

**LAPORAN PRAKTIKUM ALGORITMA
DAN PEMROGRAMAN 1**

**MODUL 4-I/O
TIPE DATA & VARIABEL**



Disusun oleh:

NAMA : Primatama Sigalingging

NIM : 109082500076

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.Scan(&detik)

    jam = detik / 3600

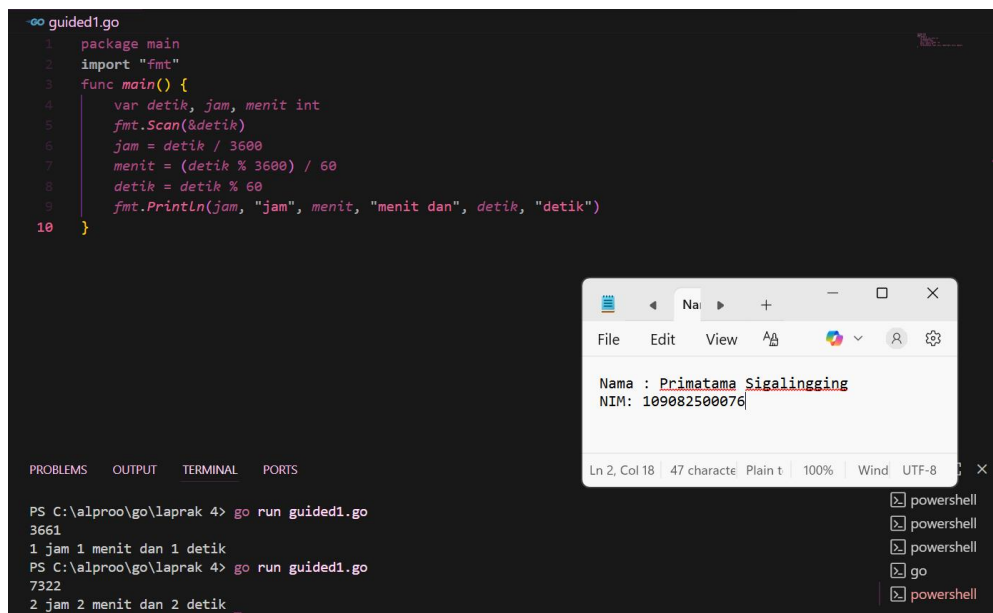
    menit = (detik % 3600) / 60

    detik = detik % 60

    fmt.Println(jam, "jam", menit, "menit dan", detik,
"detik")

}
```

Screenshoot program



Deskripsi program

➤ Tujuan Program

Program ini dibuat untuk mengubah input berupa jumlah detik menjadi bentuk waktu yang lebih mudah dibaca yaitu jam, menit, dan detik.

Misalnya, kalau seseorang tahu total waktunya 3665 detik, program ini akan memberitahukan bahwa waktu tersebut sama dengan 1 jam, 1 menit, dan 5 detik.

- Saat program dijalankan, pengguna diminta untuk memasukkan sebuah angka yang mewakili total detik.
Program kemudian akan melakukan perhitungan otomatis untuk mengetahui berapa jam, berapa menit, dan berapa detik yang setara dengan jumlah detik tersebut.

Prosesnya dilakukan dengan langkah berikut:

1. Jumlah jam dihitung dengan membagi total detik dengan 3600 (karena 1 jam = 3600 detik).
2. Setelah jam dihitung, sisa detik yang belum dikonversi digunakan untuk menghitung menit, yaitu dengan **(detik % 3600) / 60**.
3. Sisa terakhir setelah konversi menit adalah jumlah detik yang tersisa, yaitu detik **% 60**.
4. Hasil akhirnya ditampilkan dalam format kalimat yang mudah dipahami, seperti:
1 jam 1 menit dan 5 detik.

- Kesimpulan

Program ini berguna untuk mengubah satuan waktu dari detik menjadi jam, menit, dan detik, sehingga pengguna bisa memahami durasi waktu dengan lebih jelas.

Struktur perhitungannya sederhana tapi efisien, menggunakan pembagian dan modulus untuk memecah total detik menjadi satuan waktu yang lebih informatif.

2. Guided 2

Source Code

```
package main

import "fmt"

func main() {
    var bilangan, d1, d2, d3 int
    fmt.Scan(&bilangan)

    d1 = bilangan / 100
    d2 = bilangan % 100 / 10
    d3 = bilangan % 100 % 10

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

Screenshoot program

The screenshot shows a Go program named `guided2.go` and its execution results. The program is designed to check if a three-digit number is in non-decreasing order. It prompts the user to enter a number, which is then split into its hundreds, tens, and units digits. These digits are compared to see if they are in non-decreasing order. If they are, the program returns `true`; otherwise, it returns `false`.

```
1 package main
2 import "fmt"
3 func main() {
4     var bilangan, d1, d2, d3 int
5     fmt.Scan(&bilangan)
6     d1 = bilangan / 100
7     d2 = bilangan % 100 / 10
8     d3 = bilangan % 100 % 10
9     fmt.Println(d1 <= d2 && d2 <= d3)
10 }
```

The terminal output shows two test cases:

```
PS C:\alproo\go\laprak 4> go run guided2.go
362
false
PS C:\alproo\go\laprak 4> go run guided2.go
256
true
```

Deskripsi program

➤ Tujuan Program

Program ini dibuat untuk memeriksa apakah tiga digit dalam suatu bilangan tiga angka tersusun secara tidak menurun (artinya dari kiri ke kanan nilainya tidak berkurang). Dengan kata lain, program akan mengembalikan nilai `true` jika urutannya naik atau tetap, dan `false` jika tidak.

➤ Ketika program dijalankan, pengguna diminta untuk memasukkan satu bilangan bulat tiga digit misalnya **345** atau **421**.

Setelah angka dimasukkan, program akan memisahkan setiap digitnya:

1. Digit pertama (ratusan) disimpan dalam `d1`.
2. Digit kedua (puluhan) disimpan dalam `d2`.
3. Digit ketiga (satuan) disimpan dalam `d3`.

Caranya menggunakan operasi pembagian (/) dan modulus (%) untuk mengambil nilai tiap digit:

- `d1 = bilangan / 100` → mengambil angka ratusan
- `d2 = bilangan % 100 / 10` → mengambil angka puluhan
- `d3 = bilangan % 100 % 10` → mengambil angka satuan

Setelah ketiga digit diketahui, program membandingkannya dengan logika:

`d1 <= d2 && d2 <= d3`

Artinya, jika digit pertama \leq digit kedua dan digit kedua \leq digit ketiga, maka hasilnya `true` (urutan angka menaik atau sama).

Jika tidak memenuhi, maka hasilnya `false`.

➤ Kesimpulan

Program ini sederhana tapi logis:

Ia memecah bilangan menjadi tiga digit, lalu mengecek apakah angka-angka tersebut tersusun dari kecil ke besar.

Hasil true menunjukkan urutan digit menaik (atau tetap sama), sedangkan false berarti ada digit yang lebih kecil dari digit sebelumnya.

Dengan cara ini, pengguna bisa dengan mudah mengetahui apakah sebuah bilangan tiga digit memiliki pola urutan angka yang berurutan atau tidak.

3. Guided 3

Source Code

```
package main

import "fmt"

func main() {

    var beratBadan, tinggiBadan, bmi float64

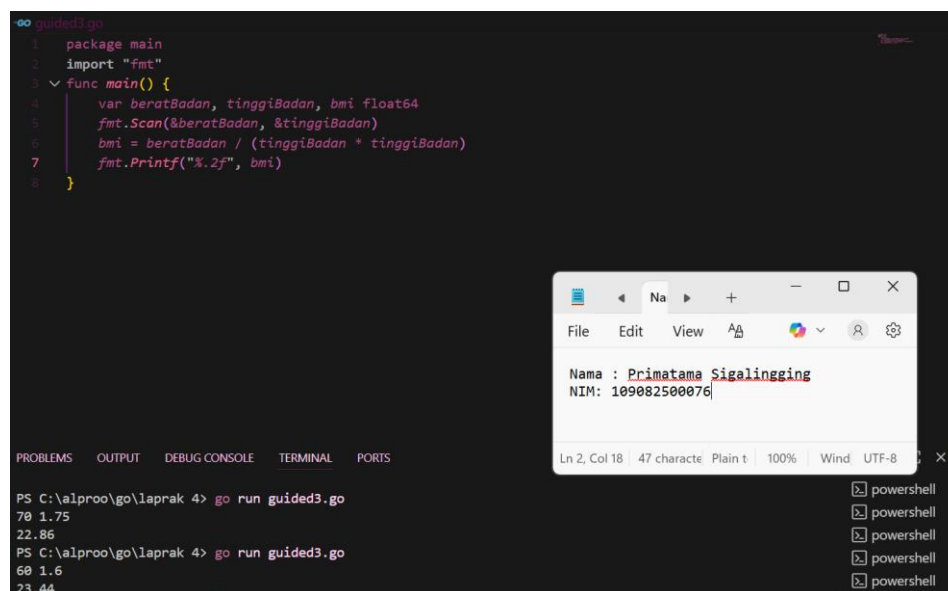
    fmt.Scan(&beratBadan, &tinggiBadan)

    bmi = beratBadan / (tinggiBadan * tinggiBadan)

    fmt.Printf("%.2f", bmi)

}
```

Screenshoot program



Deskripsi program

➤ Tujuan Program:

Program ini bertujuan untuk menghitung **Body Mass Index (BMI)** seseorang berdasarkan berat badan (dalam kilogram) dan tinggi badan (dalam meter).

Nilai BMI digunakan untuk mengukur apakah seseorang termasuk dalam kategori kurus, normal, berat badan berlebih, atau obesitas.

➤ Program ini bekerja dengan cara yang sangat sederhana dan interaktif.

Saat dijalankan, pengguna hanya perlu memasukkan dua nilai:

1. **Berat badan (kg)**

2. **Tinggi badan (m)**

Sebagai contoh, pengguna bisa mengetik: **70 1.75** ; yang berarti berat badan 70 kg dan tinggi badan 1,75 meter.

Setelah menerima input, program akan menghitung nilai BMI menggunakan rumus:

$$\text{BMI} = \frac{\text{berat badan}}{(\text{tinggi badan})^2}$$

Rumus ini berarti berat badan dibagi dengan kuadrat dari tinggi badan.

Jadi, semakin tinggi seseorang dengan berat yang sama, nilai BMI-nya akan lebih kecil karena massa tubuhnya “tersebar” pada tinggi yang lebih besar.

Dalam kode Go, proses ini dilakukan lewat baris:

bmi = beratBadan / (tinggiBadan * tinggiBadan)

Setelah dihitung, hasilnya ditampilkan dengan dua angka di belakang koma agar lebih rapi dan mudah dibaca:

fmt.Printf("%.2f", bmi)

➤ Kesimpulan:

Program ini membantu pengguna untuk mengetahui status berat badan berdasarkan nilai BMI.

Dengan hanya dua input sederhana berat dan tinggi badan pengguna bisa langsung mendapatkan hasil akurat dan terformat rapi.

Selain berguna untuk keperluan pribadi, program ini juga dapat dijadikan dasar untuk aplikasi kesehatan, kalkulator berat badan ideal, atau sistem monitoring kebugaran.

TUGAS

1. Tugas 1

Source code

```
package main

import "fmt"

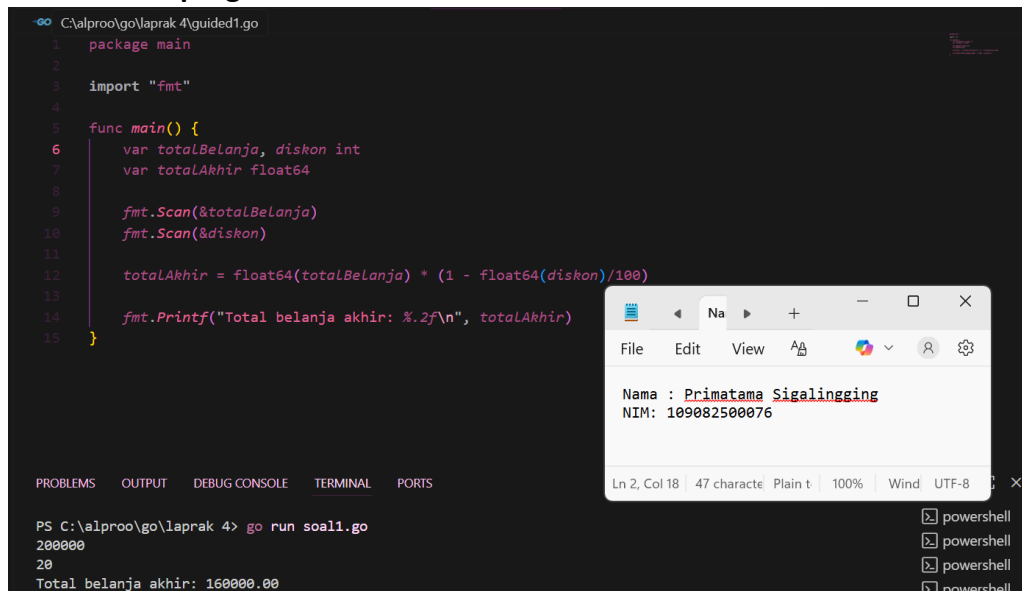
func main() {
    var totalBelanja, diskon int
    var totalAkhir float64

    fmt.Scan(&totalBelanja)
    fmt.Scan(&diskon)

    totalAkhir = float64(totalBelanja) * (1 - float64(diskon)/100)

    fmt.Printf("Total belanja akhir: %.2f\n", totalAkhir)
}
```

Screenshoot program



```
C:\alproo\go\laprak 4\guided1.go
1 package main
2
3 import "fmt"
4
5 func main() {
6     var totalBelanja, diskon int
7     var totalAkhir float64
8
9     fmt.Scan(&totalBelanja)
10    fmt.Scan(&diskon)
11
12    totalAkhir = float64(totalBelanja) * (1 - float64(diskon)/100)
13
14    fmt.Printf("Total belanja akhir: %.2f\n", totalAkhir)
15 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\alproo\go\laprak 4> go run soal1.go
200000
20
Total belanja akhir: 160000.00
```

Ln 2, Col 18 | 47 character Plain t | 100% | Wind UTF-8

powerShell
powerShell
powerShell
powerShell

Deskripsi program

- Tujuan Program:
Program ini dibuat untuk menghitung total belanja akhir setelah mendapatkan potongan harga (diskon) dalam bentuk persentase.
Dengan program ini, pengguna bisa langsung mengetahui berapa banyak uang yang harus dibayar setelah diskon diterapkan pada total belanja awal.
- Program ini bekerja secara sederhana namun praktis.
Ketika dijalankan, pengguna akan diminta memasukkan dua nilai angka:
 1. **Total belanja awal (dalam satuan rupiah)**
 2. **Besaran diskon (dalam persen)**

Sebagai contoh, pengguna bisa mengetik:

200000

20

Artinya, total belanja awal adalah Rp200.000 dan diskon yang diberikan sebesar 20%. Setelah menerima input tersebut, program akan menghitung total harga setelah diskon menggunakan rumus:

$$\text{Total Akhir} = \text{Total Belanja} \times \left(1 - \frac{\text{Diskon}}{100}\right)$$

Rumus ini berarti total belanja dikalikan dengan sisa persentase harga yang harus dibayar. Jika diskonnya 20%, maka pembeli hanya perlu membayar 80% dari total harga awal.

Dalam kode Go, perhitungannya ditulis seperti ini:

totalAkhir = float64(totalBelanja) * (1 - float64(diskon)/100)

Bagian float64 digunakan agar hasil perhitungan bisa memiliki angka desimal, sehingga lebih akurat (terutama untuk nilai rupiah yang tidak bulat). Kemudian hasil akhirnya ditampilkan dengan format dua angka di belakang koma agar terlihat rapi:

fmt.Printf("Total belanja akhir: %.2f\n", totalAkhir)

- Kesimpulan:
Program ini membantu pengguna untuk menghitung total harga yang harus dibayar setelah diskon diterapkan, tanpa perlu menghitung secara manual.
Cukup masukkan dua angka total belanja awal dan persentase diskon maka program akan langsung menampilkan hasil akhirnya secara cepat dan akurat.

Program ini bisa digunakan dalam berbagai situasi, seperti:

- Sistem kasir sederhana
- Aplikasi penjualan online
- Perhitungan promo toko
- Atau sekadar alat bantu untuk menghitung potongan harga sehari-hari

2. Tugas 2

Source code

```
package main

import "fmt"

func main() {

    var bmi, tinggiBadan, beratBadan float64

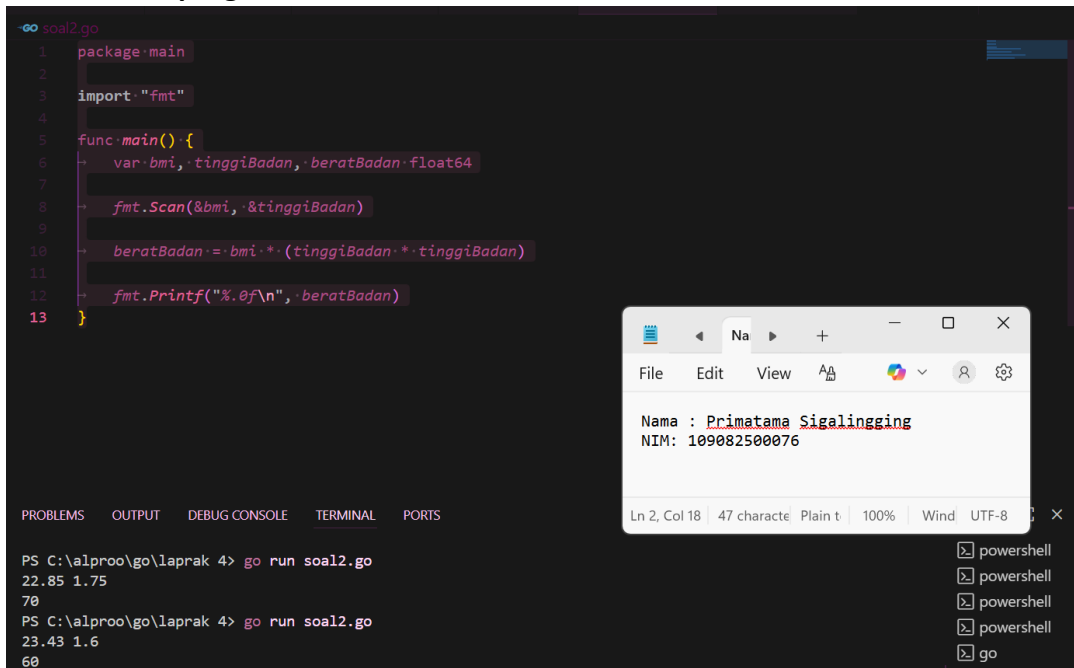
    fmt.Scan(&bmi, &tinggiBadan)

    beratBadan = bmi * (tinggiBadan * tinggiBadan)

    fmt.Printf("%.0f\n", beratBadan)

}
```

Screenshoot program



```
1 package main
2
3 import "fmt"
4
5 func main() {
6     var bmi, tinggiBadan, beratBadan float64
7
8     fmt.Scan(&bmi, &tinggiBadan)
9
10    beratBadan = bmi * (tinggiBadan * tinggiBadan)
11
12    fmt.Printf("%.0f\n", beratBadan)
13 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\alproo\go\laprak 4> go run soal2.go
22.85 1.75
70
PS C:\alproo\go\laprak 4> go run soal2.go
23.43 1.6
60

Na

File Edit View A A v u s

Nama : Primatama Sigalingging
NIM: 109082500076

Ln 2, Col 18 | 47 character Plain t | 100% Wind UTF-8 X

powershell
powershell
powershell
powershell
go

Deskripsi program

- Tujuan Program:
Program ini bertujuan untuk menghitung berat badan seseorang berdasarkan nilai Body Mass Index (BMI) dan tinggi badan (dalam meter) yang dimasukkan oleh pengguna. Dengan program ini, pengguna dapat mengetahui berapa berat badan ideal (atau aktual) yang sesuai dengan nilai BMI tertentu.
- Program ini merupakan kebalikan dari program penghitungan BMI pada umumnya. Biasanya, kita menghitung BMI dari berat dan tinggi badan. Namun kali ini, program justru menghitung berat badan berdasarkan nilai BMI dan tinggi badan.

Cara kerjanya sangat sederhana.

Saat dijalankan, pengguna akan diminta untuk memasukkan dua nilai secara berurutan:

1. **Nilai BMI (Body Mass Index)**
2. **Nilai tinggi badan (meter)**

Sebagai contoh, pengguna bisa mengetik: **22.0 1.75**

Artinya, seseorang memiliki nilai BMI 22 dan tinggi badan 1,75 meter.

Program kemudian akan menghitung berapa berat badan yang sesuai dengan nilai BMI tersebut menggunakan rumus:

$$\text{Berat Badan} = \text{BMI} \times (\text{Tinggi Badan})^2$$

Rumus ini berasal dari rumus dasar BMI:

Artinya, seseorang memiliki nilai BMI 22 dan tinggi badan 1,75 meter.

Program kemudian akan menghitung berapa berat badan yang sesuai dengan nilai BMI tersebut menggunakan rumus:

$$\text{Berat Badan} = \text{BMI} \times (\text{Tinggi Badan})^2$$

Rumus ini berasal dari rumus dasar BMI:

$$\text{BMI} = \frac{\text{Berat Badan}}{(\text{Tinggi Badan})^2}$$

yang kemudian dibalik untuk mencari berat badan.

Dalam kode Go, perhitungannya ditulis sebagai:

$$\text{beratBadan} = \text{bmi} * (\text{tinggiBadan} * \text{tinggiBadan})$$

Hasilnya kemudian ditampilkan dalam satuan kilogram (kg) dengan format tanpa angka desimal, agar lebih mudah dibaca:

fmt.Printf("%.0f\n", beratBadan)

- Kesimpulan:
Program ini membantu pengguna untuk menentukan berat badan berdasarkan nilai BMI dan tinggi badan secara cepat dan akurat. Dengan hanya dua input sederhana, program dapat menghitung berat badan yang sesuai dengan indeks massa tubuh tertentu.

Program ini sangat berguna untuk:

- Menghitung berat badan ideal berdasarkan target BMI
- Membantu perencanaan program diet atau kebugaran
- Memberikan ilustrasi matematis hubungan antara tinggi badan, berat badan, dan BMI
-

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)

    ab := math.Sqrt(math.Pow(x2-x1, 2) + math.Pow(y2-y1, 2))
    bc := math.Sqrt(math.Pow(x3-x2, 2) + math.Pow(y3-y2, 2))
    ca := math.Sqrt(math.Pow(x1-x3, 2) + math.Pow(y1-y3, 2))

    max := ab
    if bc > max {
        max = bc
    }
    if ca > max {
        max = ca
    }

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

Screenshoot program

```
8 func main() {
9
10
11     fmt.Scan(&x1, &y1)
12     fmt.Scan(&x2, &y2)
13     fmt.Scan(&x3, &y3)
14
15     ab := math.Sqrt(math.Pow(x2-x1, 2) + math.Pow(y2-y1, 2))
16     bc := math.Sqrt(math.Pow(x3-x2, 2) + math.Pow(y3-y2, 2))
17     ca := math.Sqrt(math.Pow(x1-x3, 2) + math.Pow(y1-y3, 2))
18
19     max := ab
20     if bc > max {
21         max = bc
22     }
23     if ca > max {
24         max = ca
25     }
26
27     fmt.Printf("%.2f\n", max)
28 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\alproo\go\laprak 4> go run soal3.go
1.0 1.0
4.0 1.0
2.0 5.0
4.47
```

Deskripsi program

➤ Tujuan Program:

Program ini bertujuan untuk menghitung panjang sisi-sisi segitiga yang dibentuk oleh tiga titik koordinat (A, B, dan C) pada bidang kartesius dua dimensi, kemudian menentukan sisi terpanjang di antara ketiganya.

Program ini menggunakan rumus jarak antar dua titik (Teorema Pythagoras) untuk menghitung panjang tiap sisi segitiga.

- Bayangkan kita memiliki tiga titik pada peta misalnya titik A, B, dan C yang masing-masing punya koordinat (x, y). Tiga titik ini akan membentuk sebuah segitiga, dan kita ingin tahu sisi mana yang paling panjang di antara AB, BC, dan CA.

Program ini melakukan hal tersebut secara otomatis.

Saat dijalankan, pengguna akan diminta memasukkan tiga pasang koordinat, yaitu:

1. Titik A → x1 y1
2. Titik B → x2 y2
3. Titik C → x3 y3

Sebagai contoh, input bisa seperti ini:

1.0 1.0

4.0 1.0

4.0 5.0

Artinya:

- Titik A berada di (1, 1)
- Titik B berada di (4, 1)
- Titik C berada di (4, 5)

Setelah menerima data, program akan menghitung panjang ketiga sisi segitiga menggunakan rumus jarak antar dua titik

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Dengan rumus itu, program menghitung:

- **Panjang sisi AB**
- **Panjang sisi BC**
- **Panjang sisi CA**

Dalam kode Go, rumus tersebut ditulis menggunakan fungsi **math.Sqrt()** dan **math.Pow()** dari pustaka **math**:

```
ab := math.Sqrt(math.Pow(x2-x1, 2) + math.Pow(y2-y1, 2))
```

```
bc := math.Sqrt(math.Pow(x3-x2, 2) + math.Pow(y3-y2, 2))
```

```
ca := math.Sqrt(math.Pow(x1-x3, 2) + math.Pow(y1-y3, 2))
```

Setelah ketiga panjang sisi diketahui, program membandingkan semuanya untuk mencari yang terpanjang:

```
max := ab
```

```
if bc > max {
```

```
    max = bc
```

```
}
```

```
if ca > max {
```

```
    max = ca
```

```
}
```

Kemudian hasilnya ditampilkan dengan dua angka di belakang koma agar rapi dan mudah dibaca:

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

➤ **Kesimpulan:**

Program ini memungkinkan pengguna untuk menentukan sisi terpanjang dari sebuah segitiga berdasarkan tiga titik koordinat secara cepat dan akurat.

Dengan memanfaatkan rumus jarak antar titik dan fungsi matematika dari pustaka **math**, program ini bisa digunakan untuk:

- Analisis geometri sederhana
- Perhitungan jarak antar titik pada peta
- Latihan dasar penggunaan pustaka matematika di bahasa Go