

LAPORAN PRAKTIKUM
Algoritma Pemrograman

MODUL 11
SWITCH-CASE



Disusun oleh:
GAMALIEL ALBERT NATANAEL SIMANJUNTAK
109082500067
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 jam12, jam24 int
    var label string
    fmt.Scan(&jam24)
    switch {
    case jam24 == 0:
        jam12 = 12
        label = "AM"
    case jam24 < 12:
        jam12 = jam24
        label = "AM"
    case jam24 == 12:
        jam12 = 12
        label = "PM"
    case jam24 > 12:
        jam12 = jam24 - 12
        label = "PM"
    }
    fmt.Println(jam12, label)
}
```

Screenshoot program

```
1 package main
2 import "fmt"
3 func main() {
4     var jam12, jam24 int
5     var label string
6     fmt.Scan(&jam24)
7     switch {
8     case jam24 == 0:
9         jam12 = 12
10        label = "AM"
11    case jam24 < 12:
12        jam12 = jam24
13        label = "AM"
14    case jam24 == 12:
15        jam12 = 12
16        label = "PM"
17    }
18    fmt.Println(jam12, label)
19 }
```

```
PS D:\Laprak Modul 11 Switch-Case> go run "d:\Laprak Modul 11 Switch-Case\guided1.go"
0
12 AM
PS D:\Laprak Modul 11 Switch-Case> go run "d:\Laprak Modul 11 Switch-Case\guided1.go"
13
12 PM
PS D:\Laprak Modul 11 Switch-Case> go run "d:\Laprak Modul 11 Switch-Case\guided1.go"
12
12 PM
PS D:\Laprak Modul 11 Switch-Case>
```

Deskripsi program

Program ini adalah alat konversi waktu sederhana. Tujuannya adalah mengubah format waktu 24-jam (input integer) menjadi format 12-jam (AM/PM).

Analisis Alur Logika (Step-by-Step)

1. Deklarasi Variabel:

- **jam12, jam24:** Variabel integer untuk menyimpan nilai jam (input dan output).
- **label:** Variabel string untuk menyimpan penanda waktu ("AM" atau "PM").

2. Input (fmt.Scan):

- Program menunggu pengguna memasukkan sebuah angka bulat (merepresentasikan jam dalam format 24 jam, misalnya: 0, 13, 23). Nilai ini disimpan di variabel jam24.

3. Logika Konversi (switch):

Program menggunakan struktur kendali switch tanpa ekspresi (setara dengan rangkaian if-else) untuk menentukan logika konversi berdasarkan nilai jam24:

- **Kasus Tengah Malam (jam24 == 0):**
 - Jika input adalah 0 (00:00), jam diubah menjadi 12 dan label menjadi "AM".
- **Kasus Pagi (jam24 < 12):**
 - Jika input antara 1 sampai 11, jam tetap sama dan label menjadi "AM".
- **Kasus Siang (jam24 == 12):**
 - Jika input adalah 12 (12:00 siang), jam tetap 12 tetapi label menjadi "PM".
- **Kasus Sore/Malam (jam24 > 12):**

- Jika input lebih besar dari 12 (misalnya 13, 14, dst.), program akan mengurangi nilai tersebut dengan 12 (Contoh: $14 - 12 = 2$) dan label menjadi "PM".

4. Output (fmt.Println):

- Program mencetak hasil akhir berupa jam yang sudah dikonversi diikuti oleh labelnya (Contoh output: 2 PM).

2. Guided 2

Source Code

```
package main

import "fmt"

func main() {
    var tanaman string
    fmt.Scan(&tanaman)

    switch tanaman {
    case "nepenthes":
        fmt.Println("Termasuk Tanaman Karnivora \nAsli Indonesia")
    case "venus":
        fmt.Println("Termasuk Tanaman Karnivora \nBukan Asli Indonesia")
    default:
        fmt.Println("Tidak termasuk Tanaman Karnivora")
    }
}
```

Screenshoot program

```
1 package main
2 import "fmt"
3 func main() {
4     var tanaman string
5     fmt.Scan(&tanaman)
6     switch tanaman {
7     case "nepenthes":
8         fmt.Println("Termasuk Tanaman Karnivora \nAsli Indonesia")
9     case "venus":
10        fmt.Println("Termasuk Tanaman Karnivora \nBukan Asli Indonesia")
11    default:
12        fmt.Println("Tidak termasuk Tanaman Karnivora")
13    }
14 }
```

12 PM
PS D:\Laparak Modul 11 Switch-Case> go run "d:\Laparak Modul 11 Switch-Case\guided2.go"
nepenthes
Termasuk Tanaman Karnivora
Asli Indonesia
PS D:\Laparak Modul 11 Switch-Case> go run "d:\Laparak Modul 11 Switch-Case\guided2.go"
venus
Termasuk Tanaman Karnivora
Bukan Asli Indonesia
PS D:\Laparak Modul 11 Switch-Case> go run "d:\Laparak Modul 11 Switch-Case\guided2.go"
karedok
Tidak termasuk Tanaman Karnivora
PS D:\Laparak Modul 11 Switch-Case>

NIM:100082500067
KELAS:SIIF-13-04
NAMA:GABRIEL ALBERT NATANAEL SIMANJUNTAK

Deskripsi program

Program ini berfungsi sebagai kamus sederhana untuk mengidentifikasi apakah sebuah tanaman termasuk "Tanaman Karnivora" dan asal-usulnya, berdasarkan nama tanaman yang diinputkan pengguna.

Analisis Alur Logika (Step-by-Step)

1. Deklarasi Variabel:
 - `var tanaman string`: Menyiapkan wadah untuk menyimpan teks (nama tanaman).
2. Input (`fmt.Scan`):
 - Program menunggu pengguna mengetikkan satu kata.
 - Catatan Penting: `fmt.Scan` hanya membaca teks sampai spasi pertama. Jadi, jika Anda mengetik "venus flytrap", hanya kata "venus" yang akan tersimpan.
3. Logika Percabangan (`switch tanaman`): Berbeda dengan contoh sebelumnya yang menggunakan switch tanpa kondisi (seperti rangkaian if-else), kali ini program menggunakan switch pada variabel. Program akan membandingkan isi variabel tanaman secara langsung dengan nilai di setiap case.
 - Kasus 1 (case "nepenthes"):
 - Jika input persis "nepenthes" (huruf kecil semua), program mencetak bahwa ini tanaman karnivora dan Asli Indonesia.
 - Karakter `\n` di dalam `fmt.Println` berfungsi untuk membuat baris baru (Enter), sehingga output "Asli Indonesia" akan muncul di bawahnya.
 - Kasus 2 (case "venus"):
 - Jika input persis "venus", program mencetak bahwa ini tanaman karnivora tetapi Bukan Asli Indonesia (Venus Flytrap berasal dari Amerika Serikat).

- **Default (default):**
 - **Jika input bukan "nepenthes" dan bukan "venus" (contoh: "mawar", "melati", atau bahkan "Nepenthes" dengan huruf kapital), program akan masuk ke sini dan mencetak "Tidak termasuk Tanaman Karnivora".**

3. Guided 3

Source Code

```
package main

import "fmt"

func main() {

    var kendaraan string

    var jam, tarif int

    fmt.Scan(&kendaraan, &jam)

    switch kendaraan {

    case "Motor":

        switch jam {

        case 1, 2:

            tarif = 7000

        default:

            tarif = 9000

        }

    case "Mobil":

        switch jam {

        case 1, 2:

            tarif = 15000

        default:

            tarif = 20000

        }

    case "Truk":

        switch jam {

        case 1, 2:

            tarif = 25000
```

```

        default:

            tarif = 35000

        }

    default:

        fmt.Println("Jenis kendaraan atau durasi parkir
tidak valid")

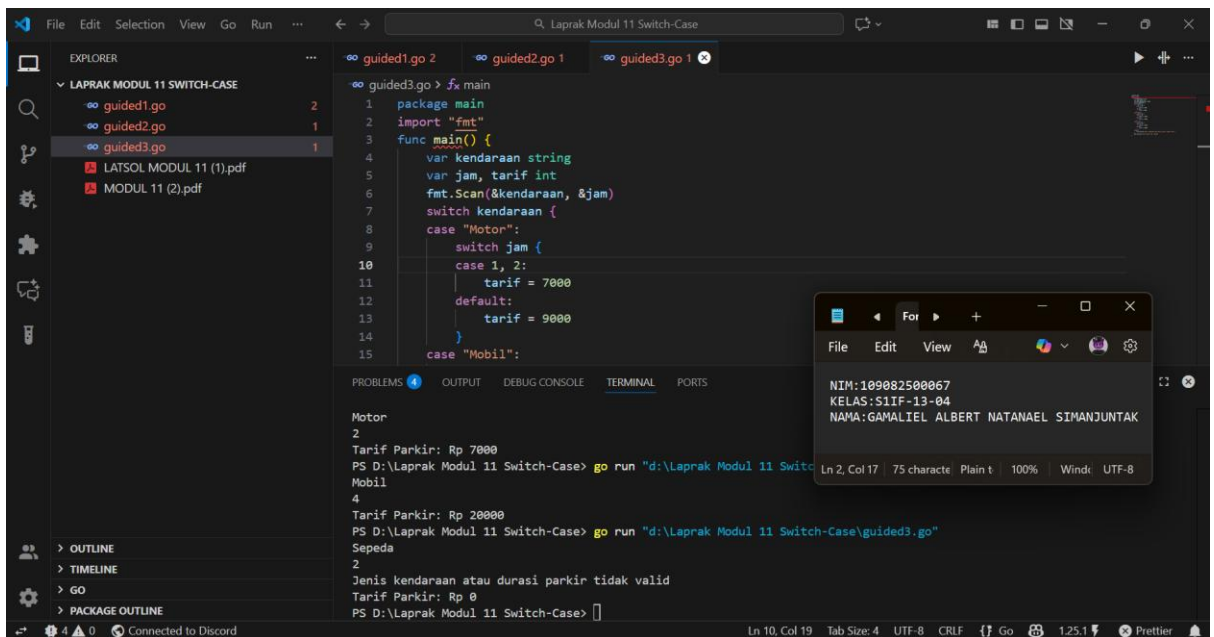
    }

    fmt.Println("Tarif Parkir: Rp", tarif)

}

```

Screenshoot program



Deskripsi program

Program ini adalah kalkulator tarif parkir sederhana. Program menentukan biaya parkir berdasarkan dua faktor: jenis kendaraan (Motor, Mobil, Truk) dan durasi parkir (jam).

Fitur utama yang ditunjukkan dalam kode ini adalah Nested Switch (Percabangan Bersarang), yaitu adanya struktur switch di dalam switch yang lain.

Analisis Alur Logika (Step-by-Step)

1. Deklarasi Variabel:

- **kendaraan (string):** Menyimpan jenis kendaraan.
- **jam (int):** Menyimpan lama parkir dalam jam.
- **tarif (int):** Menyimpan hasil perhitungan biaya.

2. Input (fmt.Scan):

- `fmt.Scan(&kendaraan, &jam)`: Program menunggu pengguna memasukkan dua data sekaligus (dipisahkan spasi).
 - Contoh input: Motor 2 (Kendaraan = "Motor", Jam = 2).
3. Logika Percabangan Bersarang (Nested Switch):
- Switch Luar (Berdasarkan Kendaraan): Program pertama kali mengecek variabel kendaraan. Perhatikan bahwa pengecekan ini case-sensitive (harus diawali huruf kapital: "Motor", "Mobil", "Truk").
 - Switch Dalam (Berdasarkan Jam): Setelah jenis kendaraan ditemukan, program masuk ke logika internal untuk mengecek durasi jam.
 - Case 1, 2: Jika parkir selama 1 atau 2 jam, tarif lebih murah (tarif dasar).
 - Default: Jika parkir selain 1 atau 2 jam (misalnya 3 jam ke atas), tarif lebih mahal (tarif maksimal/progresif).
- Rincian Tarif:
- Motor: 1-2 jam = Rp 7.000 | >2 jam = Rp 9.000
 - Mobil: 1-2 jam = Rp 15.000 | >2 jam = Rp 20.000
 - Truk: 1-2 jam = Rp 25.000 | >2 jam = Rp 35.000
4. Default (Error Handling):
- Jika input kendaraan tidak sesuai (misal: "Sepeda" atau "motor" dengan huruf kecil), program mencetak: *"Jenis kendaraan atau durasi parkir tidak valid"*.
5. Output Akhir:
- Program mencetak Tarif Parkir: Rp [nilai tarif].

TUGAS

1. Tugas 1

Source code

```
package main

import "fmt"

func main() {
    var ph float64
    fmt.Print("Masukan: ")
    fmt.Scan(&ph)

    if ph < 0 || ph > 14 {
        fmt.Println("Nilai pH tidak valid. Nilai pH harus antara 0 dan 14.")
    } else if ph >= 6.5 && ph <= 8.6 {
        fmt.Println("Air layak minum")
    }
}
```



```

} else {

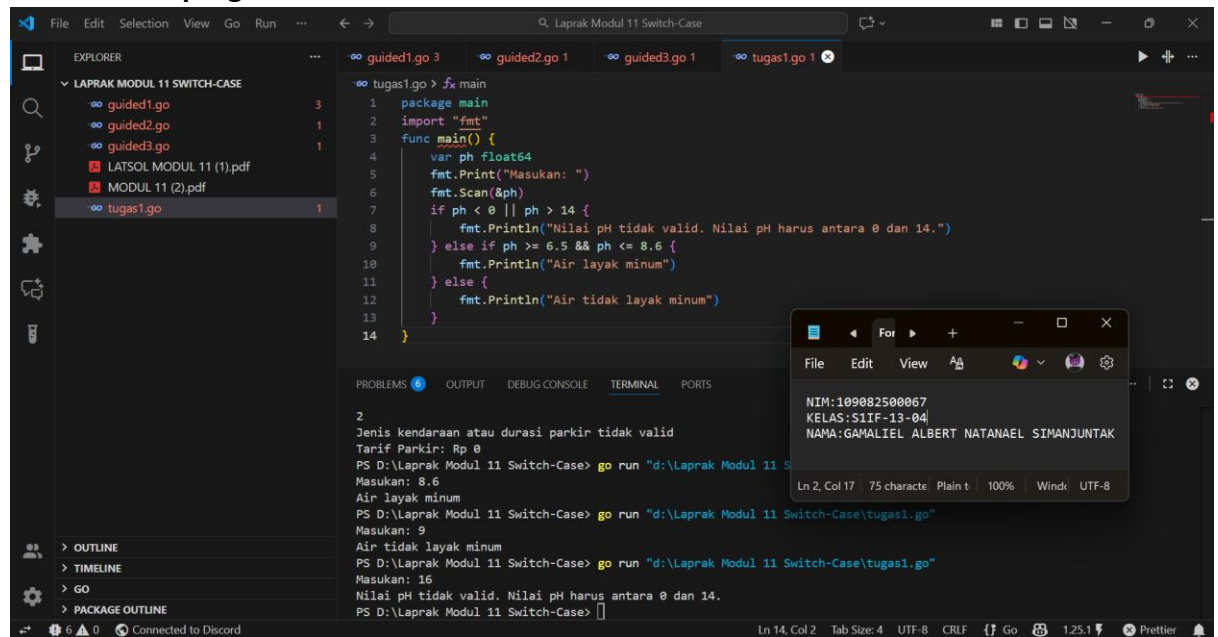
fmt.Println("Air tidak layak minum")

}

}

```

Screenshoot program



Deskripsi program

Program ini berfungsi untuk mengevaluasi kualitas air berdasarkan nilai pH yang diinputkan oleh pengguna. Program akan menentukan apakah air tersebut Layak Minum, Tidak Layak Minum, atau jika inputnya Tidak Valid.

Penjelasan Langkah demi Langkah:

- Deklarasi Variabel (var ph float64)**
 - Program menyiapkan sebuah wadah memori (variabel) bernama ph.
 - Tipe data yang digunakan adalah float64 (bilangan desimal) karena nilai pH seringkali memiliki koma (contoh: 8.6), bukan bilangan bulat saja.
- Input Pengguna (fmt.Scan)**
 - Program menampilkan teks "Masukan: " ke layar.
 - Perintah `fmt.Scan(&ph)` menunggu pengguna mengetik angka dan menyimpannya ke dalam variabel ph tadi.
- Logika Pengecekan (Percabangan If-Else)** Program memeriksa nilai ph melalui tiga tahap filter logika:
 - Filter 1: Cek Validitas (Validasi Input)**
 - `if ph < 0 || ph > 14`
 - Program mengecek apakah angka di bawah 0 ATAU (||) di atas 14.

- Jika *salah satu* benar, artinya input tidak masuk akal secara kimia. Program mencetak pesan error dan selesai.
- Filter 2: Cek Kelayakan (Range Aman)
 - else if `ph >= 6.5 && ph <= 8.6`
 - Jika lolos filter pertama (angka 0-14), program mengecek apakah angka tersebut berada di antara 6.5 DAN 8.6.
 - Operator DAN (&&) memastikan nilai harus memenuhi kedua batas tersebut (tidak boleh kurang dari 6.5 dan tidak boleh lebih dari 8.6). Jika ya, program mencetak "Air layak minum".
- Filter 3: Sisa Kondisi (Tidak Layak)
 - else
 - Jika angka valid (0-14) tetapi tidak masuk dalam range aman (misalnya: 4.0 atau 9.0), maka otomatis masuk ke kategori ini.
 - Program mencetak "Air tidak layak minum".

Kesimpulan Logika: Program memprioritaskan validasi error terlebih dahulu (angka di luar 0-14), baru kemudian mengecek spesifikasi kelayakan (6.5 - 8.6), dan sisanya dianggap tidak layak.

2. Tugas 2

Source code

```
package main

import (
    "fmt"
    "strings"
)

func main() {
    var jenisKendaraan string
    var durasiJam int

    fmt.Print("Masukan jenis kendaraan dan durasi: ")
    fmt.Scan(&jenisKendaraan, &durasiJam)
    jenisKendaraan = strings.ToLower(jenisKendaraan)
    if durasiJam < 1 {
        durasiJam = 1
    }

    var tarifPerJam int
```

```

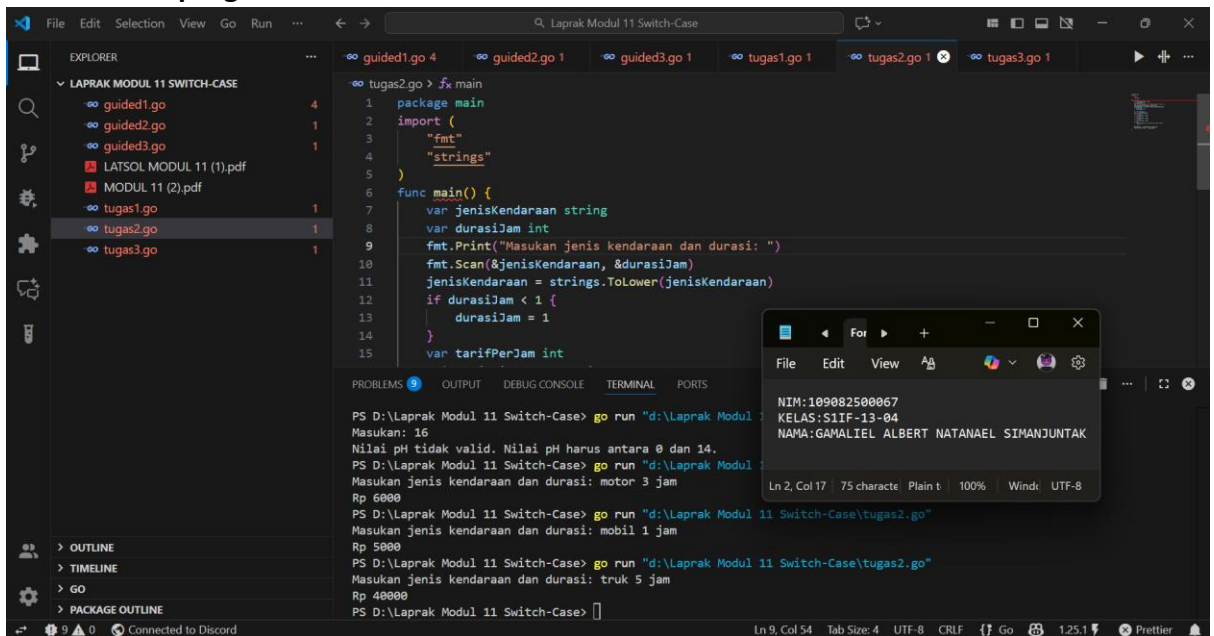
switch jenisKendaraan {
case "motor":
    tarifPerJam = 2000
case "mobil":
    tarifPerJam = 5000
case "truk":
    tarifPerJam = 8000
default:
    fmt.Println("Jenis kendaraan tidak valid")
    return
}

totalBiaya := tarifPerJam * durasiJam

fmt.Printf("Rp %d\n", totalBiaya)
}

```

Screenshoot program



Deskripsi program

Program ini bertujuan untuk menghitung total biaya parkir berdasarkan dua input: jenis kendaraan (motor, mobil, truk) dan lama parkir (dalam jam).

Penjelasan Langkah demi Langkah:

1. Deklarasi Variabel

- `var jenisKendaraan string`: Menyiapkan wadah untuk teks jenis kendaraan.
 - `var durasiJam int`: Menyiapkan wadah untuk angka durasi parkir (bilangan bulat).
2. Input Pengguna
- `fmt.Scan(&jenisKendaraan, &durasiJam)`: Program menunggu pengguna memasukkan data. Pengguna bisa mengetik sekaligus (contoh: motor 3) lalu menekan Enter.
3. Normalisasi Teks (`strings.ToLower`)
- `jenisKendaraan = strings.ToLower(jenisKendaraan)`
 - Fungsinya: Mengubah semua huruf input menjadi huruf kecil.
 - Tujuannya: Agar program tidak bingung jika pengguna mengetik dengan huruf kapital (contoh: "Motor", "MOTOR", atau "MoToR" akan dianggap sama sebagai "motor"). Ini membuat program lebih *robust* (tahan kesalahan input sepele).
4. Validasi Durasi Minimum
- `if durasiJam < 1 { durasiJam = 1 }`
 - Sesuai aturan soal, jika parkir kurang dari 1 jam (misal 0 jam atau angka negatif), durasi otomatis dianggap 1 jam. Ini mencegah perkalian dengan angka 0 atau negatif.
5. Penentuan Tarif (Switch Case) Program mengecek isi variabel `jenisKendaraan` untuk menentukan harga per jam (`tarifPerJam`):
- Jika "motor": Tarif diisi 2000.
 - Jika "mobil": Tarif diisi 5000.
 - Jika "truk": Tarif diisi 8000.
 - Default: Jika input bukan ketiganya (misal: "sepeda"), program mencetak error "Jenis kendaraan tidak valid" dan berhenti (return).
6. Kalkulasi dan Output
- `totalBiaya := tarifPerJam * durasiJam`: Menghitung total bayar.
 - `fmt.Printf("Rp %d\n", totalBiaya)`: Menampilkan hasil akhir ke layar dengan format "Rp [angka]".

Ringkasan Logika: Program menerima input -> menyamakan format huruf -> memastikan durasi minimal 1 jam -> memilih harga sesuai kendaraan -> mengalikan harga dengan durasi -> menampilkan hasil.

3. Tugas 3

Source code

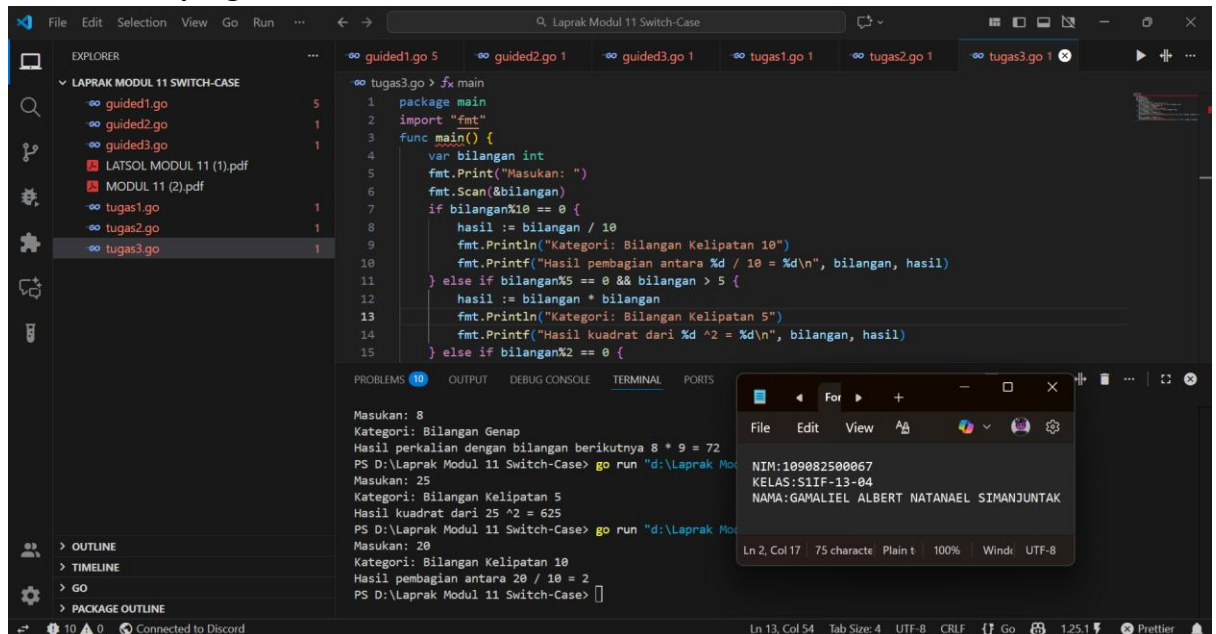
```
package main
import "fmt"
func main() {
    var bilangan int
    fmt.Print("Masukan: ")
    fmt.Scan(&bilangan)
    if bilangan%10 == 0 {
```

```

        hasil := bilangan / 10
        fmt.Println("Kategori: Bilangan Kelipatan 10")
        fmt.Printf("Hasil pembagian antara %d / 10 = %d\n",
bilangan, hasil)
    } else if bilangan%5 == 0 && bilangan > 5 {
        hasil := bilangan * bilangan
        fmt.Println("Kategori: Bilangan Kelipatan 5")
        fmt.Printf("Hasil kuadrat dari %d ^2 = %d\n",
bilangan, hasil)
    } else if bilangan%2 == 0 {
        hasil := bilangan * (bilangan + 1)
        fmt.Println("Kategori: Bilangan Genap")
        fmt.Printf("Hasil perkalian dengan bilangan
berikutnya %d * %d = %d\n", bilangan, bilangan+1, hasil)
    } else {
        hasil := bilangan + (bilangan + 1)
        fmt.Println("Kategori: Bilangan Ganjil")
        fmt.Printf("Hasil penjumlahan dengan bilangan
berikutnya %d + %d = %d\n", bilangan, bilangan+1, hasil)
    }
}

```

Screenshoot program



Deskripsi program

Program ini adalah implementasi dari logika if-else if-else berjenjang (ladder). Kunci keberhasilan program ini terletak pada urutan prioritas pengecekan.

Program menerima satu angka bulat, kemudian mengklasifikasikannya ke dalam salah satu dari 4 kategori (Kelipatan 10, Kelipatan 5, Genap, atau Ganjil) dan melakukan operasi matematika yang berbeda untuk setiap kategori.

Penjelasan Langkah demi Langkah

1. Input Data

- Program meminta pengguna memasukkan angka integer (int).
- Angka ini disimpan dalam variabel bilangan.

2. Filter Logika (Prioritas Tertinggi ke Terendah) Program memeriksa angka tersebut melewati serangkaian gerbang logika. Begitu satu syarat terpenuhi, program akan mengeksekusi perintah di dalamnya dan mengabaikan sisa kode di bawahnya.

- **Prioritas 1: Cek Kelipatan 10 (if bilangan % 10 == 0)**
 - Logika: Apakah angka habis dibagi 10? (Contoh: 10, 20, 30).
 - Aksi: Jika Ya, program membagi angka tersebut dengan 10 (bilangan / 10).
 - Penting: Pengecekan ini harus ditaruh paling atas. Jika tidak, angka 20 akan tertangkap oleh logika "Genap" atau "Kelipatan 5" duluan, yang akan menghasilkan output yang salah.
- **Prioritas 2: Cek Kelipatan 5 (else if bilangan % 5 == 0 && bilangan > 5)**
 - Logika: Apakah angka habis dibagi 5 DAN angkanya lebih besar dari 5?
 - Kenapa ada && bilangan > 5? Ini adalah trik soal. Pada tabel contoh soal, angka 5 dikategorikan sebagai "Ganjil", bukan "Kelipatan 5". Sedangkan angka 25 dikategorikan "Kelipatan 5". Jadi, kita harus mengecualikan angka 5 di sini agar dia bisa lolos ke pengecekan Ganjil nanti.
 - Aksi: Mengkuadratkan angka tersebut (bilangan * bilangan).
- **Prioritas 3: Cek Genap (else if bilangan % 2 == 0)**
 - Logika: Apakah angka habis dibagi 2?
 - Aksi: Mengalikan angka tersebut dengan angka setelahnya (bilangan * (bilangan + 1)).
 - Catatan: Angka seperti 20 tidak akan sampai ke sini karena sudah ditangkap duluan di Prioritas 1.
- **Prioritas 4: Cek Ganjil (else)**
 - Logika: Jika angka tidak memenuhi ketiga syarat di atas (bukan kelipatan 10, bukan kelipatan 5 > 5, dan bukan genap). Ini menangkap angka ganjil biasa (1, 3, 5, 7, 9, dst).
 - Aksi: Menjumlahkan angka tersebut dengan angka setelahnya (bilangan + (bilangan + 1)).