

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL II
REVIEW STRUKTUR KONTROL**



Disusun Oleh :

Nia Novela Ariandini / 2311102057

IF-11-05

Dosen Pengampu :

Arif Amrulloh, S.Kom., M.Kom

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Bahasa Go (Golang)

Bahasa Go, atau sering disebut Golang, adalah bahasa pemrograman yang dikembangkan oleh Google pada tahun 2007 dan dirilis ke publik pada 2009. Go didesain untuk menjadi bahasa yang sederhana, efisien, serta mudah dipelajari, dengan fitur yang mendukung pemrograman paralel dan konkuren. Bahasa ini banyak digunakan dalam pengembangan sistem, aplikasi web, dan infrastruktur cloud. Golang memiliki beberapa prinsip utama dalam desainnya, yaitu :

1. Kesederhanaan : Sintaks yang ringkas dan mudah dipahami.
2. Kecepatan : Waktu kompilasi cepat dan performa runtime optimal.
3. Keterbacaan : Kode yang mudah dipelihara dan dipahami.

Struktur Program Golang

Dalam Go, tipe data struct memungkinkan pengguna untuk menggabungkan beberapa kolom yang berbeda menjadi satu entitas. Program Go selalu memiliki dua komponen utama :

1. package main : Menandai bahwa file tersebut merupakan program utama.
2. func main : Berisi logika utama dari program tersebut.

Tipe Data Dalam Golang

Tipe Data Dasar :

1. Integer : Menyimpan bilangan bulat seperti int, int8, int16, int32, int64.
2. Float : Menyimpan bilangan pecahan seperti float32, float64.
3. Boolean : Menyimpan nilai `true` atau `false`.
4. String : Menyimpan teks yang diapit dengan tanda kutip ganda.

Tipe Data Terstruktur:

1. Array : Kumpulan elemen dengan tipe yang sama dan ukuran tetap
2. Slice : Versi dinamis dari array, yang dapat berubah ukurannya
3. Struct : Tipe data yang menggabungkan berbagai tipe dalam satu kesatuan.

Struktur Kontrol

Struktur kontrol adalah elemen penting yang mengatur alur program, dan Go memiliki berbagai mekanisme untuk ini :

1. **Statement If-Else** Digunakan untuk mengevaluasi suatu kondisi dan menjalankan kode berdasarkan hasil kondisi tersebut.
2. **Statement Switch** Alternatif lebih efisien dari if-else yang panjang, digunakan untuk memilih berdasarkan nilai variabel.
3. **Statement For** Satu-satunya struktur loop dalam Go, yang bisa digunakan dalam berbagai pola iterasi.

Fungsi adalah kumpulan kode yang melakukan tugas spesifik dan dapat menerima parameter serta mengembalikan nilai.

Penanganan Error dalam Go, error ditangani dengan mengembalikan nilai error dari sebuah fungsi.

II. GUIDED

1. GUIDED 1 – SOAL NAMA

Soal Studi Case : Membuat sebuah program sederhana yang meminta pengguna memasukkan nama mereka, lalu mencetak nama tersebut ke layar. Program ini dirancang menggunakan bahasa pemrograman Golang dan bertujuan untuk memahami dasar penggunaan input/output serta variabel.

Sourcecode

```
package main

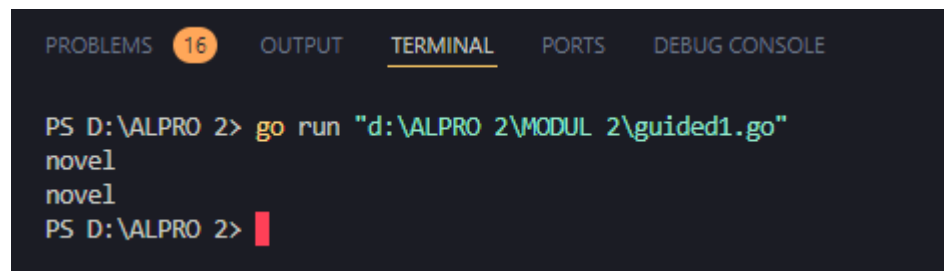
import (
    "fmt" // Mengimpor paket fmt yang menyediakan fungsi
    untuk input/output standar
)

func main() {
    var nama string // Deklarasi variabel nama dengan tipe
    string

    // Menggunakan fmt.Scanln untuk membaca input dari
    pengguna.
    // Harus menggunakan &nama karena Scanln mengharapkan
    pointer agar bisa mengisi nilai ke variabel nama
    fmt.Scanln(&nama)

    // Mencetak nilai variabel nama ke layar setelah input
    diambil dari pengguna
    fmt.Println(nama)
}
```

Screenshot Output



```
PROBLEMS 16 OUTPUT TERMINAL PORTS DEBUG CONSOLE

PS D:\ALPRO 2> go run "d:\ALPRO 2\MODUL 2\guided1.go"
novel
novel
PS D:\ALPRO 2> █
```

Deskripsi Program

Program ini adalah program sederhana yang menerima nama pengguna sebagai input dan menampilkannya di layar menggunakan paket fmt, yang dapat menangani input dan output di Golang.

Algoritma : Tipe data string harus digunakan untuk mendeklarasikan variabel nama. Dengan menggunakan `fmt.Scanln()`, ambil input pengguna dan simpan hasilnya ke dalam variabel nama. Kemudian, gunakan `fmt.Println()`, untuk mencetak nilai dari variabel nama yang mengandung input pengguna.

Cara Program Berfungsi : Program dimulai dengan mendeklarasikan variabel nama sebagai string, yang akan menampung input pengguna. Program menggunakan fungsi `fmt.Scanln()` untuk meminta pengguna memasukkan sebuah nilai (nama). Fungsi ini membaca input pengguna dan menyimpannya di variabel nama.

2. GUIDED 2 – 2B - 1

Soal Studi Case : Siswa kelas IPA di salah satu sekolah menengah atas di Indonesia sedang mengadakan praktikum kimia. Di setiap percobaan akan menggunakan 4 tabung reaksi, yang mana susunan warna cairan di setiap tabung akan menentukan hasil percobaan. Siswa diminta untuk mencatat hasil percobaan tersebut. Percobaan dikatakan berhasil apabila susunan warna zat cair pada gelas 1 hingga gelas 4 secara berturut-turut adalah ‘merah’, ‘kuning’, ‘hijau’, dan ‘ungu’ selama 5 kali percobaan berulang. Buatlah sebuah program yang menerima input berupa warna dari ke 4 gelas reaksi sebanyak 5 kali percobaan. Kemudian program akan menampilkan true apabila urutan warna sesuai dengan informasi yang diberikan pada paragraf sebelumnya, dan false untuk urutan warna lainnya. Perhatikan contoh sesi interaksi program seperti di bawah ini (teks bergaris bawah adalah input/read):

Sourcecode

```
package main

import (
    "bufio"
    "fmt"
    "os"
    "strings"
)

func main() {
    // Urutan warna yang benar
    correctOrder := []string{"merah", "kuning", "hijau",
    "ungu"}

    // Membaca input untuk 5 percobaan
    reader := bufio.NewReader(os.Stdin)
```

```

success := true

for i := 1; i <= 5; i++ {
    fmt.Printf("Percobaan %d : ", i)

    // Membaca input dari pengguna
    input, _ := reader.ReadString('\n')
    input = strings.TrimSpace(input)

    // Memisahkan input berdasarkan spasi
    colors := strings.Split(input, " ")

    // Mengecek apakah urutan warna sesuai
    for j := 0; j < 4; j++ {
        if colors[j] != correctOrder[j] {
            success = false
            break
        }
    }

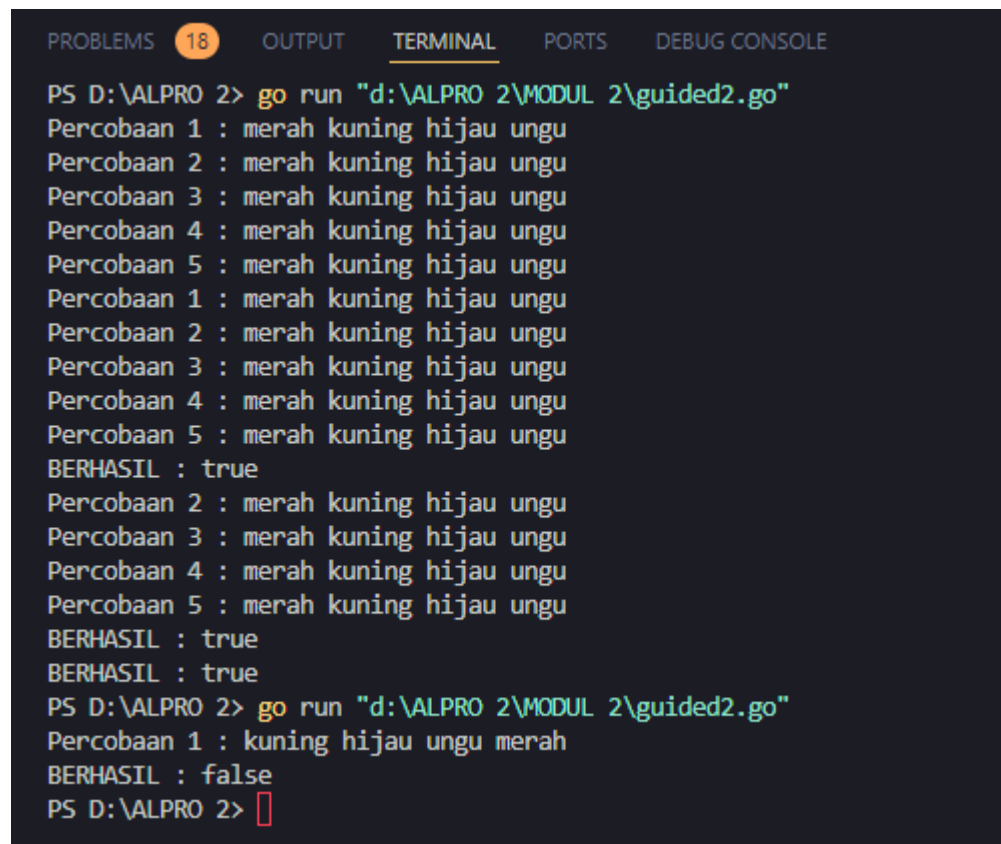
    // Jika ada percobaan yang tidak sesuai, keluar
    // dari loop
    if !success {
        break
    }
}

// Menampilkan hasil
if success {
    fmt.Println("BERHASIL : true")
} else {
    fmt.Println("BERHASIL : false")
}

// Mencetak nilai variabel nama ke layar setelah input
// diambil dari pengguna
fmt.Println(nama)
}

```

Screenshoot Output

A screenshot of a terminal window showing the execution of a Go program. The terminal has tabs for PROBLEMS, OUTPUT, TERMINAL (selected), PORTS, and DEBUG CONSOLE. The command 'go run "d:\ALPRO 2\MODUL 2\guided2.go"' is entered. The program outputs five trials of a color sequence: 'Percobaan 1 : merah kuning hijau ungu' through 'Percobaan 5 : merah kuning hijau ungu'. After each trial, it prints 'BERHASIL : true'. Then, it runs a second trial: 'Percobaan 1 : kuning hijau ungu merah', which results in 'BERHASIL : false'. The prompt 'PS D:\ALPRO 2>' is shown at the bottom with a cursor.

```
PROBLEMS 18 OUTPUT TERMINAL PORTS DEBUG CONSOLE
PS D:\ALPRO 2> go run "d:\ALPRO 2\MODUL 2\guided2.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
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
BERHASIL : true
PS D:\ALPRO 2> go run "d:\ALPRO 2\MODUL 2\guided2.go"
Percobaan 1 : kuning hijau ungu merah
BERHASIL : false
PS D:\ALPRO 2> 
```

Deskripsi Program

Program ini dirancang untuk menguji kemampuan pengguna untuk memasukkan urutan warna yang benar dalam lima percobaan. Urutan warna yang benar adalah "merah", "kuning", "hijau", dan "ungu". Jika pengguna berhasil memasukkan urutan warna dengan benar dalam lima percobaan, program akan mencetak "BERHASIL : benar", jika tidak, program akan mencetak "TIDAK BERHASIL : salah"

Algoritma untuk Program : Mulai aplikasi. Tentukan urutan warna yang benar: merah, kuning, hijau, dan ungu. Untuk memulai pembacaan input pengguna, gunakan `bufio.NewReader()`. Coba lima kali untuk membaca komentar pengguna. Baca komentar pengguna setiap kali. Untuk membedakan warna yang dimasukkan, pisahkan input berdasarkan ruang. Periksa apakah semua warna sesuai dengan urutan yang benar. Jika ada warna yang tidak sesuai dengan urutan yang benar, hentikan eksperimen dan cetak "BERHASIL : false". Jika semua eksperimen benar, cetak "BERHASIL : true".

Cara Program Berfungsi : Program dimulai dengan mendeklarasikan urutan warna yang benar dalam array urutan yang benar, yang terdiri dari "merah", "kuning", "hijau", dan "ungu". Membaca input pengguna melalui terminal, program menggunakan `bufio.NewReader()` untuk membaca input pengguna. Setiap input dianggap sebagai eksperimen, dan aplikasi akan meminta pengguna memasukkan empat warna terpisah oleh spasi. Memisahkan Input: Dengan menggunakan fungsi `strings.Split()`, input pengguna akan dipecah menjadi spasi. Misalnya, jika Anda memasukkan "merah kuning hijau ungu", data akan dibagi menjadi array yang terdiri dari "merah", "kuning", "hijau", dan "ungu".

3. GUIDED 3 – SOAL ABCDE

Soal Studi Case : Membuat program sederhana yang memungkinkan pengguna memasukkan 5 angka integer, kemudian program menjumlahkan kelima angka tersebut dan menampilkan hasilnya.

Sourcecode

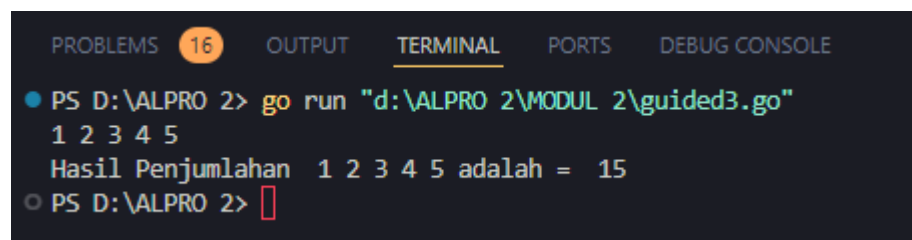
```
package main

import (
    "fmt"
)

func main() {
    var a, b, c, d, e int
    var hasil int
    fmt.Scanln(&a, &b, &c, &d, &e)

    hasil = a + b + c + d + e
    fmt.Println("Hasil Penjumlahan ", a, b, c, d, e,
        "adalah = ", hasil)
}
```

Screenshoot Output



```
PROBLEMS 16 OUTPUT TERMINAL PORTS DEBUG CONSOLE
PS D:\ALPRO 2> go run "d:\ALPRO 2\MODUL 2\guided3.go"
1 2 3 4 5
Hasil Penjumlahan 1 2 3 4 5 adalah = 15
PS D:\ALPRO 2> 
```


Deskripsi Program

Program di atas adalah aplikasi sederhana yang memungkinkan pengguna menjumlahkan lima angka integer yang dimasukkan secara berurutan dan kemudian menampilkan hasil penjumlahan. Program membaca lima angka integer yang dimasukkan secara berurutan, kemudian menghitung total angka-angka tersebut dan menampilkannya di layar.

Algoritma untuk Program : Deklarasikan lima variabel integer a, b, c, d, e untuk digunakan untuk menyimpan angka yang dimasukkan oleh pengguna. Deklarasikan juga variabel hasil untuk menyimpan hasil penjumlahan dari kelima angka. Gunakan `fmt.Scanln()` untuk membaca input pengguna untuk masing-masing dari lima angka tersebut dan menyimpannya di masing-masing variabel. Hitung hasil penjumlahan dari kelima angka: $\text{hasil} = a + b + c + d + e$. Tampilkan hasil penjumlahan dan angka yang dimasukkan oleh pengguna.

Cara Program Berfungsi : Pertama, program meminta pengguna untuk memasukkan lima angka berturut-turut; setelah mereka melakukannya, angka-angka tersebut disimpan dalam variabel a, b, c, d, dan e. Selanjutnya, program menghitung penjumlahan dari kelima angka dan menyimpan hasilnya di variabel hasil. Terakhir, program mencetak hasil penjumlahan dan menampilkan input pengguna dan hasilnya di layar.

4. GUIDED 4 – SOAL NAMA

Soal Studi Case :

Diberikan sebuah nilai akhir mata kuliah (NAM) [0..100] dan standar penilaian nilai mata kuliah (NMK) sebagai berikut :

NAM	NMK
$\text{NAM} > 80$	A
$72.5 < \text{NAM} \leq 80$	AB

$65 < \text{NAM} \leq 72.5$	B
$57.5 < \text{NAM} \leq 65$	BC
$50 < \text{NAM} \leq 57.5$	C
$40 < \text{NAM} \leq 50$	D
$\text{NAM} \leq 40$	E

Sourcecode

```
package main

import (
    "fmt"
)

func main() {
    var nilai float32
    var indeks string
```

```

// Meminta input nilai
fmt.Print("Masukkan nilai: ")
fmt.Scan(&nilai)

// Logika penentuan nilai huruf berdasarkan nilai
numerik
if nilai >= 80 {
    indeks = "A"
} else if nilai >= 70 {
    indeks = "B"
} else if nilai >= 65 {
    indeks = "C"
} else if nilai >= 45 {
    indeks = "D"
} else if nilai >= 40 {
    indeks = "E"
} else {
    indeks = "F"
}

// Menampilkan hasil
fmt.Printf("Nilai Indeks untuk nilai %.2f adalah
%s\n", nilai, indeks)
}

```

Screenshoot Output

```

PROBLEMS 16 OUTPUT TERMINAL PORTS DEBUG CONSOLE

PS D:\ALPRO 2> go run "d:\ALPRO 2\MODUL 2\guided4.go"
● Masukkan nilai: 80.1
Nilai Indeks untuk nilai 80.10 adalah A
● PS D:\ALPRO 2> go run "d:\ALPRO 2\MODUL 2\guided4.go"
Masukkan nilai: 93.5
Nilai Indeks untuk nilai 93.50 adalah A
● PS D:\ALPRO 2> go run "d:\ALPRO 2\MODUL 2\guided4.go"
Masukkan nilai: 70.6
Nilai Indeks untuk nilai 70.60 adalah B
● PS D:\ALPRO 2> go run "d:\ALPRO 2\MODUL 2\guided4.go"
Masukkan nilai: 49.5
Nilai Indeks untuk nilai 49.50 adalah D
○ PS D:\ALPRO 2>

```

Deskripsi Program

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

Jawab :

```
PROBLEMS 16 OUTPUT TERMINAL PORTS DEBUG CONSOLE
PS D:\ALPRO 2> go run "d:\ALPRO 2\MODUL 2\guided4.go"
● Masukkan nilai: 80.1
  Nilai Indeks untuk nilai 80.10 adalah A
○ PS D:\ALPRO 2> █
```

Program menampilkan seksekusi sesuai dengan spesifikasi soal, dimana nilai 80.1 merupakan lebih besar dari 80 dan mendapatkan nilai A.

- b. Apa saja kesalahan dari program tersebut? Mengapa demikian? Jelaskan alur program seharusnya!

Jawab :

Masalah batas nilai : Terdapat kesalahan dalam penentuan rentang nilai, seperti pada kondisi `nam > 80` yang membuat nilai 80.0 tidak termasuk dalam kategori "A". Solusinya adalah dengan menggunakan operator `>=` agar batas bawah juga tercakup dengan benar, misalnya nilai 80.0 harus termasuk dalam kategori "A". Penggunaan Float32 : Penggunaan tipe data float32 tidak diperlukan. Sebaiknya gunakan float64, karena lebih umum dan memberikan presisi yang lebih tinggi. Alur Program yang Tepat : Nilai `nam` harus diproses dengan rentang yang benar, di mana batas bawah setiap kategori menggunakan operator `>=`. Sebagai contoh, nilai 80.0 termasuk dalam kategori "A". Alur program seharusnya dimulai dengan memeriksa nilai tertinggi (A), kemudian secara bertahap memeriksa nilai-nilai yang lebih rendah hingga mencapai "F".

- c. Perbaiki program tersebut! Ujilah dengan masukan: 93.5; 70.6; dan 49.5. Seharusnya keluaran yang diperoleh adalah 'A', 'B', dan 'D'.

Jawab :

```
package main

import (
    "fmt"
)

func main() {
    var nilai float32
    var indeks string

    // Meminta input nilai
    fmt.Print("Masukkan nilai: ")
    fmt.Scan(&nilai)

    // Logika penentuan nilai huruf berdasarkan nilai
    numerik
    if nilai >= 80 {
```

```
        indeks = "A"
    } else if nilai >= 70 {
        indeks = "B"
    } else if nilai >= 65 {
        indeks = "C"
    } else if nilai >= 45 {
        indeks = "D"
    } else if nilai >= 40 {
        indeks = "E"
    } else {
        indeks = "F"
    }

    // Menampilkan hasil
    fmt.Printf("Nilai Indeks untuk nilai %.2f adalah\n", nilai, indeks)
}
```

Program di atas digunakan untuk mengonversi nilai numerik menjadi indeks huruf berdasarkan rentang nilai tertentu. Pertama, pengguna diminta memasukkan nilai melalui input. Program kemudian memeriksa nilai yang dimasukkan dengan serangkaian kondisi `if-else`. Jika nilai tersebut lebih besar atau sama dengan 80, indeks yang diberikan adalah "A". Jika nilainya berada antara 70 hingga kurang dari 80, diberikan indeks "B", dan seterusnya hingga nilai di bawah 40 yang mendapatkan indeks "F". Setelah semua kondisi diperiksa, program menampilkan hasil berupa indeks yang sesuai, dengan format dua angka desimal untuk nilai input. Sebagai contoh, jika pengguna memasukkan nilai 75.5, program akan menghasilkan output: "Nilai Indeks untuk nilai 75.50 adalah B".

III. UNGUIDED

1. UNGUIDED 1 – 2B - 2

Soal Studi Case : Suatu pita (string) berisi kumpulan nama-nama bunga yang dipisahkan oleh spasi dan ‘–’, contoh pita diilustrasikan seperti berikut ini. Pita: mawar – melati – tulip – teratai – kamboja – anggrek. Buatlah sebuah program yang menerima input sebuah bilangan bulat positif (dan tidak nol) N, kemudian program akan meminta input berupa nama bunga secara berulang sebanyak N kali dan nama tersebut disimpan ke dalam pita. (Petunjuk: gunakan operasi penggabungan string dengan operator “+”). Tampilkan isi pita setelah proses input selesai. Perhatikan contoh sesi interaksi program seperti di bawah ini (teks bergaris bawah adalah input/read) :

N: <u>3</u> Bunga 1: <u>Kertas</u> Bunga 2: <u>Mawar</u> Bunga 3: <u>Tulip</u> Pita: Kertas – Mawar – Tulip –	N : <u>0</u> Pita :
---	------------------------

Modifikasi program sebelumnya, proses input akan berhenti apabila user mengetikkan ‘SELESAI’. Kemudian tampilkan isi pita beserta banyaknya bunga yang ada di dalam pita. Perhatikan contoh sesi interaksi program seperti di bawah ini (teks bergaris bawah adalah input/read) :

Sourcecode

```
package main

import (
    "bufio"
    "fmt"
    "os"
    "strings"
)

func main() {
    var bunga []string
    scanner := bufio.NewScanner(os.Stdin)

    fmt.Println("Masukkan bunga, ketik 'SELESAI' untuk selesai:")
    for {
        fmt.Print("Bunga: ")
        scanner.Scan()
        input := scanner.Text()
```

```

        if input == "SELESAI" {
            break
        }

        bunga = append(bunga, input)
    }

    fmt.Println("Pita:", strings.Join(bunga, " - "))
    fmt.Println("Bunga:", len(bunga))
}
fmt.Println(nama)
}

```

Screenshoot Output

```

PROBLEMS 16 OUTPUT TERMINAL PORTS DEBUG CONSOLE

PS D:\ALPRO 2> go run "d:\ALPRO 2\MODUL 2\unguided1.go"
• Masukkan bunga, ketik 'SELESAI' untuk selesai:
Bunga: mawar
Bunga: kamboja
Bunga: sepatu
Bunga: melati
Bunga: matahari
Bunga: SELESAI
Pita: mawar - kamboja - sepatu - melati - matahari
Bunga: 5
○ PS D:\ALPRO 2> 

```

Deskripsi Program

Program ini memungkinkan pengguna memasukkan nama bunga berulang hingga mereka mengetik "SELESAI". Setelah itu, program akan menampilkan daftar bunga yang telah dimasukkan, yang dipisahkan oleh tanda hubung ("-") dan jumlah total bunga yang dimasukkan.

Algoritma untuk program : Slice atau buat array dinamis untuk menyimpan nama bunga. Untuk membaca input pengguna, gunakan `bufio.NewScanner`. Tampilkan pesan yang meminta pengguna mengetik nama bunga. Jika mereka mengetik "SELESAI", hentikan pengisian. Dalam array bunga, masukkan semua bunga yang diberikan. Setelah selesai, gabungkan nama bunga dengan tanda hubung (-) di antara masing-masing bunga, lalu tampilkan daftarnya. Hitung dan tampilkan total bunga. Hentikan program. program sederhana yang menerima nama pengguna sebagai input dan menampilkannya di layar menggunakan paket `fmt`, yang dapat menangani input dan output di Golang.

Cara Program Berfungsi : Program mendeklarasikan variabel bunga sebagai slice, yang menampung daftar bunga dan mempersiapkan scanner untuk membaca input pengguna. Pesan dikeluarkan oleh program yang meminta pengguna menulis nama-nama bunga satu per satu. Dalam slice bunga, setiap input bunga disimpan. Program akan menerima input sampai pengguna mengetik kata "SELESAI". Jika pengguna mengetik "SELESAI", program akan berhenti menerima input dan keluar dari loop. Menampilkan Daftar Bunga: Setelah pengguna memasukkan nama bunga, program menggabungkan semua nama dengan tanda hubung ("-") sebagai pemisah, dan kemudian mencetak daftar. Menghitung Jumlah Bunga : Program ini menghitung dan menampilkan jumlah bunga yang telah dimasukkan oleh pengguna.

2. UNGUIDED 1 – 2B - 2

Soal Studi Case : Setiap hari Pak Andi membawa banyak barang belanjaan dari pasar dengan mengendarai sepeda motor. Barang belanjaan tersebut dibawa dalam kantong terpal di kiri-kanan motor. Sepeda motor tidak akan oleng jika selisih berat barang di kedua kantong sisi tidak lebih dari 9 kg. Buatlah program Pak Andi yang menerima input dua buah bilangan real positif yang menyatakan berat total masing-masing isi kantong terpal. Program akan terus meminta input bilangan tersebut hingga salah satu kantong terpal berisi 9 kg atau lebih. Perhatikan contoh sesi interaksi program seperti di bawah ini (teks bergaris bawah adalah input/read):

```
Masukan berat belanjaan di kedua kantong: 5.5 1.0
Masukan berat belanjaan di kedua kantong: 7.1 8.5
Masukan berat belanjaan di kedua kantong: 2 6
Masukan berat belanjaan di kedua kantong: 9 5.8
Proses selesai.
```

Pada modifikasi program tersebut, program akan menampilkan true jika selisih kedua isi kantong lebih dari atau sama dengan 9 kg. Program berhenti memproses apabila total berat isi kedua kantong melebihi 150 kg atau salah satu kantong beratnya negatif. Perhatikan contoh sesi interaksi program seperti di bawah ini (teks bergaris bawah adalah input/read):

```
Masukan berat belanjaan di kedua kantong: 5 10
Sepeda motor pak Andi akan oleng: false
Masukan berat belanjaan di kedua kantong: 55.6 70.2
Sepeda motor pak Andi akan oleng: true
Masukan berat belanjaan di kedua kantong: 72.3 66.9
Sepeda motor pak Andi akan oleng: false
Masukan berat belanjaan di kedua kantong: 59.5 98.7
Proses selesai.
```

Sourcecode

```
package main

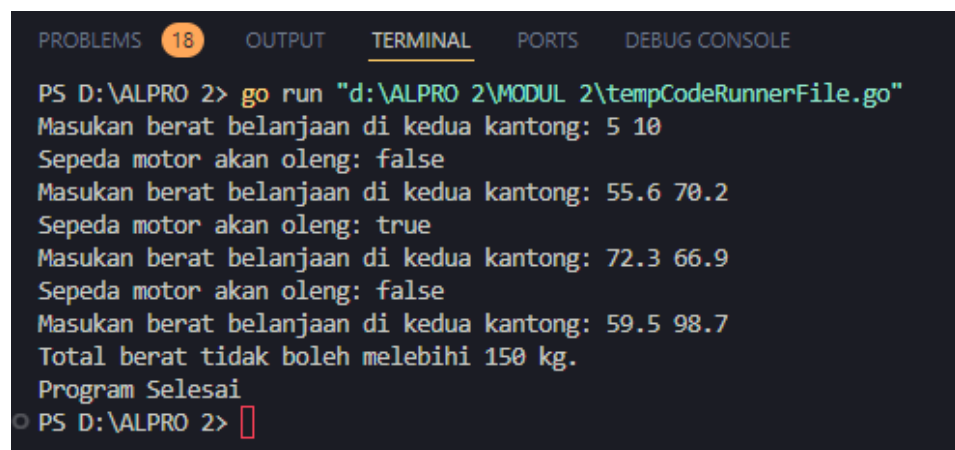
import (
    "fmt"
    "math"
)

func main() {
    var kantong1, kantong2 float64
    for {
        fmt.Print("Masukan berat belanjaan di kedua kantong: ")
        fmt.Scanln(&kantong1, &kantong2)

        total := kantong1 + kantong2

        if total > 150 {
            fmt.Println("Total berat tidak boleh melebihi 150 kg.")
            fmt.Println("Program Selesai")
            break // Keluar dari loop dan program selesai
        } else if kantong1 < 0 || kantong2 < 0 {
            fmt.Println("Berat tidak boleh negatif.")
        } else if math.Abs(kantong1-kantong2) < 9 {
            fmt.Println("Sepeda motor akan oleng: false")
        } else {
            fmt.Println("Sepeda motor akan oleng: true")
        }
    }
}
```

Screenshoot Output



The screenshot shows a terminal window with the following output:

```
PROBLEMS 18 OUTPUT TERMINAL PORTS DEBUG CONSOLE
PS D:\ALPRO 2> go run "d:\ALPRO 2\MODUL 2\tempCodeRunnerFile.go"
Masukan berat belanjaan di kedua kantong: 5 10
Sepeda motor akan oleng: false
Masukan berat belanjaan di kedua kantong: 55.6 70.2
Sepeda motor akan oleng: true
Masukan berat belanjaan di kedua kantong: 72.3 66.9
Sepeda motor akan oleng: false
Masukan berat belanjaan di kedua kantong: 59.5 98.7
Total berat tidak boleh melebihi 150 kg.
Program Selesai
PS D:\ALPRO 2> 
```


Deskripsi Program

Program ini dirancang untuk memeriksa keseimbangan berat belanjaan yang dimasukkan ke dalam dua kantong sepeda motor. Ini mengumpulkan berat di kedua kantong dan menghitung apakah sepeda motor akan oleng (miring) berdasarkan perbedaan berat di antara mereka. Jika input berat total melebihi 150 kg atau jika ada input berat negatif, program akan berhenti.

Algoritma program : Buat variabel kantong 1, dan variabel kantong 2, untuk menyimpan berat belanjaan di masing-masing kantong. Jalankan loop untuk menerima input berat belanjaan secara berulang. Ini dapat dilakukan dengan mendapatkan input berat dua kantong dari pengguna. Hitung berat total kedua kantong. Tampilkan pesan "Berat tidak boleh negatif" di salah satu kantong dan hentikan program jika total berat melebihi 150 kg, "Total berat tidak boleh melebihi 150 kg". Jika berat antara dua kantong kurang dari 9 kg, tampilkan pesan "Sepeda motor akan oleng: benar", yang berarti motor akan oleng, dan jika berat antara dua kantong lebih besar atau sama dengan 9 kg, tampilkan pesan "Sepeda motor akan oleng : benar" Ulangi prosedur di atas hingga persyaratan untuk menghentikan program terpenuhi.

Cara Program Berfungsi: Inisialisasi Variabel untuk menyimpan berat masing-masing kantong, program mendeklarasikan dua variabel, kantong1 dan kantong2. Program memasukkan loop tak terbatas yang akan meminta pengguna memasukkan berat kedua kantong sampai salah satu kondisi keluar terpenuhi. Program menghitung berat kedua kantong dan menghitung totalnya. Jika beratnya melebihi 150 kg, pesan akan ditampilkan dan program akan dihentikan. Jika salah satu berat dimasukkan negatif, program akan menampilkan pesan bahwa berat tidak boleh negatif dan meminta input ulang sambil tetap dalam loop. Program menghitung berat dua kantong. Program menampilkan pesan bahwa sepeda motor akan oleng jika selisihnya kurang dari 9 kg. Jika selisihnya lebih dari 9 kg, program menampilkan pesan bahwa sepeda motor akan oleng. Program tidak berhenti meminta input sampai kondisi berat lebih dari 150 kg memaksa keluar dari loop dan menghentikan program.

3. UNGUIDED 3 – 2B - 4

Soal Studi Case :

Diberikan sebuah persamaan sebagai berikut ini.

$$f(k) = \frac{(4k + 2)^2}{(4k + 1)(4k + 3)}$$

Buatlah sebuah program yang menerima input sebuah bilangan sebagai K, kemudian menghitung dan menampilkan nilai f(K) sesuai persamaan di atas. Perhatikan contoh sesi interaksi program seperti di bawah ini (teks bergaris bawah adalah input/read) :

```
Nilai K = 100
Nilai f(K) = 1.0000061880
```

Sourcecode

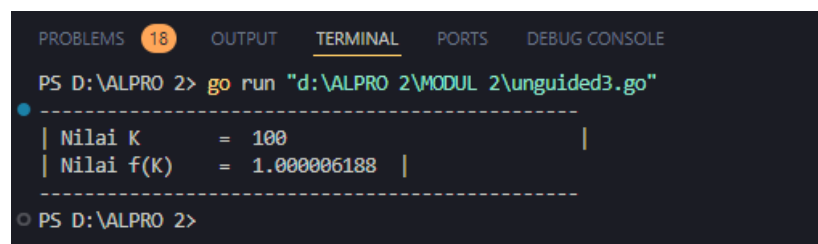
```
package main

import "fmt"

func main() {
    var k int = 100
    var f float64 = 1.0000061880

    fmt.Println("-----")
    fmt.Println("| Nilai K\t= ", k, " \t\t\t|")
    fmt.Println("| Nilai f(K)\t= ", f, " |")
    fmt.Println("-----")
}
```

Screenshoot Output



```
PROBLEMS 18 OUTPUT TERMINAL PORTS DEBUG CONSOLE
PS D:\ALPRO 2> go run "d:\ALPRO 2\MODUL 2\unguided3.go"
-----
| Nilai K      = 100                                |
| Nilai f(K)   = 1.0000061880                        |
-----
PS D:\ALPRO 2>
```

Deskripsi Program

Program ini adalah aplikasi sederhana yang dapat menampilkan dua variabel, yaitu k yang bertipe integer dan f yang bertipe float, dalam tabel dengan format tertentu. Program ini hanya mencetak nilai variabel tersebut dalam tampilan yang rapi dengan menggunakan tanda garis dan tabulasi.

Algoritma untuk Program : Deklarasikan variabel k, yang memiliki tipe data integer, dan nilai awal 100. Deklarasikan juga variabel f, yang memiliki tipe data float64 dan nilai awal 1.0000061880. Cetak header dan footer tabel

yang terdiri dari garis pemisah. Cetak nilai variabel k dan f dalam tabel, dengan nilai k di kolom pertama dan nilai f di kolom kedua.

Cara Program Berfungsi : Dua variabel diidentifikasi oleh program f adalah float64 dengan nilai 1.0000061880. k adalah integer dengan nilai 100. Kemudian, program menggunakan `fmt.Println` untuk mencetak garis horizontal sebagai batas atas tabel. Kemudian, program mencetak dua baris tabel dengan nilai variabel k dan f. Program mencetak Nilai K = 100 dan tabulasi untuk merapikan kolomnya pada baris pertama. Pada baris kedua, program mencetak Nilai $f(K) = 1.0000061880$. Program menggunakan `fmt` untuk menutup tabel dengan garis pemisah lagi setelah nilai ditampilkan.

4. UNGUIDED 4 – 2C - 1

Soal Studi Case :

PT POS membutuhkan aplikasi perhitungan biaya kirim berdasarkan berat parcel. Maka, buatlah program `BiayaPos` untuk menghitung biaya pengiriman tersebut dengan ketentuan sebagai berikut! Dari berat parcel (dalam gram), harus dihitung total berat dalam kg dan sisanya (dalam gram). Biaya jasa pengiriman adalah Rp. 10.000,- per kg. Jika sisa berat tidak kurang dari 500 gram, maka tambahan biaya kirim hanya Rp. 5,- per gram saja. Tetapi jika kurang dari 500 gram, maka tambahan biaya akan dibebankan sebesar Rp. 15,- per gram. Sisa berat (yang kurang dari 1kg) digratiskan biayanya apabila total berat ternyata lebih dari 10kg.

Perhatikan contoh sesi interaksi program seperti di bawah ini (teks bergaris bawah adalah input/read) :

1	<p>Contoh #1</p> <p>Berat parcel (gram): <u>8500</u></p> <p>Detail berat: 8 kg + 500 gr</p> <p>Detail biaya: Rp. 80000 + Rp. 2500</p> <p>Total biaya: Rp. 82500</p>
2	<p>Contoh #2</p> <p>Berat parcel (gram): <u>9250</u></p> <p>Detail berat: 9 kg + 250 gr</p> <p>Detail biaya: Rp. 90000 + Rp. 3750</p> <p>Total biaya: Rp. 93750</p>
3	<p>Contoh #3</p> <p>Berat parcel (gram): <u>11750</u></p> <p>Detail berat: 11 kg + 750 gr</p> <p>Detail biaya: Rp. 110000 + Rp. 3750</p> <p>Total biaya: Rp. 110000</p>

Sourcecode

```
package main

import (
    "fmt"
)

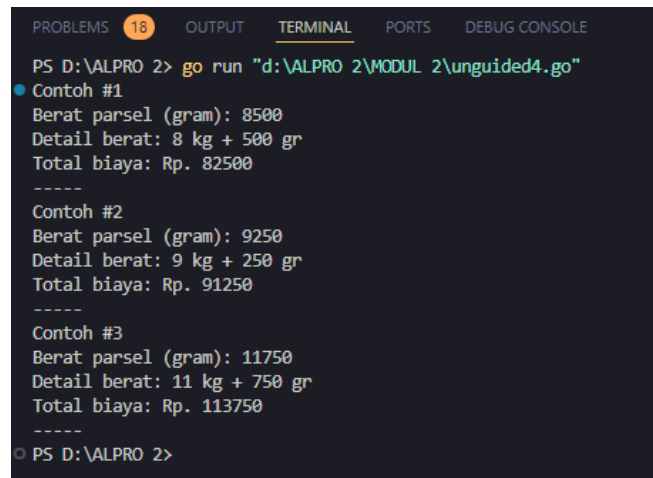
func main() {
    // Contoh data parsel
    parcels := []map[string]interface{}{
        {"berat": 8500, "detailBerat": []int{8, 500}},
        {"berat": 9250, "detailBerat": []int{9, 250}},
        {"berat": 11750, "detailBerat": []int{11,
750}},
    }

    // Hitung biaya untuk setiap parsel
    for i, parcel := range parcels {
        totalBiaya :=
hitungBiayaParcel(parcel["berat"].(int))
        parcels[i]["totalBiaya"] = totalBiaya
        fmt.Printf("Contoh #%d\n", i+1)
        fmt.Printf("Berat   parsel   (gram):   %d\n",
parcel["berat"])
        fmt.Printf("Detail berat: %d kg + %d gr\n",
parcel["detailBerat"].([]int)[0],
parcel["detailBerat"].([]int)[1])
        fmt.Printf("Total biaya: Rp. %d\n", totalBiaya)
        fmt.Println("-----")
    }
}

// Fungsi untuk menghitung biaya parsel berdasarkan berat
func hitungBiayaParcel(berat int) int {
    biayaPerKg := 10000
    biayaTambahanPerGram := 5
    beratKg := berat / 1000
    sisaGram := berat % 1000

    biaya := (beratKg * biayaPerKg) + (sisaGram *
biayaTambahanPerGram)
    return biaya
}
```

Screenshoot Output

A screenshot of a terminal window with a dark background. At the top, there are tabs for 'PROBLEMS' (with a count of 18), 'OUTPUT', 'TERMINAL' (which is active), 'PORTS', and 'DEBUG CONSOLE'. The terminal shows the command 'go run "d:\ALPRO 2\MODUL 2\unguided4.go"' being executed. The output displays three examples of parcel calculations. Each example shows the parcel weight in grams, the breakdown into kilograms and grams, and the total cost in Indonesian Rupiah (Rp.).

```
PS D:\ALPRO 2> go run "d:\ALPRO 2\MODUL 2\unguided4.go"
Contoh #1
Berat parcel (gram): 8500
Detail berat: 8 kg + 500 gr
Total biaya: Rp. 82500
-----
Contoh #2
Berat parcel (gram): 9250
Detail berat: 9 kg + 250 gr
Total biaya: Rp. 91250
-----
Contoh #3
Berat parcel (gram): 11750
Detail berat: 11 kg + 750 gr
Total biaya: Rp. 113750
-----
PS D:\ALPRO 2>
```

Deskripsi Program

Program ini dibuat untuk menghitung biaya pengiriman parcel berdasarkan beratnya. Itu mengelola data parcel dengan berat dalam gram dan detail berat dalam gram. Setelah menghitung biaya setiap parcel, program mencetak informasi yang relevan seperti berat parcel, detail berat, dan total biaya pengiriman.

Algoritma untuk Program : Buat slice dari map untuk menyimpan informasi tentang parcel, seperti berat dalam gram dan detail berat. Untuk menghitung biaya pengiriman, putar melalui setiap parcel dalam data parcel. Untuk menghitung biaya total, ambil nilai berat dari setiap parcel dan gunakan fungsi `hitungBiayaParcel`. Simpan biaya total ke dalam map parcel yang sesuai. Cetak nomor contoh, berat, berat detail, dan total biaya parcel. Definisi Fungsi Hitung Biaya : Buat fungsi untuk menghitung biaya parcel. Fungsi Hitung Biaya didefinisikan sebagai berikut : Buat fungsi untuk menghitung biaya barang dengan berat dalam gram dan menghitung biaya berdasarkan tarif per kilogram dan tarif tambahan per gram. Kalkulasi Biaya: Dalam fungsi ini, hitung biaya total dengan menghitung biaya berdasarkan berat dalam gram. Menambahkan biaya yang dihitung dengan sisa berat dalam gram. Hasil Output : Cetak hasil perhitungan untuk setiap parcel.

Cara Program Berfungsi : Data parcel didefinisikan dalam bentuk slice map dengan berat parcel dalam gram dan detail berat dalam gram. Program melakukan iterasi pada setiap parcel. Untuk melakukan ini, program memanggil fungsi `hitungBiayaParcel` dan menyimpan biaya total ke dalam map parcel. Kalkulasi Biaya fungsi ini menghitung biaya pengiriman berdasarkan beratnya : Biaya tambahan per gram adalah Rp. 10.000. Biaya

per kilogram adalah Rp. 10.000, dan biaya per gram adalah Rp. 5. Anda dapat menghitung total biaya dengan menambahkan biaya per kilogram dan gram. Menampilkan Output: Program mencetak informasi setiap parsel ke konsol dalam format yang terstruktur, termasuk berat, detail berat, dan total biaya.

5. UNGUIDED 5 – 2C - 3

Soal Studi Case : Sebuah bilangan bulat b memiliki faktor bilangan $f > 0$ jika f habis membagi b . Contoh: 2 merupakan faktor dari bilangan 6 karena 6 habis dibagi 2. Buatlah program yang menerima input sebuah bilangan bulat b dan $b > 1$. Program harus dapat mencari dan menampilkan semua faktor dari bilangan tersebut!

Perhatikan contoh sesi interaksi program seperti di bawah ini (teks bergaris bawah adalah input/read):

Bilangan: 12	Bilangan: 7
Faktor: 1 2 3 4 6 12	Faktor: 1 7

Bilangan bulat $b > 0$ merupakan bilangan prima p jika dan hanya jika memiliki persis dua faktor bilangan saja, yaitu 1 dan dirinya sendiri. Lanjutkan program sebelumnya. Setelah menerima masukan sebuah bilangan bulat $b > 0$. Program tersebut mencari dan menampilkan semua faktor bilangan tersebut. Kemudian, program menentukan apakah b merupakan bilangan prima. Perhatikan contoh sesi interaksi program seperti di bawah ini (teks bergaris bawah adalah input/read) :

Bilangan: 12	Bilangan: 7
Faktor: 1 2 3 4 6 12	Faktor: 1 7
Prima: false	Prima: true

Sourcecode

```
package main

import (
    "bufio"
    "fmt"
    "os"
    "strings"
)

func main() {
    var bunga []string
    scanner := bufio.NewScanner(os.Stdin)
```

```

        fmt.Println("Masukkan bunga, ketik 'SELESAI' untuk
selesai:")
        for {
            fmt.Print("Bunga: ")
            scanner.Scan()
            input := scanner.Text()

            if input == "SELESAI" {
                break
            }

            bunga = append(bunga, input)
        }

        fmt.Println("Pita:", strings.Join(bunga, " - "))
        fmt.Println("Bunga:", len(bunga))
    }
    fmt.Println(nama)
}

```

Screenshoot Output

```

PROBLEMS 18 OUTPUT TERMINAL PORTS DEBUG CONSOLE
PS D:\ALPRO 2> go run "d:\ALPRO 2\MODUL 2\unguided5.go"
Bilangan: 12
Faktor: 1 2 3 4 6 12
Prima: false
PS D:\ALPRO 2> go run "d:\ALPRO 2\MODUL 2\unguided5.go"
Bilangan: 7
Faktor: 1 7
Prima: true
PS D:\ALPRO 2> 

```

Deskripsi Program

Program ini meminta pengguna untuk memasukkan sebuah bilangan bulat. Setelah menerima input, program akan menghitung semua faktor dari bilangan tersebut dan mengetahui apakah bilangan tersebut adalah bilangan prima. Setelah itu, hasilnya ditampilkan di layar.

Algoritma program memerlukan input angka : Menampilkan faktor, program meminta pengguna untuk memasukkan bilangan bulat dan menyimpannya dalam variabel b : Program melakukan iterasi dari nilai 1 hingga b. Selama setiap iterasi, program memeriksa apakah i dapat dibagi habis oleh b. Jika hasilnya adalah ya, maka i dicetak sebagai faktor dari b. Lihat Jumlah Prima : Variabel prima dengan nilai benar diinisialisasi oleh program. Karena bilangan negatif, 0, dan 1 bukan merupakan bilangan prima, variabel prima diubah menjadi false jika b kurang dari atau sama dengan 1. Untuk bilangan lebih dari 1, program melakukan pengecekan dengan iterasi dari 2 hingga akar kuadrat b. Jika ditemukan pembagi (yaitu,

b dapat dibagi habis oleh salah satu angka dalam rentang tersebut), maka prima diubah menjadi false.

Hasil yang Dihasilkan : Program mendeklarasikan variabel b untuk menyimpan input pengguna dan mencetak hasil yang menunjukkan apakah bilangan b adalah bilangan prima berdasarkan nilai dari variabel prima. Kemudian, program meminta pengguna untuk memasukkan nilai dengan perintah `fmt.Scanln(&b)`. Setelah menerima input, program mencetak "Faktor:" dan mulai iterasi dari 1 hingga b. Dalam setiap iterasi, program mengecek apakah b dapat dibagi oleh i, dan jika pembagian menghasilkan sisa 0, maka i dicetak sebagai faktor. Setelah menampilkan semua faktor, program mengecek apakah b adalah bilangan prima. Program menetapkan prima menjadi false jika b kurang dari atau sama dengan 1. Untuk bilangan di atas 1, program melakukan loop dari 2 hingga akar kuadrat b untuk mencari pembagi. Jika ditemukan, prima diatur menjadi false. Terakhir, sistem mencetak nilai variabel prima untuk mengetahui apakah angka tersebut merupakan bilangan prima.

IV. KESIMPULAN

Dalam bahasa pemrograman Go, setiap file harus diatur dalam sebuah paket, dengan setidaknya satu file yang berisi paket main, yang berfungsi sebagai titik awal untuk menjalankan program. Untuk memuat package lain, seperti `fmt`, yang sering digunakan untuk fungsi input/output teks, perintah `import` harus digunakan, seperti `fmt.Println()` dan `fmt.Printf()`. Go mendukung berbagai jenis data, termasuk string, nilai boolean, `uint`, `int`, bilangan bulat positif, dan bilangan desimal (`float32`, `float64`). Di Go, kata kunci `for` dapat digunakan untuk pengulangan dalam bentuk loop seperti `while-loop` atau `repeat-until`. Sebaliknya, `var` dapat digunakan untuk mendeklarasikan variabel baru, sementara konstanta dapat digunakan untuk menyimpan nilai yang tetap, seperti π . Bentuk `if-else` dan `switch-case` tersedia dalam Go untuk struktur percabangan. Go menawarkan bentuk `if-else` dan `switch-case` untuk struktur percabangan. `Switch` dapat digunakan dengan atau tanpa ekspresi, memberikan fleksibilitas dalam penulisan logika kondisional.

V. REFERENSI

- [1] Modul 2 Praktikum Algoritma 2
- [2] <https://codingstudio.id/blog/golang-adalah/>
- [3] <https://www.staditek.id/insight/golang-adalah/>