA

DAN PEMROGRAMAN 1

MODUL 4

I/O, TIPE DATA & VARIABLE

Disusun oleh:

MUHAMMAD FIRDAUS ARDIANSYAH

109082500126

S1IF-13-07

Asisten Praktikum

Adithana dharma putra

Apri pandu wicaksono

PROGRAM STUDI S1 INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

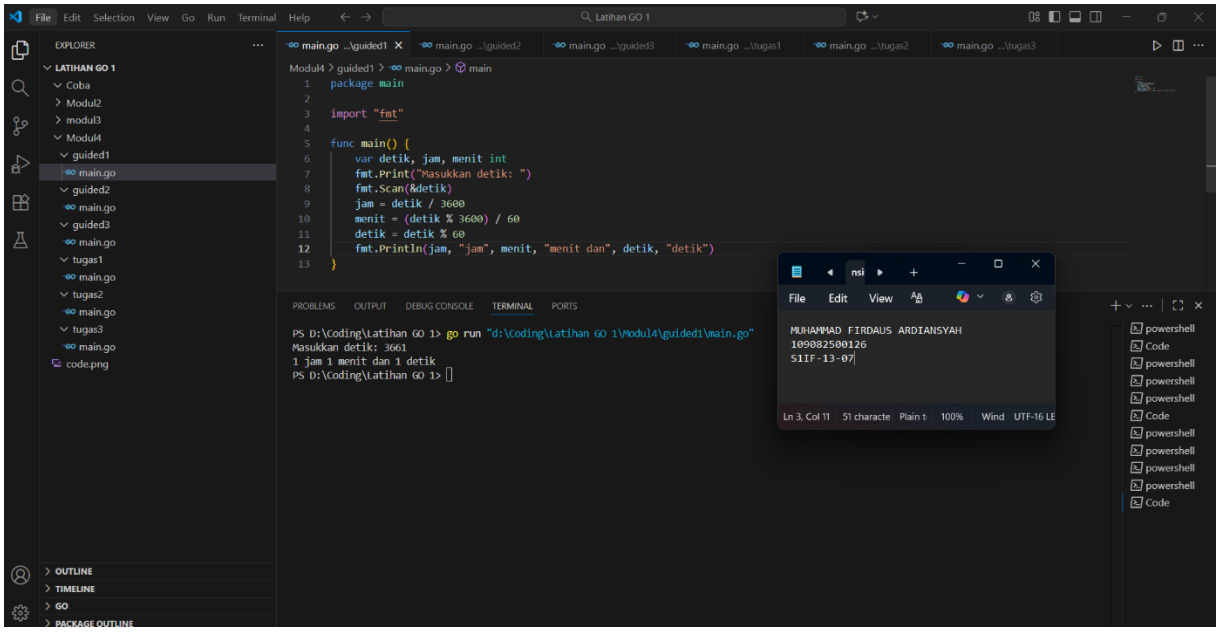
LATIHAN KELAS – GUIDED

1. **Guided 1**

Source Code

```
package main


import "fmt"


func main() {
    var detik, jam, menit int
    fmt.Print("Masukkan detik: ")
    fmt.Scan(&detik)
    jam = detik / 3600
    menit = (detik % 3600) / 60
    detik = detik % 60
    fmt.Println(jam, "jam", menit, "menit dan", detik,
"detik")
}
```

Screenshoot program



Deskripsi program

Ini adalah program sederhana yang ditulis dalam bahasa Go (package main), dirancang untuk mengambil total durasi waktu dalam satuan detik dan mengubahnya menjadi format jam, menit, dan detik sisa. Intinya, program ini meminta pengguna memasukkan jumlah detik. Kemudian, ia menggunakan operasi matematika dasar—tepatnya, pembagian untuk mendapatkan jam dan modulo (%) untuk menghitung sisa waktu—guna memilah total detik menjadi komponen jam, menit, dan detik yang terpisah.

Setelah semua perhitungan selesai, hasilnya dicetak langsung ke konsol. Tampilan terminal menunjukkan program berhasil mengonversi input (yang tampaknya adalah 3661 detik) menjadi "1 jam 1 menit dan 1 detik". Ini adalah contoh program pemula yang jelas, menunjukkan cara kerja operasi input/output (fmt.Scan, fmt.Println) dan logika aritmatika dasar di Go untuk memecah satuan waktu.

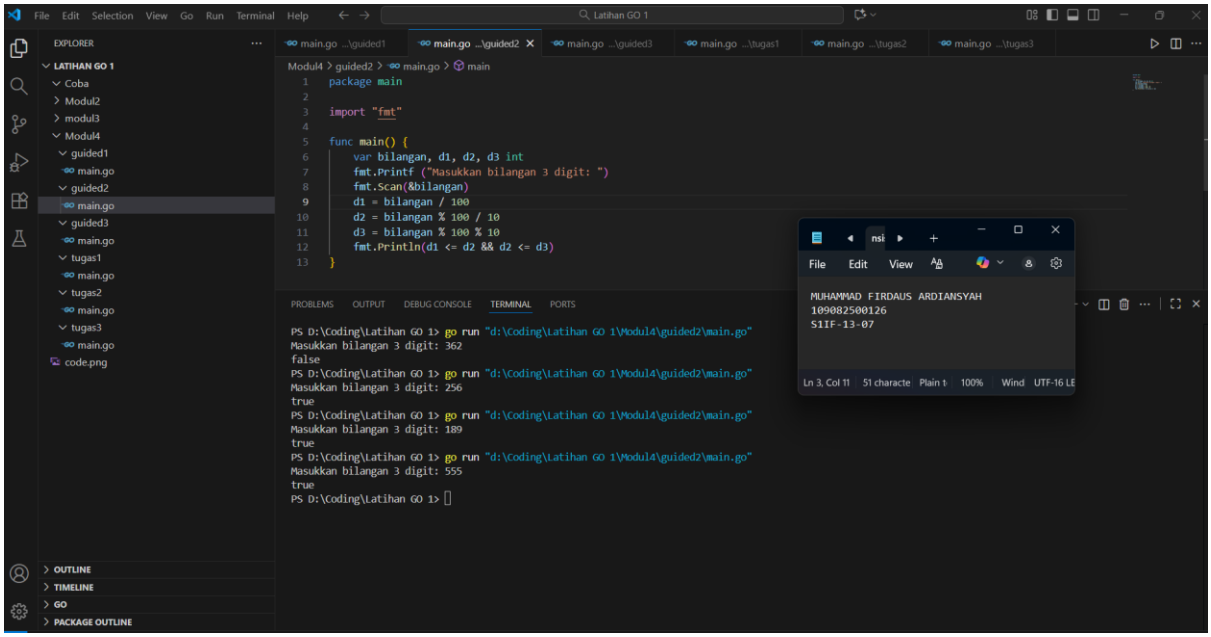
2. Guided 2

Source Code

```
package main


import "fmt"


func main() {

    var bilangan, d1, d2, d3 int

    fmt.Printf ("Masukkan bilangan 3 digit: ")
```

```
        fmt.Scan(&bilangan)

        d1 = bilangan / 100

        d2 = bilangan % 100 / 10

        d3 = bilangan % 100 % 10

        fmt.Println(d1 <= d2 && d2 <= d3)

    }
```

Screenshoot program



Deskripsi program

Program ini meminta kita memasukkan bilangan tiga digit dan segera memecahnya menjadi tiga digit penyusunnya. Proses pemecahan menggunakan kombinasi operasi pembagian integer dan operator modulo (%) secara berurutan: Digit pertama (d1) diambil dengan membagi bilangan dengan 100; Digit kedua (d2) didapatkan dari sisa bagi dengan 100, lalu hasilnya dibagi 10; dan Digit ketiga (d3) diperoleh dari sisa bagi dengan 10.

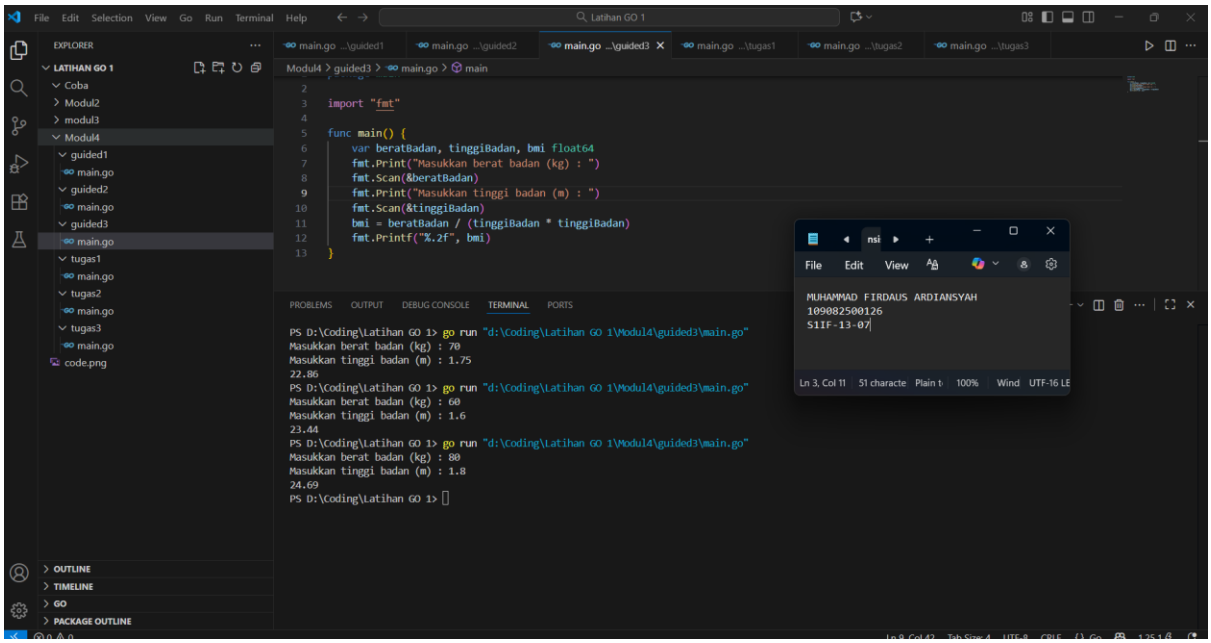
Setelah digit-digit (d1,d2,d3) berhasil diisolasi, program menjalankan sebuah operasi logika Boolean. Tujuannya adalah mengecek apakah digit-digit tersebut tersusun dalam urutan menurun atau sama (d1≥d2≥d3). Hasil dari pengecekan kondisi ini yaitu true atau false kemudian dicetak sebagai *output* tunggal. Pada contoh di terminal, input 362 menghasilkan d1=3, d2=6, d3=2. Karena 6 tidak lebih kecil dari 3, kondisi tersebut gagal, sehingga *output* yang dihasilkan adalah false.

3. Guided 3

Source Code

```
package main


import "fmt"


func main() {
    var beratBadan, tinggiBadan, bmi float64
    fmt.Print("Masukkan berat badan (kg) : ")
    fmt.Scan(&beratBadan)
    fmt.Print("Masukkan tinggi badan (m) : ")
    fmt.Scan(&tinggiBadan)
    bmi = beratBadan / (tinggiBadan * tinggiBadan)
    fmt.Printf("%.2f", bmi)
}
```

Screenshoot program



Deskripsi program

Program Go ini berfungsi sebagai kalkulator sederhana untuk menghitung Indeks Massa Tubuh (BMI). Di awal, program mendeklarasikan variabel beratBadan, tinggiBadan, dan bmi sebagai tipe data float64 karena perhitungan BMI memerlukan angka desimal. Program kemudian meminta pengguna memasukkan berat badan (dalam kg) dan tinggi badan (dalam meter).

Perhitungan BMI dilakukan menggunakan rumus standar: berat badan dibagi kuadrat tinggi badan (BMI= Berat/Tinggi^2). Dalam kode, ini diekspresikan sebagai bmi = beratBadan / (tinggiBadan * tinggiBadan). Setelah menghitungnya, program mencetak nilai BMI ke konsol. Tiga kali eksekusi yang terlihat di terminal menunjukkan program bekerja dengan baik, menghasilkan angka BMI seperti 22.86, 23.44, dan 24.69 berdasarkan *input* berat dan tinggi yang berbeda. Ini adalah contoh penggunaan dasar tipe data *floating-point* dan formula matematika dalam bahasa Go.

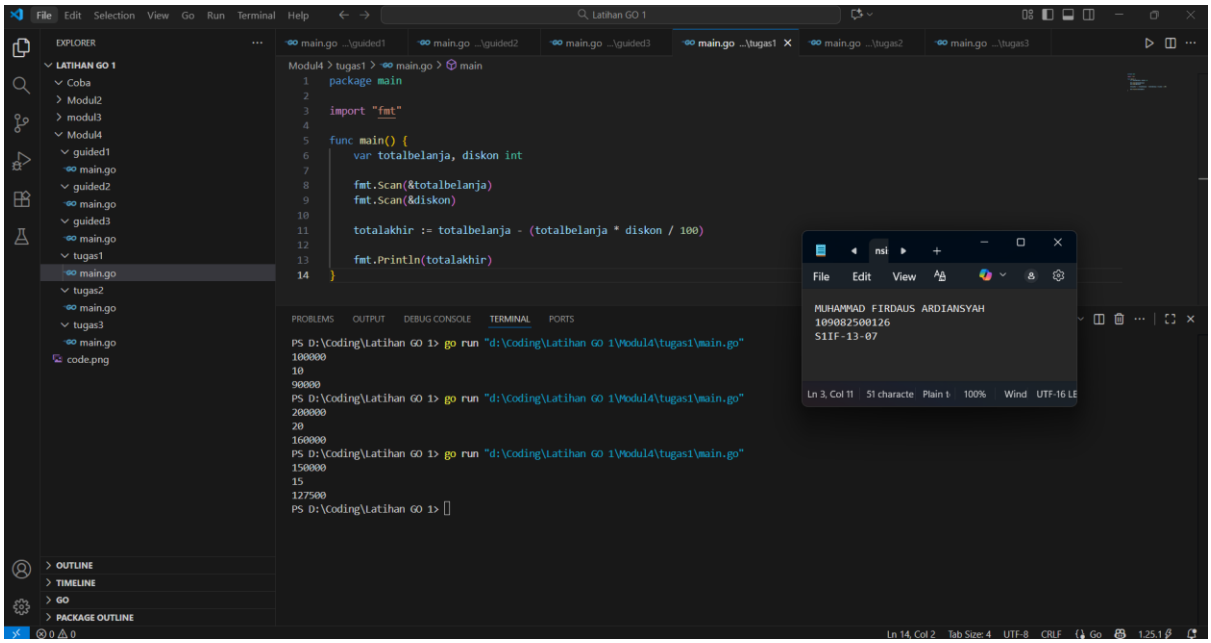
TUGAS

1. **Tugas 1**

   Source code

```
package main


import "fmt"


func main() {
    var totalbelanja, diskon int


    fmt.Scan(&totalbelanja)
    fmt.Scan(&diskon)


    totalakhir := totalbelanja - (totalbelanja * diskon
/ 100)


    fmt.Println(totalakhir)
}
```

Screenshoot program



Deskripsi program

Program Go sederhana ini berfungsi sebagai kalkulator untuk menentukan harga akhir setelah diskon. Program di fungsi main meminta dua *input* dari pengguna: total belanja dan persentase diskon yang diberikan. Kedua variabel ini (totalbelanja dan diskon) dideklarasikan sebagai tipe data int (integer).

Perhitungan harga akhir dilakukan dalam satu baris, dan ini adalah inti dari program:

Totalakhir=totalbelanja – (totalbelanja x diskon/100)

Pada kode, ini ditulis sebagai totalakhir = totalbelanja - (totalbelanja * diskon / 100). Program ini mengambil total belanja, menghitung nilai diskon (yaitu totalbelanja * diskon / 100), lalu menguranginya dari total belanja awal untuk mendapatkan harga akhir (totalakhir). Hasil harga akhir ini kemudian dicetak ke konsol. Tiga contoh eksekusi di terminal menunjukkan operasi ini berjalan benar, seperti menghitung harga akhir dari belanja Rp200.000 dengan diskon 20%, yang menghasilkan Rp160.000.

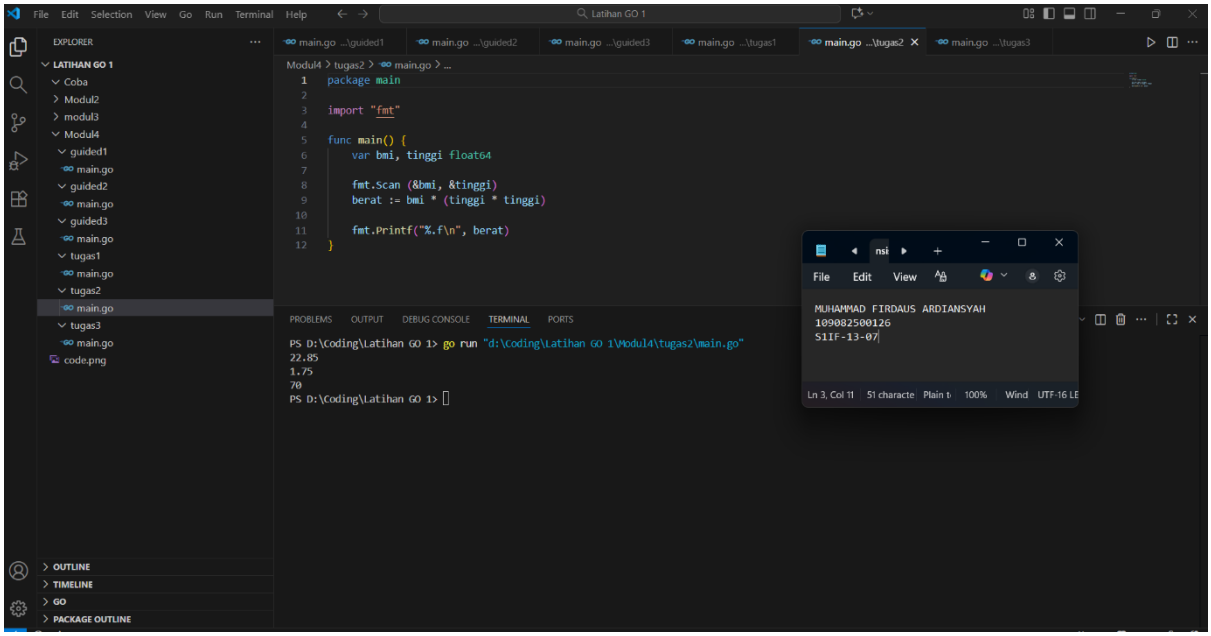
2. Tugas 2

Source code

```
package main


import "fmt"
```

```
func main() {
    var bmi, tinggi float64

    fmt.Scan (&bmi, &tinggi)
    berat := bmi * (tinggi * tinggi)

    fmt.Printf("%.f\n", berat)
}
```

Screenshoot program



Deskripsi program

Program Go sederhana ini bertugas menghitung berat badan yang seharusnya berdasarkan input BMI ideal dan tinggi badan seseorang. Program mendeklarasikan tiga variabel, bmi, tinggi, dan berat, semuanya bertipe float64 untuk mengakomodasi nilai desimal. Program kemudian meminta pengguna memasukkan dua nilai: nilai BMI ideal dan tinggi badan (dalam meter). Perhatikan, meskipun kodenya memanggil variabel bmi dan tinggi, urutan input dari terminal tampaknya adalah BMI diikuti Tinggi (22.85 dan 1.75).

Inti dari program ini adalah mengaplikasikan kembali rumus BMI yang dimodifikasi untuk mencari berat badan. Rumus aslinya adalah BMI=Berat/Tinggi^2. Program ini mengubahnya menjadi: BMI x Tinggi^2. Dalam kode, ini diekspresikan sebagai berat = bmi * (tinggi * tinggi). Setelah mendapatkan nilai berat yang dihasilkan, program mencetak hasilnya. Berdasarkan *output* terminal, dengan BMI 22.85 dan Tinggi 1.75, berat badan ideal yang dihitung adalah 70 kg. Ini menunjukkan program ini efektif membalikkan rumus BMI untuk menghitung berat badan yang sesuai.

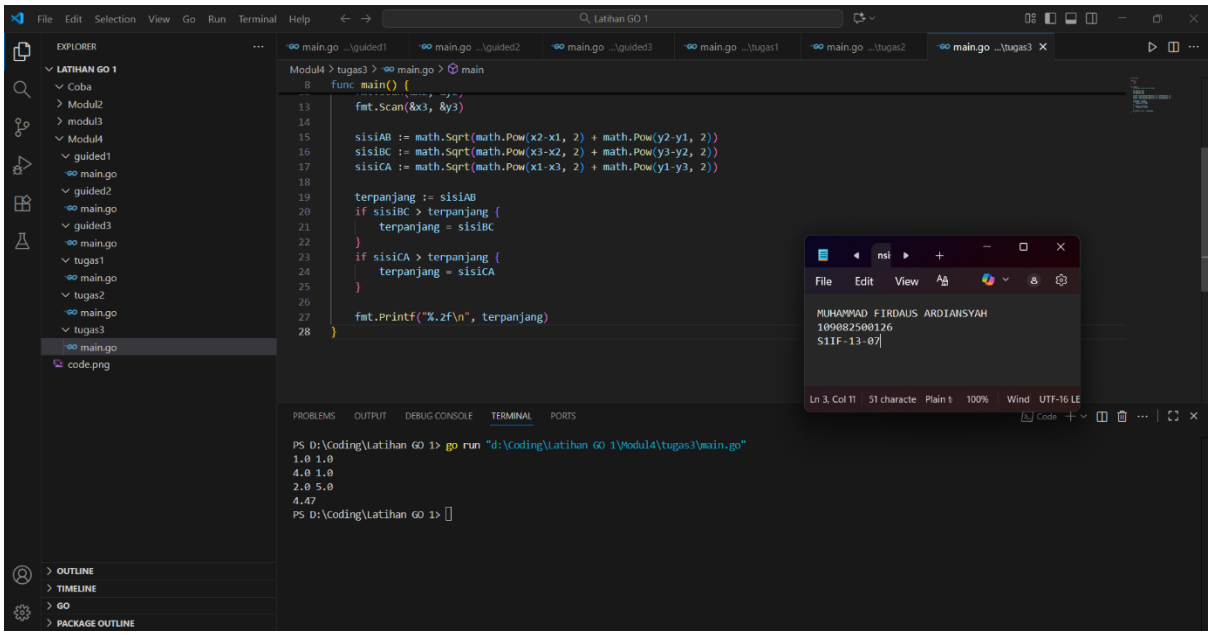
3. Tugas 3

Source code

```
package main

import (
    "fmt"
    "math"
)

func main() {
    var x1, y1, x2, y2, x3, y3 float64

    fmt.Scan(&x1, &y1)
    fmt.Scan(&x2, &y2)
    fmt.Scan(&x3, &y3)

    sisiAB := math.Sqrt(math.Pow(x2-x1, 2) +
math.Pow(y2-y1, 2))
    sisiBC := math.Sqrt(math.Pow(x3-x2, 2) +
math.Pow(y3-y2, 2))
    sisiCA := math.Sqrt(math.Pow(x1-x3, 2) +
math.Pow(y1-y3, 2))

    terpanjang := sisiAB
    if sisiBC > terpanjang {
        terpanjang = sisiBC
    }
    if sisiCA > terpanjang {
        terpanjang = sisiCA
```

```
        }

        fmt.Printf("%.2f\n", terpanjang)
    }
```

Screenshoot program



Deskripsi program

Program ini dimulai dengan meminta enam *input* dari kita, yang mewakili koordinat (x,y) dari tiga titik sudut segitiga. Menggunakan fungsi-fungsi dari paket math seperti math.Sqrt (akar kuadrat) dan math.Pow (pangkat), program ini menghitung panjang ketiga sisi segitiga (sisi AB, sisi BC, sisi CA) menggunakan rumus jarak antara dua titik.

Setelah panjang ketiga sisi (semuanya bertipe *float64*) dihitung, program menggunakan serangkaian pernyataan kondisional if untuk membandingkan nilai-nilai tersebut. Variabel terpanjang diinisialisasi dengan sisi AB , kemudian diperbarui jika sisi BC atau sisi CA ternyata lebih besar. Logika perbandingan ini menjamin bahwa variabel terpanjang akan menyimpan nilai maksimum dari ketiga sisi. Nilai panjang sisi terpanjang tersebut kemudian dicetak ke konsol. Berdasarkan *output* terminal, program berhasil menemukan dan mencetak 4.47 sebagai panjang sisi terpanjang untuk koordinat yang dimasukkan.