

**LAPORAN PRAKTIKUM
ALGORITMA PEMROGRAMAN 2
MODUL 2
REVIEW STRUKTUR KONTROL**



Oleh:

ADITHANA DHARMA PUTRA

2311102207

IF – 11- 02

**S1 TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
2024**

I. DASAR TEORI

a) Struktur Program Go

Dalam kerangka program yang ditulis dalam bahasa pemrograman Go, program utama selalu mempunyai dua komponen berikut:

- Package main merupakan penanda bahwa file ini berisi program utama.
- func main(berisi kode utama dari sebuah program Go. Komentar, bukan bagian dari kode program, dan dapat ditulis di mana saja di dalam program:
- Satu baris teks yang diawali dengan garis miring ganda ("/*") s.d. akhir baris, atau. Beberapa baris teks yang dimulai dengan pasangan karakter "/*" dan diakhiri dengan "*/"

b) Koding, Kompilasi, dan Eksekusi Go Koding

- Tidak berbeda dengan penulisan program sumber dalam bahasa lain, program Go harus dibuat menggunakan penyunting teks dan disimpan dalam format teks, bukan dalam format dokumen (doc, docx, atau lainnya)
- Setiap program go disimpan dalam file teks dengan ekstensi .go, dengan nama bebas. Sebaiknya nama file adalah nama untuk program tersebut,
- Setiap satu program lengkap Go disimpan dalam satu folder tersendiri. Nama folder merupakan nama program tersebut, Karena itu secara prinsip, satu program Go dapat dipecah dalam beberapa file dengan ekstensi *.go selama disimpan dalam folder yang sama:

c) Kompilasi

Beberapa bahasa pemrograman dirancang untuk diimplementasikan sebagai interpreter dan lainnya sebagai kompilator. Interpreter akan membaca setiap baris instruksi dan kemudian langsung mengeksekusinya, dengan hanya sedikit pemeriksaan apakah penulisan keseluruhan program sudah benar atau belum. Kompilator akan memeriksa keseluruhan program sumber dan kemudian mengubahnya menjadi program eksekutabel, sehingga konsistensi penulisan (seperti penggunaan tipe data) sudah diperiksa sebelum eksekusi. Selain itu karena program dibuat menjadi eksekutabel lebih dahulu, proses optimasi dapat dilakukan sehingga program menjadi sangat efisien.

Catatan : Semua proses terkait bahasa go dilakukan melalui utilitas go. Beberapa opsi dengan utilitas go : Go build : mengkompilasi program sumber yang ada dalam folder menjadi sebuah program

- Go build file.go : mengkompilasi program sumber file.go saja.
- Go fmt : membaca semua program sumber dalam folder dan mereformat penulisannya agar sesuai dengan standar penulisan program sumber go.
- Go clean : membersihkan file-file dalam folder sehingga tersisa program sumber nya saja

d) Variabel

Variabel adalah nama dari suatu lokasi di memori, yang data dengan tipe tertentu dapat disimpan. ' Nama variabel dimulai dengan huruf dan dapat diikuti dengan sejumlah huruf, angka, atau garisbawah

- Tipe data yang umum tersedia adalah integer, real, boolean, karakter, dan string. Lihat tabel berikut ini untuk variasi tipe data yang disediakan dalam bahasa Go.
- Nilai data yang tersimpan dalam variabel dapat diperoleh dengan menyebutkan langsung nama variabelnya. Contoh: Menyebutkan nama found akan mengambil nilai tersimpan dalam memori untuk variabel found, pastinya.
- Informasi alamat atau lokasi dari variabel dapat diperoleh dengan menambahkan prefiks & di depan nama variabel tersebut. Contoh: &found akan mendapatkan alamat memori untuk menyimpan data pada found.
- Jika variabel berisi alamat memori, prefiks * pada variabel tersebut akan memberikan nilai yang tersimpan dalam memori yang lokasinya disimpan dalam variabel tersebut.

e) Struktur Kontrol Perulangan

Go hanya mempunyai kata kunci for untuk semua jenis perulangan yang kita pelajari dalam notasi algoritma. Dua bentuk yang kita gunakan di sini adalah struktur while-loop dan repeat-until. Bentuk While-Loop Bentuk while-loop memastikan setiap kali memasuki loop, ada kondisi yang harus terpenuhi (benar>true). Inijuga berarti saat keluar dari loop, maka nilai kondisi tersebut pasti salah>false! Bentuk Repeat-Until Bentuk repeat-until di perulangan dilakukan terus menerus sampai kondisi keluar terpenuhi. Artinya selama kondisi belum terpenuhi (salah>false) maka perulangan akan

terus dilakukan. Pada saat keluar dari loop maka nilai kondisi pasti benar/true!

f) Struktur Kontrol Percabangan

Untuk analisa kasus, bahasa Go mendukung dua bentuk percabangan, yaitu if-else dan switch-case. 1) Bentuk If-Else Berikut ini bentuk-bentuk ifelse yang mungkin dilakukan dalam bahasa Go. Semua bentuk di bawah merupakan satu instruksi if-else-endif saja (hanya satu endif). Bentuk if-else yang bersarang (dengan beberapa endif) dapat dibentuk dengan komposisi beberapa if-else-endif tersebut

Bentuk Switch-Case Dalam bahasa Co ada dua variasi bentuk switchcase. Bentuk yang biasa digunakan adalah ekspresi ditulis pada perintah switch dan nilai ditulis dalam setiap label case-nya. Bentuk yang kedua mempunyai switch tanpa ekspresi, tetapi setiap case boleh berisi ekspresi boolean. Tentunya bentuk yang kedua lebih bersifat umum, dan merupakan penyederhanaan bentuk

II. GUIDED

1. Guided 1

Source Code

```
package main
import "fmt"

func main() {
//deklarasi variabel
var (
    satu, dua, tiga string
    temp string
)

//meminta input
fmt.Print("Masukan input string: ")
fmt.Scanln(&satu)

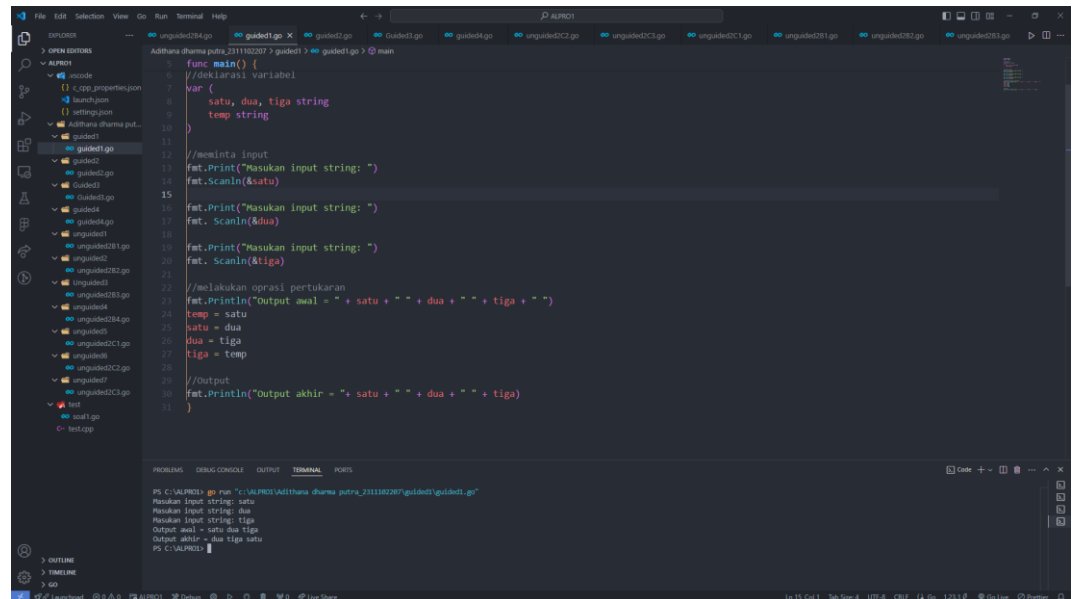
fmt.Print("Masukan input string: ")
fmt. Scanln(&dua)

fmt.Print("Masukan input string: ")
fmt. Scanln(&tiga)

//melakukan oprasi pertukaran
fmt.Println("Output awal = " + satu + " " + dua + " " + tiga + " ")
temp = satu
satu = dua
dua = tiga
tiga = temp

//Output
fmt.Println("Output akhir = "+ satu + " " + dua + " " + tiga)
}
```

Screenshot



```
1 func main() {
2     //deklarasi variabel
3     var (
4         satu, dua, tiga string
5         temp string
6     )
7
8     //meminta input
9     fmt.Print("Masukan input string: ")
10    fmt.Scanln(&satu)
11
12    //meminta input
13    fmt.Print("Masukan input string: ")
14    fmt.Scanln(&dua)
15
16    //meminta input
17    fmt.Print("Masukan input string: ")
18    fmt.Scanln(&tiga)
19
20    //melakukan operasi pertukaran
21    fmt.Println("Output awal = " + satu + " + dua + " + tiga + " ")
22    temp = satu
23    satu = dua
24    dua = tiga
25    tiga = temp
26
27    //Output
28    fmt.Println("Output akhir = " + satu + " + dua + " + tiga + " ")
29 }
```

PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL PORTS

```
PS C:\VALPROG> go run "C:\VALPROG\Addithana putra_23111802807\guided1\guided1.go"
Masukan input string: satu
Masukan input string: dua
Masukan input string: tiga
Output awal = satu dua tiga
Output akhir = dua tiga satu
PS C:\VALPROG>
```

Deskripsi:

Program dimulai dengan mendeklarasikan variabel satu, dua, tiga, dan temp untuk menyimpan nilai string. Setelah itu, program meminta pengguna untuk memasukkan tiga string yang disimpan dalam variabel satu, dua, dan tiga. Nilai awal dari ketiga string tersebut kemudian ditampilkan. Selanjutnya, program melakukan operasi pertukaran di mana nilai satu disimpan sementara di temp, satu diisi dengan nilai dua, dua diisi dengan nilai tiga, dan tiga diisi dengan nilai temp. Akhirnya, program menampilkan nilai akhir dari ketiga string setelah pertukaran.

2. Guided 2

Source Code

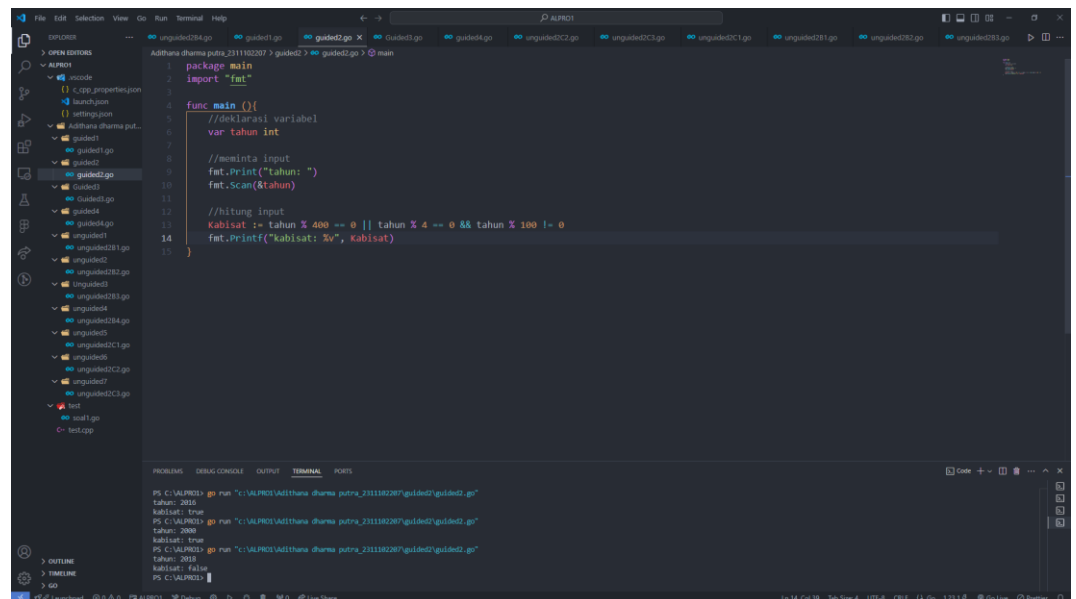
```
package main
import "fmt"

func main () {
    //deklarasi variabel
    var tahun int

    //meminta input
    fmt.Print("tahun: ")
    fmt.Scan(&tahun)

    //hitung input
    Kabisat := tahun % 400 == 0 || tahun % 4 == 0
    && tahun % 100 != 0
    fmt.Printf("kabisat: %v", Kabisat)
}
```

Screenshot



Deskripsi:

Kode ini adalah program Go yang menentukan apakah suatu tahun adalah tahun kabisat atau bukan. Program dimulai dengan mendeklarasikan variabel tahun untuk menyimpan input dari pengguna. Setelah itu, program meminta pengguna untuk memasukkan tahun yang ingin diperiksa. Kemudian, program menghitung apakah tahun tersebut adalah tahun kabisat dengan menggunakan kondisi logika: tahun tersebut harus habis dibagi 400, atau habis dibagi 4 tetapi tidak habis dibagi 100. Kondisi logika di atas merupakan definisi tahun kabisat yang ditetapkan

dalam kalender Gregorian, **dan tidak dapat di rubah**. Selanjutnya oleh Hasil dari perhitungan ini disimpan dalam variabel Kabisat, yang akan bernilai true jika tahun tersebut adalah tahun kabisat, dan false jika tidak.

3. Guided 3

Source Code

```
package main

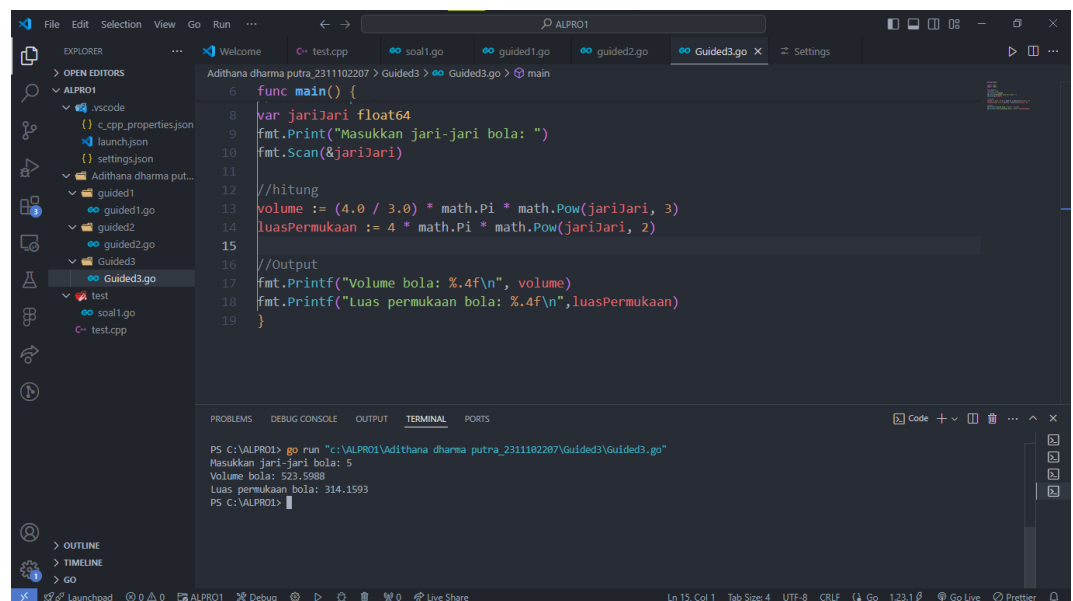
import "fmt"
import "math"

func main() {
    //Meminta Input
    var jariJari float64
    fmt.Print("Masukkan jari-jari bola: ")
    fmt.Scan(&jariJari)

    //hitung
    volume := (4.0 / 3.0) * math.Pi * math.Pow(jariJari, 3)
    luasPermukaan := 4 * math.Pi * math.Pow(jariJari, 2)

    //Output
    fmt.Printf("Volume bola: %.4f\n", volume)
    fmt.Printf("Luas permukaan bola: %.4f\n", luasPermukaan)
}
```

Screenshot



The screenshot displays the Visual Studio Code (VS Code) interface. The Explorer panel on the left shows the project structure with folders like 'ALPRO1' and 'guided3'. The main editor window shows the Go source code for 'Guided3.go', which is identical to the code provided in the 'Source Code' section. The bottom panel shows the 'TERMINAL' output, where the command 'go run "c:\ALPRO1\Adithana dharna putra_231182287\Guided3\Guided3.go"' has been executed. The output shows the program prompting for the radius, receiving the input '5', and then calculating and displaying the volume as '523.5988' and the surface area as '314.1593'.

```
PS C:\ALPRO1> go run "c:\ALPRO1\Adithana dharna putra_231182287\Guided3\Guided3.go"
Masukkan jari-jari bola: 5
Volume bola: 523.5988
Luas permukaan bola: 314.1593
PS C:\ALPRO1>
```


Deskripsi:

Kode ini adalah program Go yang menghitung volume dan luas permukaan bola berdasarkan jari-jari yang diberikan oleh pengguna. Program dimulai dengan mengimpor paket `fmt` untuk input/output dan `math` untuk fungsi matematika. Setelah itu, program meminta pengguna untuk memasukkan nilai jari-jari bola. Nilai ini kemudian digunakan untuk menghitung volume bola dengan rumus $(4.0 / 3.0) * \text{math.Pi} * \text{math.Pow}(\text{jariJari}, 3)$ dan luas permukaan bola dengan rumus $4 * \text{math.Pi} * \text{math.Pow}(\text{jariJari}, 2)$.

4. Guided 4**Source Code**

```
package main
import "fmt"

func main () {
    var celsius float64
    var fahrenheit float64
    var reamur float64
    var kelvin float64

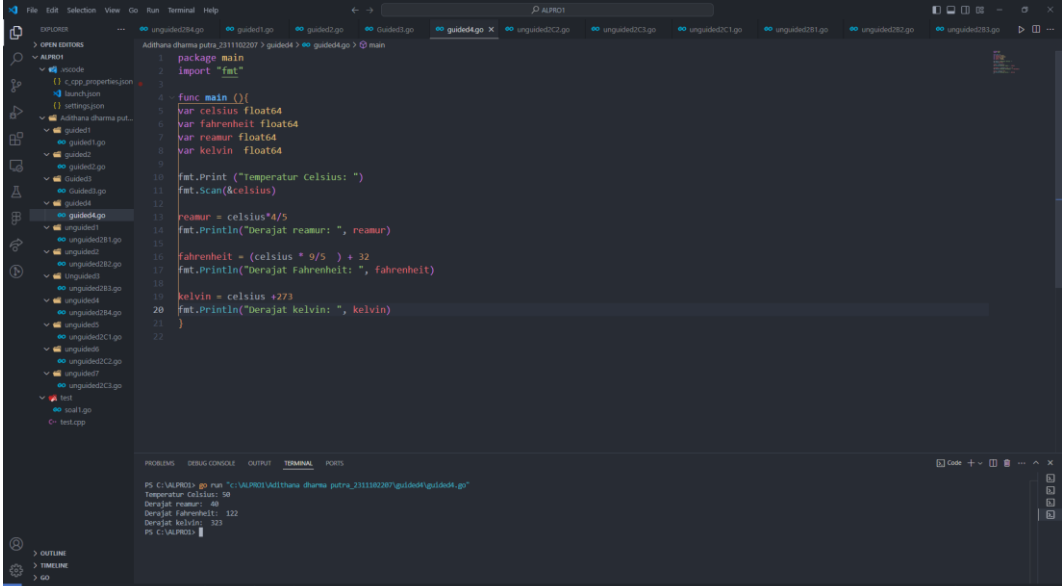
    fmt.Print ("Temperatur Celsius: ")
    fmt.Scan(&celsius)

    reamur = celsius*4/5
    fmt.Println("Derajat reamur: ", reamur)

    fahrenheit = (celsius * 9/5 ) + 32
    fmt.Println("Derajat Fahrenheit: ", fahrenheit)

    kelvin = celsius +273
    fmt.Println("Derajat kelvin: ", kelvin)
}
```

Screenshot



The screenshot shows a Go program in a VS Code editor. The program is named `guide4.go` and is located in the `guide4` directory. The code defines a `main` package and a `main` function. It declares four variables: `celsius` (float64), `fahrenheit` (float64), `reamur` (float64), and `kelvin` (float64). It prompts the user to enter a temperature in Celsius, reads the input, and then calculates the equivalent values in Reamur, Fahrenheit, and Kelvin using the following formulas:

- `reamur = celsius * 4 / 5`
- `fahrenheit = (celsius * 9 / 5) + 32`
- `kelvin = celsius + 273`

The program prints the results for each scale. The terminal output shows the following results:

```
PS C:\VALPROD> go run "C:\VALPROD\Addithana Putra_23111802807\guide4\guide4.go"
Temperatur Celsius: 50
Derajat reamur: 40
Derajat Fahrenheit: 122
Derajat kelvin: 323
PS C:\VALPROD>
```

Deskripsi:

Kode ini adalah program Go yang mengonversi suhu dari Celsius ke tiga skala suhu lainnya: Reamur, Fahrenheit, dan Kelvin. Program dimulai dengan mendeklarasikan empat variabel `celsius`, `fahrenheit`, `reamur`, dan `kelvin` untuk menyimpan nilai suhu dalam masing-masing skala. Setelah itu, program meminta pengguna untuk memasukkan suhu dalam Celsius. Nilai yang dimasukkan kemudian dikonversi ke Reamur dengan rumus $\text{reamur} = \text{celsius} \times 4 / 5$, ke Fahrenheit dengan rumus $\text{fahrenheit} = (\text{celsius} \times 9 / 5) + 32$, dan ke Kelvin dengan rumus $\text{kelvin} = \text{celsius} + 273$.

III. UNGUIDED

1. Unguided 1

Source Code

```
package main
import "fmt"

//fungsi Pengecekan
func cek(Input [][]string, urutan []string) bool {
    for i := 0; i < 5; i++ {
        for j := 0; j < len(urutan); j++ {
            if Input[i][j] != urutan[j] {
                return false
            }
        }
    }
    return true
}

func main() {
    //deklarasi array
    warnaInput := make([][]string, 5)
    for i := range warnaInput {
        warnaInput[i] = make([]string, 5)
    }
    //deklarasi susunan yang benar
    urutanWarna := []string{"merah", "kuning",
    "hijau", "ungu"}

    //meminta input
    for i := 0; i < 5; i++ {
        fmt.Printf("Percobaan %v: ", i+1)
        fmt.Scan(&warnaInput[i][0], &warnaInput[i][
1], &warnaInput[i][2], &warnaInput[i][3])
    }

    //memanggil fungsi cek dan mengeluarkan Output
    Hasil := cek(warnaInput, urutanWarna)
    fmt.Print("BERHASI: ", Hasil)
}
```

Screenshot

```
func cek(input [][]string, urutan []string) bool {  
    for i := 0; i < len(input); i++ {  
        if input[i][0] != urutan[i] {  
            return false  
        }  
    }  
    return true  
}  
  
func main() {  
    //deklarasi array  
    warnaInput := make([][]string, 5)  
    for i := range warnaInput {  
        warnaInput[i] = make([]string, 5)  
    }  
    //deklarasi susunan yang benar  
    urutanWarna := []string{"merah", "kuning", "hijau", "ungu"}  
    //meminta input  
    for i := 0; i < 5; i++ {  
        fmt.Printf("Percobaan %i: ", i+1)  
        fmt.Scan(&warnaInput[i][0], &warnaInput[i][1], &warnaInput[i][2], &warnaInput[i][3])  
    }  
    //memanggil fungsi cek dan mengeluarkan Output  
    Hasil := cek(warnaInput, urutanWarna)  
    fmt.Print("BERHASIL: ", Hasil)  
}
```

PROBLEM DEBUG CONSOLE OUTPUT TERMINAL PORTS

```
c:\ALPROG\adithana dhama putra_2311180207\unguided\unguide081.go:8  
PS C:\ALPROG> go run "c:\ALPROG\adithana dhama putra_2311180207\unguided\unguide081.go"  
Percobaan 1: merah kuning hijau ungu  
Percobaan 2: merah kuning hijau ungu  
Percobaan 3: merah kuning hijau ungu  
Percobaan 4: merah kuning hijau ungu  
Percobaan 5: merah kuning hijau ungu  
BERHASIL: true  
PS C:\ALPROG> go run "c:\ALPROG\adithana dhama putra_2311180207\unguided\unguide081.go"  
Percobaan 1: merah kuning hijau ungu  
Percobaan 2: ungu hijau kuning merah  
Percobaan 3: merah kuning hijau ungu  
Percobaan 4: merah kuning hijau ungu  
Percobaan 5: merah kuning hijau ungu  
BERHASIL: false  
PS C:\ALPROG>
```

Deskripsi:

Kode ini adalah program Go yang memeriksa apakah input warna dari pengguna sesuai dengan urutan warna yang telah ditentukan. Program dimulai dengan mendeklarasikan fungsi cek yang memeriksa apakah setiap elemen dalam array dua dimensi Input sesuai dengan urutan warna yang benar dalam array urutan. Fungsi ini mengembalikan true jika semua elemen sesuai, dan false jika ada yang tidak sesuai.

Dalam fungsi main, program mendeklarasikan array dua dimensi warnaInput untuk menyimpan input warna dari pengguna. Kemudian, program mendeklarasikan array urutanWarna yang berisi urutan warna yang benar: “merah”, “kuning”, “hijau”, dan “ungu”.

Program meminta pengguna untuk memasukkan warna dalam lima percobaan, di mana setiap percobaan terdiri dari empat warna. Input ini disimpan dalam array warnaInput. Setelah semua input diterima, program memanggil fungsi cek untuk memeriksa apakah input warna sesuai dengan urutan yang benar dan menampilkan hasilnya.

Dengan demikian, program ini memberikan cara untuk memverifikasi urutan warna yang dimasukkan oleh pengguna terhadap urutan yang telah ditentukan sebelumnya.

2. Unguided 2

Source Code

```
package main
import "fmt"

func main () {
    //deklarasi unsign integer (yang tidak
    bernilai negatif)
    var N uint64
    var i uint64
    var j uint64

    //membuat array
    fmt.Print("N: ")
    fmt.Scan(&N)
    bunga := make([]string, N)

    //memberikan peringatan jika inputan
    negatif atau kurang dari 0
    if N <= 0 {
        fmt.Printf("N: %v \nPita:
        \nPERINGATAN!\nINPUT BUKAN BILANGAN POSITIF
        ATAU SAMA DENGAN 0", N )
    }

    //melakukan input ke array
    for i = 0; i < N; i++ {
        fmt.Printf("Bunga %v: ", i+1)
        fmt.Scan(&bunga[i])
    }

    //mengoutputkan isi array sesuai format
    pita
    for j = 0 ; j < N ; j++ {
        fmt.Print(bunga[j] + "-")
    }
    fmt.Println()

    //Modifikasi
    fmt.Print("\n\nSETELAH MODIFIKASI\n\n")

    //membuat array
    var input string
    bungaMod := make([]string, 0)

    //melakukan input dan mengecek input !=
    SELESAI
    for i = 0; input != "SELESAI"; i++ {
        fmt.Printf("Bunga %v: ", i+1)
        fmt.Scan(&input)
        if input != "SELESAI" {
```

```

        bungaMod = append(bungaMod, input) }
    }

    //mencetak output sesuai format
    fmt.Print("Pita: ")
    for _, b := range bungaMod {
        fmt.Print(b + "-")
    }
    fmt.Println()

    //mencetak jumlah input
    fmt.Printf("Jumlah bunga yang dimasukkan:
    %v\n", len(bungaMod))
}

```

Screenshot

The screenshot shows a Go IDE with the following code in the editor:

```

package main
import "fmt"

func main() {
    //deklarasi unsigned integer (yang tidak bernilai negatif)
    var N uint64
    var i uint64
    var j uint64

    //membuat array
    fmt.Print("N: ")
    fmt.Scan(&N)
    bunga := make([]string, N)

    //memberikan peringatan jika inputan negatif atau kurang dari 0
    if N<=0{
        fmt.Printf("N: %v\nPita: \nPERINGATAN!\nINPUT BUKAN BILANGAN POSITIF ATAU SAMA DENGAN 0", N)
    }

    //melakukan input ke array
    for i = 0; i<N; i++ {
        fmt.Print("Bunga %v: ", i+1)
        fmt.Scan(&bunga[i])
    }

    //menampilkan isi array sesuai format pita
}

```

The terminal output shows the execution of the program:

```

PS C:\VALPROJ> go run "C:\VALPROJ\adithana putra_231102027\unguided2\unguided281.go"
N: 3
Bunga 1: kertas
Bunga 2: mawar
Bunga 3: tulip
Pita: kertas-mawar-tulip-
Jumlah bunga yang dimasukkan: 3
PS C:\VALPROJ>

```

Deskripsi

Kode ini adalah program Go yang meminta pengguna untuk memasukkan nama bunga, menampilkan daftar bunga dalam format pita, dan kemudian memodifikasi daftar bunga berdasarkan input tambahan dari pengguna. Program dimulai dengan mendeklarasikan variabel N untuk menyimpan jumlah bunga yang akan dimasukkan, serta variabel i dan j untuk iterasi. Jika N kurang dari atau sama dengan 0, program memberikan peringatan bahwa input tidak valid.

Setelah itu, program meminta pengguna untuk memasukkan nama bunga sebanyak N kali dan menyimpannya dalam array bunga. Daftar bunga ini

kemudian ditampilkan dalam format pita, di mana setiap nama bunga dipisahkan oleh tanda hubung.

Selanjutnya, pada bagian modifikasi, kita diminta membuat pengguna dapat terus memasukkan nama bunga sampai mereka mengetik "SELESAI". Nama bunga yang dimasukkan disimpan dalam array bungaMod. Setelah pengguna selesai memasukkan nama bunga, program menampilkan daftar bunga yang dimodifikasi dalam format pita dan mencetak jumlah total bunga yang dimasukkan.

3. Unguided 3

Source Code

```
package main
import "fmt"

func main () {
    //DEKLARASI VARIABEL
    var beratKanan float64
    var beratKiri float64

    //MEMERIKSA KONDISI
    for beratKanan < 9 && beratKanan >= 0 &&
        beratKiri < 9 && beratKiri >= 0 {
        fmt.Print("masukkan berat belanjaan di
        kedua kantong: ")
        fmt.Scan(&beratKanan, &beratKiri)
    }
    fmt.Print("Proses selesai")

    //Modifikasi
    fmt.Print("\n\nSETELAH MODIFIKASI\n\n")

    for {
        //INPUT
        fmt.Print("Masukkan berat belanjaan di
        kedua kantong: ")
        fmt.Scan(&beratKanan, &beratKiri)

        //CEK BERAT
        if beratKanan < 0 || beratKiri < 0 ||
            beratKanan+beratKiri > 150 {
            break
        }

        //CEK KONDISI
        if beratKanan-beratKiri >= 9 ||
            beratKiri-beratKanan >= 9 {
```

```

        fmt.Println("Sepeda motor pak Andi
akan oleng: true")
    } else {
        fmt.Println("Sepeda motor pak Andi
akan oleng: false")
    }
}
fmt.Print("Proses selesai.")
}

```

Screenshot

The screenshot shows a Go program in VS Code. The source code in the editor is as follows:

```

package main
import "fmt"

func main() {
    //DEKLARASI VARIABEL
    var beratKanan float64
    var beratKiri float64

    //MEMERIKSA KONDISI
    for beratKanan <= 0 && beratKiri <= 0 && beratKiri <= 0 {
        fmt.Print("Masukkan berat belanja di kedua kantong: ")
        fmt.Scan(&beratKanan, &beratKiri)
        fmt.Print("Proses selesai")
    }

    //Modifikasi
    fmt.Print("\n\nSETELAH MODIFIKASI\n\n")

    for {
        //INPUT
        fmt.Print("Masukkan berat belanja di kedua kantong: ")
        fmt.Scan(&beratKanan, &beratKiri)

        //CEK BERAT
        if beratKanan < 0 || beratKiri < 0 || beratKanan+beratKiri > 150 {
            break
        }
    }
}

```

The terminal output shows the program's execution:

```

PS C:\VALPROD> go run "C:\VALPROD\Adithana dhuma putra_2311102027\unguided283.go"
Masukkan berat belanja di kedua kantong: 5.5 1.0
Masukkan berat belanja di kedua kantong: 2.1 1.5
Masukkan berat belanja di kedua kantong: 2 6
Masukkan berat belanja di kedua kantong: 8 5.8
Proses selesai

SETELAH MODIFIKASI
Masukkan berat belanja di kedua kantong: 5.10
Sepeda motor pak Andi akan oleng: false
Masukkan berat belanja di kedua kantong: 55.6 70.2
Sepeda motor pak Andi akan oleng: true
Masukkan berat belanja di kedua kantong: 72.3 66.9
Sepeda motor pak Andi akan oleng: false
Masukkan berat belanja di kedua kantong: 59.5 96.7
Proses selesai.
PS C:\VALPROD>

```

Deskripsi:

Kode ini adalah program Go yang memeriksa keseimbangan berat belanjaan di dua kantong untuk memastikan sepeda motor Pak Andi tidak oleng. Program dimulai dengan mendeklarasikan dua variabel `beratKanan` dan `beratKiri` untuk menyimpan berat belanjaan di masing-masing kantong.

Program meminta pengguna untuk memasukkan berat belanjaan di kedua kantong dalam loop yang terus berjalan selama berat di kedua kantong kurang dari 9 kg dan tidak negatif. Jika kondisi ini tidak terpenuhi, loop berhenti dan program menampilkan pesan “Proses selesai”.

Setelah modifikasi, program memasuki loop tak terbatas yang meminta pengguna untuk memasukkan berat belanjaan di kedua kantong. Program memeriksa apakah berat di salah satu kantong negatif atau jika total berat melebihi 150 kg. Jika salah satu kondisi ini terpenuhi, loop berhenti.

Selanjutnya, program memeriksa apakah perbedaan berat antara kedua kantong lebih dari atau sama dengan 9 kg. Jika ya, program menampilkan pesan bahwa sepeda motor Pak Andi akan oleng (true), jika tidak, menampilkan pesan bahwa sepeda motor tidak akan oleng (false). Loop ini terus berjalan sampai kondisi berhenti terpenuhi.

4. Unguided 4

Source Code

```
package main

import "fmt"
import "math"

func main() {
    //DEKLARASII VARIABEL
    var kr float64
    fmt.Print("Masukkan nilai K: ")
    fmt.Scan(&kr)

    //MASUKKAN NILAI KE FUNGSI
    f := (math.Pow(4*kr+2, 2)) / ((4*kr + 1) *
(4*kr + 3))
    fmt.Printf("Nilai f(K) = %.10f\n",f)

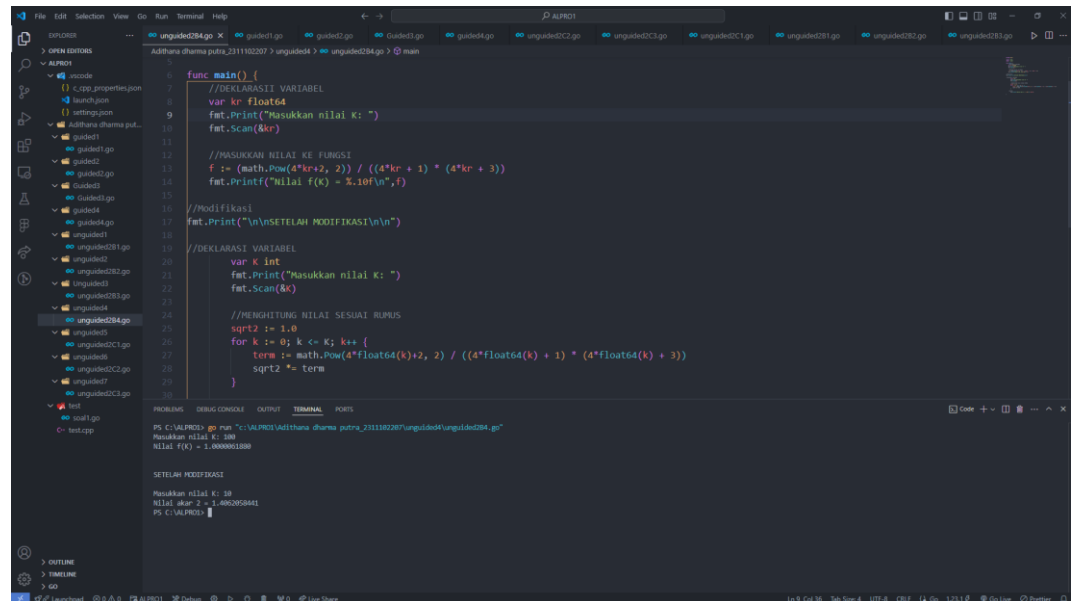
    //Modifikasi
    fmt.Print("\n\nSETELAH MODIFIKASI\n\n")

    //DEKLARASI VARIABEL
    var K int
    fmt.Print("Masukkan nilai K: ")
    fmt.Scan(&K)

    //MENGHITUNG NILAI SESUAI RUMUS
    sqrt2 := 1.0
    for k := 0; k <= K; k++ {
        term := math.Pow(4*float64(k)+2, 2) /
((4*float64(k) + 1) * (4*float64(k) + 3))
        sqrt2 *= term
    }

    //OUTPUT
    fmt.Printf("Nilai akar 2 = %.10f\n",sqrt2)
}
```

Screenshot



```
func main() {  
    //DEKLARASI VARIABEL  
    var kr float64  
    fmt.Println("Masukkan nilai k: ")  
    fmt.Scan(&kr)  
  
    //MASUKKAN NILAI KE FUNGSI  
    f := (math.Pow(4*kr+2, 2)) / ((4*kr + 1) * (4*kr + 3))  
    fmt.Printf("Nilai f(k) = %.10f\n", f)  
  
    //Modifikasi  
    fmt.Println("\nSetelah Modifikasi\n")  
  
    //DEKLARASI VARIABEL  
    var k int  
    fmt.Println("Masukkan nilai K: ")  
    fmt.Scan(&k)  
  
    //MENGHITUNG NILAI SESUAI RUMUS  
    sqrt2 := 1.0  
    for k := 0; k <= K; k++ {  
        term := math.Pow(4*float64(k)+2, 2) / ((4*float64(k) + 1) * (4*float64(k) + 3))  
        sqrt2 += term  
    }  
}
```

PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL PORTS

```
PS C:\ALPRODI> go run "C:\ALPRODI\Addithana putra_2311102207\unguided4\unguided094.go"  
Masukkan nilai K: 10  
Nilai f(k) = 1.0000001000  
  
Setelah Modifikasi  
Masukkan nilai K: 10  
Nilai akar 2 = 1.4062059441  
PS C:\ALPRODI>
```

Deskripsi:

Kode ini adalah program Go yang menghitung nilai fungsi matematika berdasarkan input pengguna. Program dimulai dengan meminta pengguna untuk memasukkan nilai kr , kemudian menghitung nilai fungsi ($f(K)$) menggunakan rumus yang ada di modul dan menampilkan hasilnya dengan presisi sepuluh desimal.

Setelah modifikasi, program meminta pengguna untuk memasukkan nilai K sebagai bilangan bulat. Program kemudian menghitung nilai perkiraan akar 2 menggunakan produk dari serangkaian nilai yang dihitung dengan rumus yang sama. Loop berjalan dari 0 hingga K , mengalikan setiap nilai $term$ ke dalam variabel $sqrt2$. Hasil akhir dari perkiraan akar 2 ditampilkan dengan presisi sepuluh desimal.

5. Unguided 5

Source Code

```
package main

import "fmt"

func main() {
    //deklarasi variabel
    var beratGram int
    fmt.Print("Masukkan berat parsel (dalam gram):")
    ")
    fmt.Scan(&beratGram)

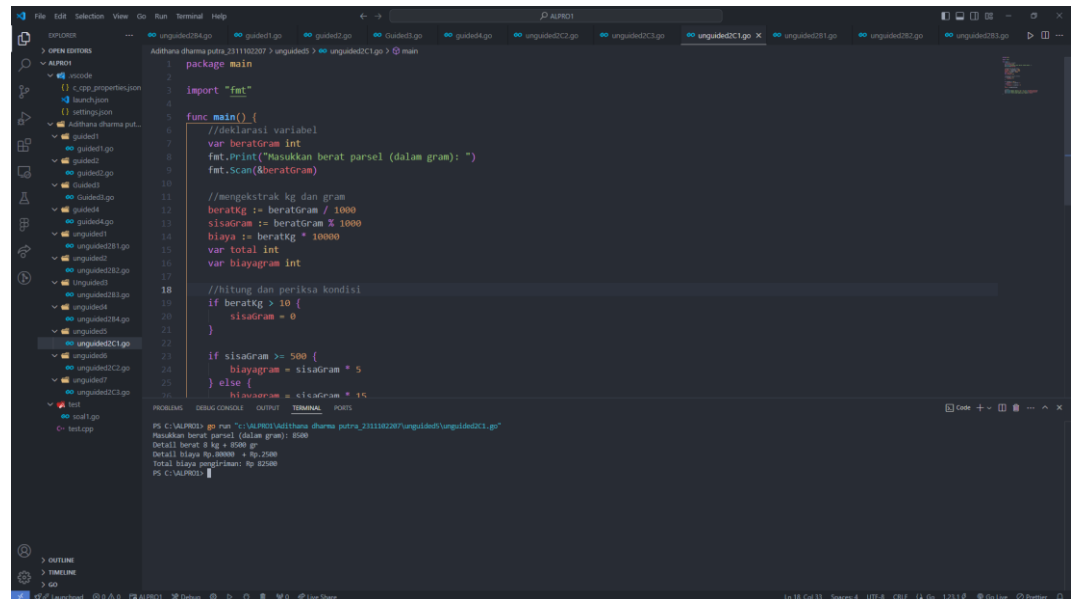
    //mengestrak kg dan gram
    beratKg := beratGram / 1000
    sisaGram := beratGram % 1000
    biaya := beratKg * 10000
    var total int
    var biayagram int

    //hitung dan periksa kondisi
    if beratKg > 10 {
        sisaGram = 0
    }

    if sisaGram >= 500 {
        biayagram = sisaGram * 5
    } else {
        biayagram = sisaGram * 15
    }
    total = biayagram+biaya

    //output
    fmt.Printf("Detail berat %v kg + %v\n", beratKg, beratGram)
    fmt.Printf("Detail biaya Rp.%v +\n", biaya, biayagram)
    fmt.Printf("Total biaya pengiriman:\n", total)
    Rp %d\n", total)
}
```

Screenshot



Deskripsi:

Kode ini adalah program Go yang menghitung biaya pengiriman parcel berdasarkan beratnya dalam gram. Program dimulai dengan meminta pengguna untuk memasukkan berat parcel dalam gram. Berat ini kemudian dipecah menjadi kilogram dan sisa gram. Biaya dasar dihitung berdasarkan berat kilogram, dengan tarif Rp 10.000 per kilogram.

Selanjutnya, program menentukan biaya tambahan berdasarkan sisa gram: jika sisa gram lebih dari atau sama dengan 500 gram, biaya tambahan dihitung dengan tarif Rp 5 per gram, jika kurang dari 500 gram, tarifnya Rp 15 per gram. Total biaya pengiriman adalah jumlah dari biaya dasar dan biaya tambahan.

Akhirnya, program menampilkan detail berat dalam kilogram dan gram, rincian biaya dasar dan tambahan, serta total biaya pengiriman.

6. Unguided 6 Source Code

```
package main
import "fmt"

func main() {
    var nam float64
    var nmk string
```

```

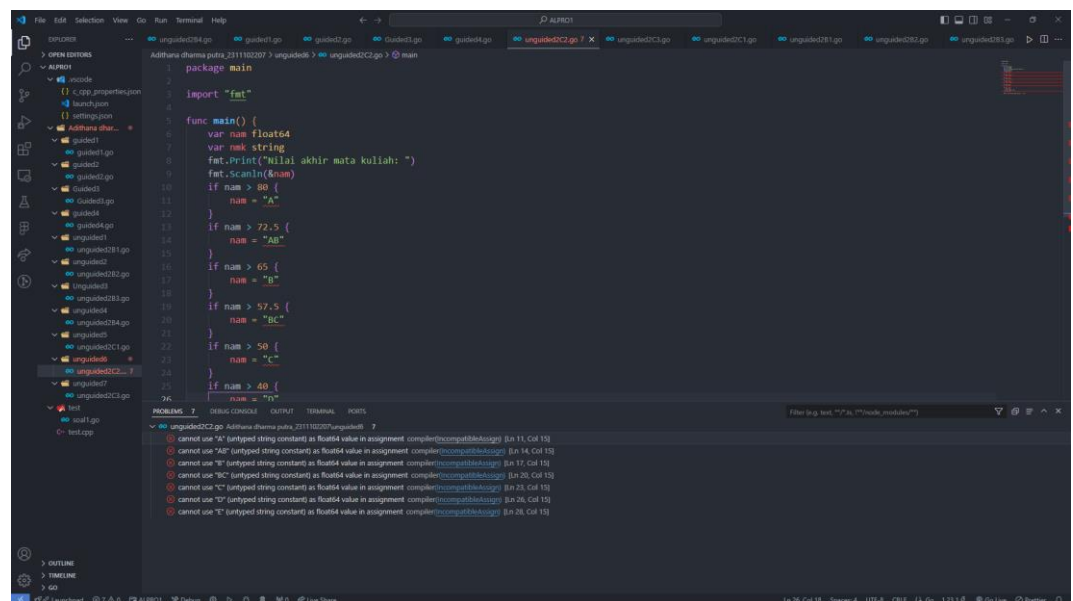
    fmt.Println("Nilai akhir mata kuliah: ")
    fmt.Scanln(&nam)

    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 mata kuliah: ", nmk)
}

```

Screenshot



Gambar 1.a kode error sebelum eksekusi

The screenshot shows a Go IDE with a file explorer on the left, a code editor in the center, and a terminal at the bottom. The code editor displays a Go program with a `main` function. The program declares a variable `nam` of type `string` and uses `fmt.Scanln(&nam)` to read input. It then uses a series of `if` and `else if` statements to assign a grade (`rnk`) based on the input value. The conditions are: `if nam > 80 { rnk = "A" }`, `else if nam > 72.5 { rnk = "Aa" }`, `else if nam > 65 { rnk = "B" }`, `else if nam > 57.5 { rnk = "Bc" }`, `else if nam > 50 { rnk = "C" }`, `else if nam > 40 { rnk = "D" }`, and `else if nam <= 40 { rnk = "E" }`. Finally, it prints the grade using `fmt.Println("Nilai mata kuliah: ", rnk)`. The terminal shows the output of the program for various input values: 85.5 (A), 78.6 (Aa), 8 (B), 49.5 (D), and 0 (E).

```
func main() {  
    var rnk string  
  
    fmt.Println("Nilai akhir mata kuliah: ")  
    fmt.Scanln(&nam)  
  
    if nam > 80 {  
        rnk = "A"  
    } else if nam > 72.5 {  
        rnk = "Aa"  
    } else if nam > 65 {  
        rnk = "B"  
    } else if nam > 57.5 {  
        rnk = "Bc"  
    } else if nam > 50 {  
        rnk = "C"  
    } else if nam > 40 {  
        rnk = "D"  
    } else if nam <= 40 {  
        rnk = "E"  
    }  
  
    fmt.Println("Nilai mata kuliah: ", rnk)  
}
```

PROBLEMS DEBAG CONSOLE OUTPUT TERMINAL PORTS
PS C:\ALPRO1> go run "C:\ALPRO1\Addhana_dharma_putra_2311102207\unguided0\unguided02.go"
Nilai akhir mata kuliah: 85.5
Nilai mata kuliah: A
PS C:\ALPRO1> go run "C:\ALPRO1\Addhana_dharma_putra_2311102207\unguided0\unguided02.go"
Nilai akhir mata kuliah: 78.6
Nilai mata kuliah: Aa
PS C:\ALPRO1> go run "C:\ALPRO1\Addhana_dharma_putra_2311102207\unguided0\unguided02.go"
Nilai akhir mata kuliah: 8
Nilai mata kuliah: B
PS C:\ALPRO1> go run "C:\ALPRO1\Addhana_dharma_putra_2311102207\unguided0\unguided02.go"
Nilai akhir mata kuliah: 49.5
Nilai mata kuliah: D
PS C:\ALPRO1>

Gambar 1.b kode setelah diperbaiki

Deskripsi

- 1) Jawaban untuk nomor 1, kode pada awalnya belum bisa dijalankan karena terdapat kesalahan deklarasi seperti yang dapat dilihat pada gambar 1.a, ini karena kita mendefinisikan `nam` pada setiap cabang `if` sebagai string. Setelah kita perbaiki ini, lalu kita masukkan 80,1 maka outputnya adalah D
- 2) Kesalahan program tersebut adalah dalam menyatakan kondisi `if` dan `elseif` nya. Pada kode di modul `if` berjalan langsung pada yang memiliki pasangan `else if` sedangkan kondisi `if` lainnya diabaikan.
- 3) Pada kode terlampir telah saya perbaiki, terlampir output pada gambar 1.b

7. Unguided 7

Source Code

```
package main

import "fmt"
import "math"

// Fungsi untuk mencari faktor
func cariFaktor(n int) []int {
    var faktor []int
    for i := 1; i <= n; i++ {
        if n%i == 0 {
            faktor = append(faktor,i)
        }
    }
    return faktor
}

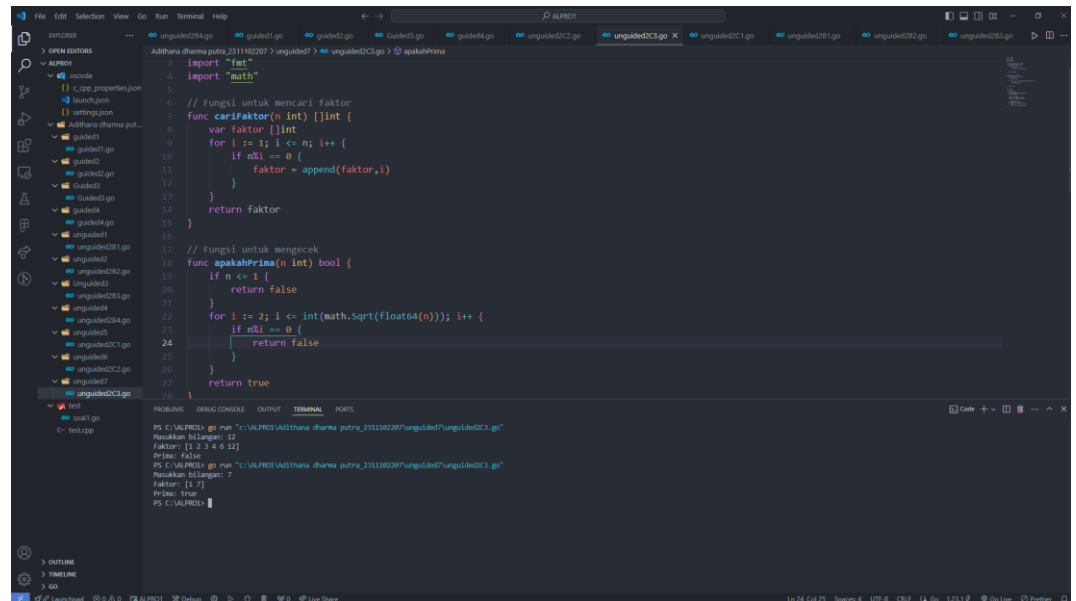
// Fungsi untuk mengecek
func apakahPrima(n int) bool {
    if n <= 1 {
        return false
    }
    for i := 2; i <= int(math.Sqrt(float64(n)));
i++ {
        if n%i == 0 {
            return false
        }
    }
    return true
}

func main() {
    //input
    var angka int
    fmt.Print("Masukkan bilangan: ")
    fmt.Scan(&angka)

    //panggil fungsi
    faktor := cariFaktor(angka)
    fmt.Println("Faktor:",faktor)

    //cek prima
    if apakahPrima(angka) {
        fmt.Println("Prima: true")
    } else {
        fmt.Println("Prima: false")}
}
```

Screenshot



```
1 import "fmt"
2
3 // Fungsi untuk mencari faktor
4 func cariFaktor(n int) []int {
5     var faktor []int
6     for i := 1; i <= n; i++ {
7         if n%i == 0 {
8             faktor = append(faktor, i)
9         }
10    }
11    return faktor
12}
13
14// Fungsi untuk mengecek
15func apakahPrima(n int) bool {
16    if n <= 1 {
17        return false
18    }
19    for i := 2; i <= int(math.Sqrt(float64(n))); i++ {
20        if n%i == 0 {
21            return false
22        }
23    }
24    return true
25}
```

Terminal Output:

```
PS C:\VALPRO> go run "C:\VALPRO\Addithana putra_2311102207\unguided7\unguided023.go"
Masukkan bilangan: 12
Faktor: [1 2 3 4 6 12]
Prima: false
PS C:\VALPRO> go run "C:\VALPRO\Addithana putra_2311102207\unguided7\unguided023.go"
Masukkan bilangan: 7
Faktor: [1 7]
Prima: true
PS C:\VALPRO>
```

Deskripsi

Kode ini adalah program Go yang melakukan dua fungsi utama: mencari faktor dari sebuah bilangan dan memeriksa apakah bilangan tersebut adalah bilangan prima.

Fungsi cariFaktor

Fungsi ini menerima sebuah bilangan n dan mengembalikan slice yang berisi semua faktor dari n.

Loop berjalan dari 1 hingga n, dan jika n habis dibagi i, maka i ditambahkan ke slice faktor.

Fungsi apakahPrima

Fungsi ini memeriksa apakah sebuah bilangan n adalah bilangan prima.

Jika n kurang dari atau sama dengan 1, fungsi mengembalikan false.

Loop berjalan dari 2 hingga akar kuadrat dari n. Jika n habis dibagi i, maka n bukan bilangan prima dan fungsi mengembalikan false.

Jika tidak ada pembagi yang ditemukan, fungsi mengembalikan true.

Fungsi cariFaktor dipanggil dengan angka sebagai argumen, dan hasilnya ditampilkan.

Fungsi apakahPrima dipanggil untuk memeriksa apakah angka adalah bilangan prima, dan hasilnya ditampilkan.

Contoh Eksekusi

Jika pengguna memasukkan angka 12:

Faktor dari 12 adalah: [1, 2, 3, 4, 6, 12]

12 bukan bilangan prima, sehingga outputnya adalah Prima: false.

Jika pengguna memasukkan angka 7:

Faktor dari 7 adalah: [1, 7]

7 adalah bilangan prima, sehingga outputnya adalah Prima: true