

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL 2
REVIEW STRUKTUR KONTROL**



Disusun Oleh:

Boutefhika Nuha Ziyadatul Khair/2311102316

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

A. Struktur Program Go

- **Penggunaan Keyword package**
Setiap file program harus memiliki package. Setiap project harus ada minimal satu file dengan nama package main. File yang ber-package main, akan dieksekusi pertama kali ketika program di jalankan.
- **Penggunaan Keyword import**
Keyword import digunakan untuk meng-import atau memasukan package lain ke dalam file program, agar isi dari package yang di-import bisa dimanfaatkan.
Package fmt merupakan salah satu package bawaan yang disediakan oleh Go, isinya banyak fungsi untuk keperluan I/O yang berhubungan dengan text.
- **Keyword var digunakan untuk membuat variabel baru.**
- **Penggunaan Fungsi main()**
Dalam sebuah proyek harus ada file program yang di dalamnya berisi sebuah fungsi bernama main(). Fungsi tersebut harus berada di file yang package-nya bernama main.
- **Penggunaan Fungsi fmt.Println()**
Fungsi fmt.Println() digunakan untuk memunculkan text ke layar.
Fungsi fmt.Println() berada dalam package fmt, maka untuk menggunakannya perlu package tersebut untuk di-import terlebih dahulu.
Fungsi fmt.Println() dapat menampung parameter yang tidak terbatas jumlahnya. Semua data parameter akan dimunculkan dengan pemisah tanda spasi.
- **Fungsi fmt.Printf()**
Fungsi ini digunakan untuk menampilkan output dalam bentuk tertentu. Kegunaannya sama seperti fungsi fmt.Println(), hanya saja struktur outputnya didefinisikan di awal.

B. Tipe Data dan Intruksi Dasar

Tipe Data

Go mengenal beberapa jenis tipe data, di antaranya adalah tipe data numerik (desimal & non-desimal), string, dan boolean.

- **Tipe Data Numerik Non-Desimal**
Tipe data numerik non-desimal atau non floating point di Go ada beberapa jenis. Secara umum ada 2 tipe data kategori ini yang perlu diketahui.
 - uint, tipe data untuk bilangan cacah (bilangan positif).
 - int, tipe data untuk bilangan bulat (bilangan negatif dan positif).
- **Tipe Data Numerik Desimal**

Tipe data numerik desimal yang perlu diketahui ada 2, float32 dan float64. Perbedaan kedua tipe data tersebut berada di lebar cakupan nilai desimal yang bisa ditampung.

- Tipe Data bool (Boolean)
Tipe data bool berisikan hanya 2 variansi nilai, true dan false. Tipe data ini biasa dimanfaatkan dalam seleksi kondisi dan perulangan
- Tipe Data string
Ciri khas dari tipe data string adalah nilainya di apit oleh tanda quote atau petik dua (").
- Nil untuk alamat memori
- Karakter NUL (lihat tabel ASCII) untuk karakter
- 0.0E+0 untuk bilangan real

Konstanta Simbolik

Konstanta dapat diberi nama untuk memudahkan mengingat maksud dan manfaat dari nilai yang diberi nama tersebut. Seperti PI untuk merepresentasikan konstanta π .

C. 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.

Dalam konsep pemrograman terstruktur, setiap rancangan algoritma harus memenuhi syarat satu pintu masuk dan satu pintu keluar. Karena itu tidaklah diperkenankan untuk membuat program sumber yang mempunyai struktur loop yang mempunyai pintu keluar lebih dari satu, seperti:

- Satu pintu keluar dari kondisi for dan satu lagi dari instruksi If-break
- Atau mempunyai instruksi If-break yang lebih dari satu.

1. Bentuk While-Loop

Bentuk while-loop memastikan setiap kali memasuki loop, ada kondisi yang harus terpenuhi (benar/true). Ini juga berarti saat keluar dari loop, maka nilai kondisi tersebut pasti salah/false!

2. 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!

D. Struktur Kontrol Percabangan

1. Bentuk If-Else

Berikut ini bentuk-bentuk if-else 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

2. Bentuk Swich-Case

Dalam bahasa Go ada dua variasi bentuk switch-case. 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 (atau alias dari) susunan suatu if-elseif...-else-endif.

II. GUIDED

1. Program menginput nama

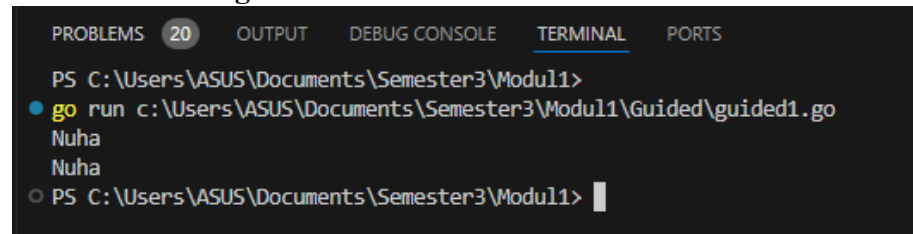
Sourcecode

```
package main

import (
    "fmt"
)

func main() {
    var nama string
    fmt.Scanln(&nama)
    fmt.Println(nama)
}
```

Screenshoot Program



Deskripsi Program

Program diatas membaca input dari pengguna berupa nama dan kemudian mencetaknya kembali. Pertama, variabel **`nama`** bertipe string digunakan untuk menyimpan input yang diambil dari pengguna melalui fungsi **`fmt.Scanln(&nama)`**. Setelah pengguna memasukkan nama dan menekan Enter, program akan mencetak nama tersebut menggunakan **`fmt.Println(nama)`**. Misalnya, jika pengguna memasukkan "Nuha", program akan menampilkan "Nuha" sebagai output.

2. Program penjumlahan 5 bilangan bulat

Sourcecode

```
package main

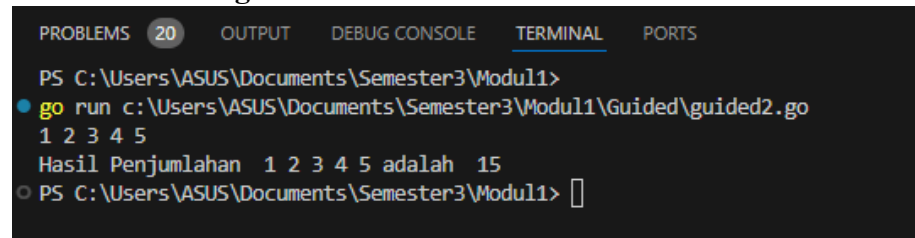
import (
    "fmt"
)

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 Program



```
PROBLEMS 20 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\ASUS\Documents\Semester3\Modul1>
go run c:\Users\ASUS\Documents\Semester3\Modul1\Guided\guided2.go
1 2 3 4 5
Hasil Penjumlahan 1 2 3 4 5 adalah 15
PS C:\Users\ASUS\Documents\Semester3\Modul1>
```

Deskripsi Program

Program diatas berfungsi untuk penjumlahan dari lima bilangan bulat yang dimasukkan oleh pengguna, lalu menampilkan hasilnya. Pertama, lima variabel `a`, `b`, `c`, `d`, dan `e` bertipe integer dideklarasikan untuk menyimpan input. Program membaca lima bilangan dari pengguna dengan `fmt.Scanln(&a, &b, &c, &d, &e)`, kemudian menjumlahkan bilangan-bilangan tersebut dan menyimpan hasilnya dalam variabel `hasil`. Setelah itu, program mencetak semua bilangan yang diinput dan menampilkan hasil penjumlahan mereka. Misalnya, jika pengguna memasukkan angka 1, 2, 3, 4, dan 5, output yang dihasilkan adalah: "Hasil Penjumlahan 1 2 3 4 5 adalah 15".

3. 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 berturutan 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. Kemuan program akan menampilkan true apabila urutan warna sesuai dengan informasi yang diberikan pada paragraf sebelumnya, dan false untuk urutan warna lainnya.

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, " ")

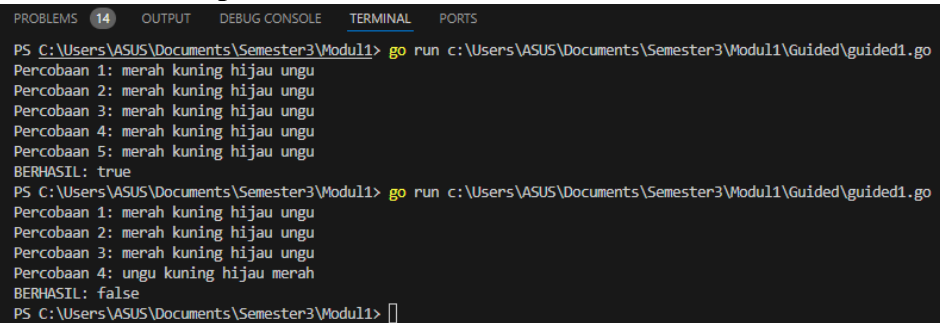
        // Mengecek apakah urutan warna sesuai
        for j := 0; j < 4; j++ {
            if colors[j] != correctOrder[j] {
                success = false
                break
            }
        }

        // 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")
    }
}

```

Screenshoot Output



```

PROBLEMS 14 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\ASUS\Documents\Semester3\Modul1> go run c:\Users\ASUS\Documents\Semester3\Modul1\Guided\guided1.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:\Users\ASUS\Documents\Semester3\Modul1> go run c:\Users\ASUS\Documents\Semester3\Modul1\Guided\guided1.go
Percobaan 1: merah kuning hijau ungu
Percobaan 2: merah kuning hijau ungu
Percobaan 3: merah kuning hijau ungu
Percobaan 4: ungu kuning hijau merah
BERHASIL: false
PS C:\Users\ASUS\Documents\Semester3\Modul1>

```

Deskripsi Program

Program diatas mengecek apakah pengguna memasukkan urutan warna yang benar dalam lima percobaan. Pertama, program mendefinisikan urutan warna yang benar, yaitu "merah", "kuning", "hijau", dan "ungu". Kemudian, pengguna diminta untuk memasukkan urutan warna pada setiap percobaan, yang dibaca sebagai string menggunakan ``bufio.NewReader``. String input dipisahkan berdasarkan spasi untuk mendapatkan setiap warna secara individual. Program membandingkan urutan warna yang dimasukkan dengan urutan yang benar. Jika pada salah satu percobaan urutan tidak sesuai, program langsung menghentikan proses dan menandai percobaan sebagai gagal. Setelah semua percobaan selesai, program menampilkan "BERHASIL: true" jika semua percobaan berhasil, atau "BERHASIL: false" jika ada yang salah. Contoh output jika pengguna memasukkan urutan yang benar untuk lima kali adalah "BERHASIL: true".

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

NAM	NMK
NAM > 80	A
72.5 < NAM <= 80	AB
65 < NAM <= 72.5	B
57.5 < NAM <= 65	BC
50 < NAM <= 57.5	C
40 < NAM <= 50	D
NAM <= 40	E

Program berikut menerima input sebuah bilangan riil yang menyatakan NAM. Program menghitung NMK dan menampilkannya.

```
package main

import "fmt"

func main() {
    var nam float32
    var nmk string

    fmt.Print("Masukan nilai :")
    fmt.Scan(&nam)

    if nam > 80 {
        nmk = "A"
    } else if nam > 72.5 {
        nmk = "B"
    } else if nam > 65 {
        nmk = "C"
    } else if nam > 50 {
        nmk = "D"
    }
```



```

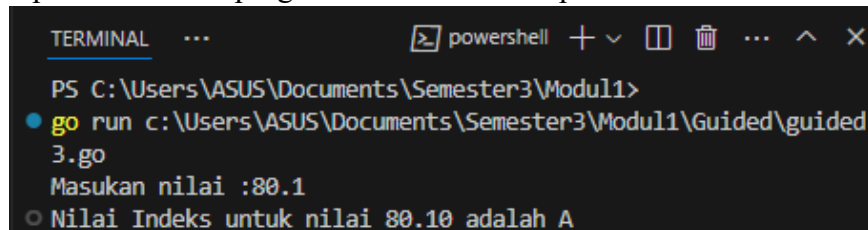
    } else if nam > 40 {
        nmk = "E"
    } else {
        nmk = "F"
    }

    fmt.Printf("Nilai Indeks untuk nilai %.2f adalah %s\n", nam, nmk)
}

```

Jawablah pertanyaan-pertanyaan berikut:

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



```

TERMINAL ... powershell
PS C:\Users\ASUS\Documents\Semester3\Modul1>
go run c:\Users\ASUS\Documents\Semester3\Modul1\Guided\guided3.go
Masukan nilai :80.1
Nilai Indeks untuk nilai 80.10 adalah A

```

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

Masalah Batasan Nilai: Ada kesalahan dalam penentuan rentang nilai, seperti kondisi `nam > 80` yang menyebabkan nilai 80.0 tidak masuk kategori "A". Solusinya, gunakan operator `>=` agar batas bawah tercakup dengan tepat, misalnya nilai 80.0 harus termasuk kategori "A".

Penggunaan Float32: Float32 tidak diperlukan, lebih baik menggunakan float64 karena lebih umum dan lebih presisi.

Alur Program Seharusnya: Nilai `nam` harus diproses dengan rentang yang tepat, di mana batas bawah dari tiap kategori diikuti dengan `>=`. Contohnya, nilai 80.0 termasuk dalam kategori "A". Alur seharusnya dimulai dari kondisi nilai tertinggi (A), kemudian memeriksa nilai berikutnya secara bertahap ke bawah hingga "F".

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

Sourcecode

```

package main

import "fmt"

func main() {
    var nam float64
    var nmk string

```

```

    fmt.Print("Masukan nilai: ")
    fmt.Scan(&nam)

    // Mengatur rentang nilai dengan
    operator >= untuk batas bawah
    if nam >= 80 {
        nmk = "A"
    } else if nam >= 70 {
        nmk = "B"
    } else if nam >= 60 {
        nmk = "C"
    } else if nam >= 50 {
        nmk = "D"
    } else if nam >= 40 {
        nmk = "E"
    } else {
        nmk = "F"
    }

    fmt.Printf("Nilai Indeks untuk nilai
    %.2f adalah %s\n", nam, nmk)
}

```

Sreenshoot Output

```

PS C:\Users\ASUS\Documents\Semester3\Modul1> go run c:\Users\ASUS\Documents\Semester3\Modul1
Masukan nilai: 93.5
Nilai Indeks untuk nilai 93.50 adalah A
PS C:\Users\ASUS\Documents\Semester3\Modul1> go run c:\Users\ASUS\Documents\Semester3\Modul1
Masukan nilai: 70.6
Nilai Indeks untuk nilai 70.60 adalah B
PS C:\Users\ASUS\Documents\Semester3\Modul1> go run c:\Users\ASUS\Documents\Semester3\Modul1
Masukan nilai: 49.5
Nilai Indeks untuk nilai 49.50 adalah E
PS C:\Users\ASUS\Documents\Semester3\Modul1> 

```

Deskripsi Program

Program diatas digunakan untuk mengonversi nilai angka menjadi nilai indeks huruf berdasarkan rentang nilai tertentu. Pertama, pengguna diminta untuk memasukkan nilai angka melalui input. Program kemudian memeriksa nilai yang dimasukkan menggunakan serangkaian kondisi `if-else`. Jika nilai lebih besar atau sama dengan 80, indeks yang diberikan adalah "A". Jika nilainya berada dalam rentang 70 hingga kurang dari 80, diberikan indeks "B", dan seterusnya hingga nilai di bawah 40, yang akan mendapatkan indeks "F". Setelah semua kondisi

dicek, program menampilkan hasil berupa nilai indeks yang sesuai dengan format dua angka desimal untuk nilai input. Sebagai contoh, jika pengguna memasukkan nilai 75.5, program akan mengeluarkan hasil: "Nilai Indeks untuk nilai 75.50 adalah B".

III. UNGUIDED

1. Suatu pita (string) berisi kumpulan nama-nama bunga yang dipisahkan oleh spasi dan '-', contoh pita diilustrasikan seperti berikut ini.

Pita: mawar-melati-tulip-teratal-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.

Sourcecode

```
package main

import (
    "fmt"
)

func main() {
    var n int          // Variabel untuk menyimpan
    jumlah bunga yang akan dimasukkan
    var pita string // Variabel untuk menyimpan
    nama-nama bunga yang digabungkan

    fmt.Print("N: ") // Meminta pengguna memasukkan
    jumlah bunga
    fmt.Scan(&n)      // Membaca input jumlah bunga
    dari pengguna

    // Loop untuk mengumpulkan nama bunga sebanyak
    n kali
    for i := 1; i <= n; i++ {
        var bunga string          // Variabel
        untuk menyimpan nama bunga yang dimasukkan
        fmt.Printf("Bunga %d: ", i) //
        Menampilkan prompt untuk memasukkan nama bunga
        fmt.Scan(&bunga)           // Membaca
        nama bunga dari pengguna

        // Menambahkan nama bunga ke dalam pita
        if pita == "" { // Jika pita masih kosong
            (bunga pertama)
            pita = bunga // Assign nama bunga
            ke pita
        }
    }
}
```

```

        } else { // Jika pita sudah berisi bunga
lain
            pita += " - " + bunga // Tambahkan
nama bunga baru dengan pemisah '-'
        }
    }

    fmt.Printf("Pita: %s -\n", pita) // Menampilkan
isi pita dengan tanda '-' di akhir
}

```

Modifikasi program sebelumnya, proses input akan berhenti apabila user mengetikkan 'SELESAI'. Kemudian tampilkan isi pita beserta banyaknya bunga yang ada di dalam pita

Sourcecode

```

package main

import (
    "fmt"
    "strings"
)

func main() {
    var pita string // Variabel untuk menyimpan nama-
nama bunga yang digabungkan
    var n int       // Variabel untuk menghitung jumlah
bunga yang dimasukkan

    // Loop tak terbatas untuk menerima input nama
bunga
    for {
        var bunga string // Variabel untuk
menyimpan nama bunga yang dimasukkan
        fmt.Printf("Bunga %d: ", n+1) // Menampilkan
prompt untuk memasukkan nama bunga
        fmt.Scan(&bunga) // Membaca input
nama bunga dari pengguna

        // Mengecek apakah input adalah "SELESAI"
        if strings.ToUpper(bunga) == "SELESAI" {
            break // Menghentikan input jika user
mengetikkan "SELESAI"
        }

        // Menambahkan nama bunga ke dalam pita
        if pita == "" {
            pita = bunga // Jika pita kosong, assign
bunga yang baru dimasukkan
        } else {
            pita += " - " + bunga // Jika tidak kosong,
tambahkan dengan spasi sebagai pemisah
        }
        n++ // Menambah jumlah bunga
    }
}

```

```

        // Menampilkan isi pita dan banyaknya bunga
        fmt.Printf("Pita: %s -\n", pita) // Menampilkan
        semua nama bunga yang dimasukkan
        fmt.Printf("Bunga: %d\n", n)      // Menampilkan
        jumlah bunga yang dimasukkan
    }

```

Screenshoot Output

Sebelum dimodifikasi

```

PS C:\Users\ASUS\Documents\Semester3\Modul1>
go run c:\Users\ASUS\Documents\Semester3\Modul1\Unguided\Unguided1a.go
N: 3
Bunga 1: Kertas
Bunga 2: Mawar
Bunga 3: Tulip
Pita: Kertas - Mawar - Tulip -
PS C:\Users\ASUS\Documents\Semester3\Modul1> go run c:\Users\ASUS\Documents\Semester3\Modul1\Unguided\Unguided1a.go
N: 0
Pita: -
PS C:\Users\ASUS\Documents\Semester3\Modul1>

```

Setelah dimodifikasi

```

PS C:\Users\ASUS\Documents\Semester3\Modul1> go run c:\Users\ASUS\Documents\Semester3\Modul1\Unguided\Unguided1b.go
Bunga 1: Kertas
Bunga 2: Mawar
Bunga 3: Tulip
Bunga 4: SELESAI
Pita: Kertas - Mawar - Tulip -
Bunga: 3
PS C:\Users\ASUS\Documents\Semester3\Modul1> go run c:\Users\ASUS\Documents\Semester3\Modul1\Unguided\Unguided1b.go
Bunga 1: SELESAI
Pita: -
Bunga: 0
PS C:\Users\ASUS\Documents\Semester3\Modul1>

```

Deskripsi Program

Program pertama berfungsi untuk mengumpulkan nama-nama bunga dari pengguna sebanyak jumlah yang ditentukan dan menggabungkannya menjadi satu string dengan pemisah "-", lalu menampilkan hasilnya. Dalam modifikasi menjadi program kedua, logika diubah untuk menerima input nama bunga secara terus-menerus hingga pengguna mengetikkan "SELESAI", tanpa batasan jumlah bunga yang dimasukkan. Program ini menggunakan loop tak terbatas untuk meminta pengguna memasukkan nama bunga, dan jika input yang diberikan adalah "SELESAI", program akan berhenti mengumpulkan nama bunga. Nama-nama bunga yang dimasukkan akan disimpan dalam variabel `pita` dengan format yang sama, yaitu dipisahkan dengan "-", dan setiap kali nama bunga dimasukkan, jumlah bunga yang diterima akan dihitung. Setelah pengguna selesai memasukkan nama-nama bunga, program menampilkan semua nama bunga yang digabungkan dalam variabel `pita` dan juga jumlah bunga yang dimasukkan. Contoh output program ini mungkin terlihat seperti: "Pita: Mawar - Melati - Anggrek - Bunga: 3", jika pengguna memasukkan tiga nama bunga sebelum mengetik "SELESAI".

2. 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.

Sourcecode

```
package main

import (
    "fmt"
)

func main() {
    var kantongKiri, kantongKanan float64 // Variabel
    untuk menyimpan berat belanjaan di kedua kantong

    for { // Memulai loop tak terbatas
        fmt.Print("Masukkan berat belanjaan di kedua
kantong: ") // Menampilkan prompt untuk memasukkan
berat

        fmt.Scan(&kantongKiri, &kantongKanan) //
Membaca input berat belanjaan dari pengguna

        // Mengecek jika berat di salah satu kantong
        >= 9
        if kantongKiri >= 9 || kantongKanan >= 9 {
            fmt.Println("Proses selesai.") //
Menampilkan pesan dan keluar dari loop
            break
        }

        // Mengecek apakah selisih berat di kedua
kantong > 9
        if kantongKiri-kantongKanan > 9 ||
kantongKanan-kantongKiri > 9 {
            fmt.Println("Sepeda motor bisa oleng!") //
Menampilkan peringatan jika berat tidak seimbang
        }
    }
}
```

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.

Sourcecode

```
package main

import (
    "fmt"
    "math"
)

func main() {
    var kantongKiri, kantongKanan float64 // Variabel
    untuk menyimpan berat belanjaan di kedua kantong

    for { // Memulai loop tak terbatas
        fmt.Print("Masukkan berat belanjaan di kedua
kantong: ") // Menampilkan prompt untuk memasukkan
berat
        fmt.Scan(&kantongKiri, &kantongKanan) //
Membaca input berat belanjaan dari pengguna

        // Mengecek jika berat di salah satu kantong
negatif
        if kantongKiri < 0 || kantongKanan < 0 {
            fmt.Println("Proses selesai.") //
Menampilkan pesan dan keluar dari loop
            break
        }

        // Mengecek jika total berat di kedua kantong
> 150
        if kantongKiri+kantongKanan > 150 {
            fmt.Println("Proses selesai.") //
Menampilkan pesan dan keluar dari loop
            break
        }

        // Menghitung selisih berat antara kedua
kantong
        selisih := math.Abs(kantongKiri - kantongKanan)

        // Mengecek apakah selisih berat >= 9
        if selisih >= 9 {
            fmt.Println("Sepeda motor pak Andi akan
oleng: true") // Menampilkan peringatan jika berat
tidak seimbang
```

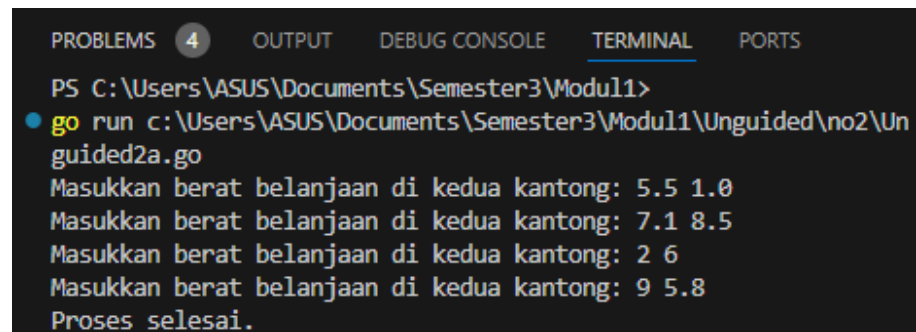
```

        } else {
            fmt.Println("Sepeda motor pak Andi akan
oleng: false") // Menampilkan pesan jika berat seimbang
        }
    }
}

```

Screenshoot Program

Sebelum dimodifikasi

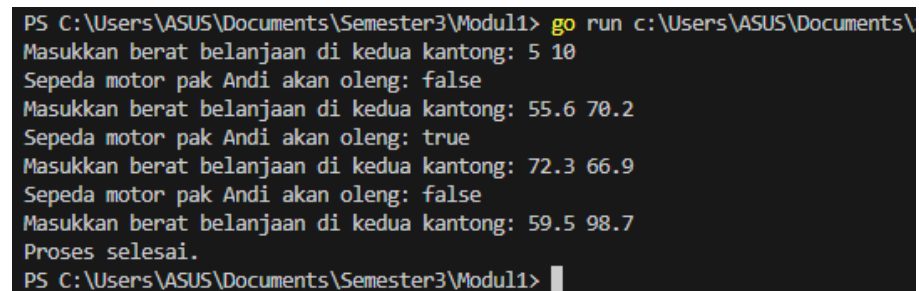


```

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\ASUS\Documents\Semester3\Modul1>
go run c:\Users\ASUS\Documents\Semester3\Modul1\Unguided\Unguided2a.go
Masukkan berat belanjaan di kedua kantong: 5.5 1.0
Masukkan berat belanjaan di kedua kantong: 7.1 8.5
Masukkan berat belanjaan di kedua kantong: 2 6
Masukkan berat belanjaan di kedua kantong: 9 5.8
Proses selesai.

```

Setelah dimodifikasi



```

PS C:\Users\ASUS\Documents\Semester3\Modul1> go run c:\Users\ASUS\Documents\
Masukkan berat belanjaan di kedua kantong: 5 10
Sepeda motor pak Andi akan oleng: false
Masukkan berat belanjaan di kedua kantong: 55.6 70.2
Sepeda motor pak Andi akan oleng: true
Masukkan berat belanjaan di kedua kantong: 72.3 66.9
Sepeda motor pak Andi akan oleng: false
Masukkan berat belanjaan di kedua kantong: 59.5 98.7
Proses selesai.
PS C:\Users\ASUS\Documents\Semester3\Modul1>

```

Deskripsi Program

Program pertama dirancang untuk memeriksa berat belanjaan di dua kantong dan memberi tahu jika salah satu kantong memiliki berat 9 kg atau lebih, atau jika selisih berat antara kedua kantong lebih dari 9 kg, yang dapat menyebabkan sepeda motor menjadi oleng. Program kedua memodifikasi logika ini dengan menambahkan beberapa kondisi baru: jika berat di salah satu kantong negatif atau jika total berat kedua kantong melebihi 150 kg, program akan menampilkan pesan dan menghentikan proses. Selain itu, program ini menggunakan fungsi dari paket ``math`` untuk menghitung selisih absolut antara kedua kantong dan memberikan pesan yang menunjukkan apakah sepeda motor pak Andi akan oleng atau tidak, berdasarkan apakah selisih tersebut lebih besar atau sama dengan 9 kg. Dengan demikian, program ini memberikan informasi lebih lengkap terkait keseimbangan beban di kedua kantong. Jika pengguna memasukkan berat yang tidak memenuhi

kriteria, program akan menampilkan pesan yang sesuai, misalnya "Sepeda motor pak Andi akan oleng: true" jika selisih berat di atas 9, atau "Proses selesai." ketika kondisi akhir terpenuhi.

3. 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 diatas.

Sourcecode

```
package main

import (
    "fmt"
)

func main() {
    var k float64 // Mendeklarasikan variabel k bertipe float64 untuk menyimpan input

    // Meminta input nilai K
    fmt.Print("Nilai K = ") // Menampilkan pesan untuk meminta input dari pengguna
    fmt.Scan(&k) // Membaca input dari pengguna dan menyimpannya ke variabel k

    // Menghitung f(k)
    fk := (4*k + 2) * (4*k + 2) / ((4*k + 1) * (4*k + 3)) // Menghitung nilai f(k) berdasarkan rumus yang diberikan

    // Menampilkan hasil
    fmt.Printf("Nilai f(K) = %.10f\n", fk) // Menampilkan hasil perhitungan dengan format 10 angka di belakang koma
}
```

$\sqrt{2}$ merupakan bilangan irasional. Meskipun demikian, nilai tersebut dapat dihampiri dengan rumus berikut:

$$\sqrt{2} = \prod_{k=0}^{\infty} \frac{(4k + 2)^2}{(4k + 1)(4k + 3)}$$

Modifikasi program sebelumnya yang menerima input integer K dan menghitung $\sqrt{2}$ untuk K tersebut. Hampiran $\sqrt{2}$ dituliskan dalam ketelitian 10 angka dibelakang koma.

Sourcecode

```
package main

import (
    "fmt"
)

func main() {
    var K int // Mendeklarasikan variabel K bertipe int
    untuk menyimpan input

    fmt.Print("Nilai K = ") // Menampilkan pesan untuk
    meminta input dari pengguna
    fmt.Scan(&K) // Membaca input dari
    pengguna dan menyimpannya ke variabel K

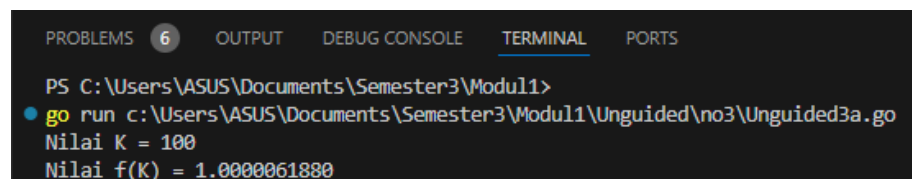
    sqrt2 := 1.0 // Mendeklarasikan variabel sqrt2
    bertipe float64 untuk menyimpan hasil perhitungan akar
    2

    // Melakukan perulangan dari 0 hingga K
    for k := 0; k <= K; k++ {
        k := float64(k) // Mengonversi k menjadi float64
        // Menghitung nilai hasil berdasarkan rumus yang
        diberikan
        hasil := (4*k + 2) * (4*k + 2) / ((4*k + 1) *
        (4*k + 3))
        sqrt2 *= hasil // Mengalikan hasil ke dalam
        variabel sqrt2
    }

    // Menampilkan hasil akhir dengan format 10 angka
    di belakang koma
    fmt.Printf("Nilai akar 2 = %.10f\n", sqrt2)
}
```

Screenshoot Program

Sebelum dimodifikasi



```
PS C:\Users\ASUS\Documents\Semester3\Modul1>
go run c:\Users\ASUS\Documents\Semester3\Modul1\Unguided\Unguided3a.go
Nilai K = 100
Nilai f(K) = 1.0000061880
```

Setelah dimodifikasi

```
PS C:\Users\ASUS\Documents\Semester3\Modul1> go run c:\Users\ASUS\Documents\Semester3\Modul1\Unguided\no3\U
Nilai K = 10
Nilai akar 2 = 1.4062058441
PS C:\Users\ASUS\Documents\Semester3\Modul1> go run c:\Users\ASUS\Documents\Semester3\Modul1\Unguided\no3\U
Nilai K = 100
Nilai akar 2 = 1.4133387072
PS C:\Users\ASUS\Documents\Semester3\Modul1> go run c:\Users\ASUS\Documents\Semester3\Modul1\Unguided\no3\U
Nilai K = 1000
Nilai akar 2 = 1.4141252651
PS C:\Users\ASUS\Documents\Semester3\Modul1> []
```

Deskripsi Program

Program pertama menghitung nilai $f(k)$ berdasarkan rumus tertentu untuk sebuah nilai k yang diinput oleh pengguna, sementara program kedua memodifikasi logika ini untuk menghitung nilai $\sqrt{2}$ dengan menggunakan metode iterasi. Program kedua meminta pengguna untuk memasukkan nilai k , lalu melakukan perulangan dari 0 hingga k . Dalam setiap iterasi, program menghitung hasil berdasarkan rumus yang sama dengan program pertama, kemudian mengalikan hasil tersebut dengan variabel ``sqrt2`` yang diinisialisasi dengan nilai 1.0. Setelah perulangan selesai, program mencetak nilai akhir dari ``sqrt2``, yang merupakan perkiraan akar dua dengan menggunakan rumus yang dihasilkan. Sebagai contoh, jika pengguna memasukkan nilai $K = 10$, outputnya akan menampilkan nilai akar dua dengan format 10 angka di belakang koma, seperti "Nilai akar 2 = 1.4062058441".

4. PT POS membutuhkan aplikasi perhitungan biaya kirim berdasarkan berat parcel. Maka, buatlah program BlayaPos 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 Nibebankan sebesar Rp. 15,- per gram. Sisa berat (yang kurang dari 1kg) digratiskan biayanya apabila total berat ternyata lebih dari 10kg.

Sourcecode

```
package main

import (
    "fmt"
)

// Fungsi untuk menghitung biaya kirim berdasarkan berat
dalam gram
func hitungBiayaKirim(beratGr int) (int, int, int, int)
{
```

```

        beratKg := beratGr / 1000 // Menghitung berat dalam
kilogram
        sisaGr := beratGr % 1000 // Menghitung sisa berat
dalam gram setelah konversi ke kilogram

        biayaKg := beratKg * 10000 // Menghitung biaya
berdasarkan berat dalam kilogram
        biayaSisa := 0 // Inisialisasi biaya
untuk sisa berat

        // Menghitung biaya untuk sisa berat
        if sisaGr >= 500 {
            biayaSisa = sisaGr * 5 // Biaya untuk sisa berat
            jika >= 500 gram
        } else if sisaGr > 0 {
            biayaSisa = sisaGr * 15 // Biaya untuk sisa
berat jika > 0 dan < 500 gram
        }

        // Jika berat >= 10 kg, maka tidak ada biaya untuk
sisa berat
        if beratKg >= 10 {
            return beratKg, sisaGr, biayaKg, 0 //
Mengembalikan biaya tanpa biaya sisa
        }

        return beratKg, sisaGr, biayaKg, biayaSisa //
Mengembalikan semua biaya
    }

func main() {
    var beratGr int // Mendeklarasikan variabel beratGr
bertipe int

    fmt.Print("Masukkan berat parsel (gram): ") //
Menampilkan pesan untuk meminta input
    fmt.Scan(&beratGr) //
Membaca input dari pengguna

    // Memanggil fungsi hitungBiayaKirim dan menyimpan
hasilnya ke variabel
    beratKg, sisaGr, biayaKg, biayaSisa :=
hitungBiayaKirim(beratGr)
    totalBiaya := biayaKg + biayaSisa // Menghitung
total biaya

    // Menampilkan detail berat dan biaya
    fmt.Printf("Detail berat: %d kg + %d gr\n", beratKg,
sisaGr)

```

```

        fmt.Printf("Detail biaya: Rp. %d + Rp. %d\n",
        biayaKg, biayaSisa)
        fmt.Printf("Total biaya: Rp. %d\n", totalBiaya)
    }

```

Screenshoot Program

```

PROBLEMS 6 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\ASUS\Documents\Semester3\Modul1>
go run c:\Users\ASUS\Documents\Semester3\Modul1\Unguided\Unguided4.go
Masukkan berat parcel (gram): 8500
Detail berat: 8 kg + 500 gr
Detail biaya: Rp. 80000 + Rp. 2500
Total biaya: Rp. 82500
PS C:\Users\ASUS\Documents\Semester3\Modul1> go run c:\Users\ASUS\Documents\Semester3\Modul1\Unguided\Unguided4.go
Masukkan berat parcel (gram): 9250
Detail berat: 9 kg + 250 gr
Detail biaya: Rp. 90000 + Rp. 3750
Total biaya: Rp. 93750
PS C:\Users\ASUS\Documents\Semester3\Modul1> go run c:\Users\ASUS\Documents\Semester3\Modul1\Unguided\Unguided4.go
Masukkan berat parcel (gram): 11750
Detail berat: 11 kg + 750 gr
Detail biaya: Rp. 110000 + Rp. 0
Total biaya: Rp. 110000
PS C:\Users\ASUS\Documents\Semester3\Modul1>

```

Deskripsi Program

Program diatas adalah program untuk menghitung biaya pengiriman berdasarkan berat parcel yang dimasukkan oleh pengguna dalam gram. Pertama, program meminta pengguna untuk memasukkan berat, lalu mengonversi berat tersebut ke dalam kilogram dan menghitung sisa gram. Biaya pengiriman dihitung dengan tarif Rp. 10.000/kilogram, sementara untuk sisa berat, jika mencapai 500 gram, biayanya Rp. 5/gram; jika kurang dari 500 gram, biayanya Rp. 15/gram. Jika berat mencapai 10 kilogram, biaya untuk sisa berat diabaikan. Setelah semua perhitungan, program menampilkan detail berat dalam kilogram dan gram, rincian biaya per kategori, serta total biaya pengiriman. Misalnya, jika pengguna memasukkan berat 3500 gram, outputnya akan menunjukkan rincian berat dan total biaya pengiriman sebesar Rp.32.500.

5. 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!

Sourcecode

```

package main

```

```

import (
    "fmt"
)

// Fungsi untuk mencari faktor dari bilangan b
func cariFaktor(b int) []int {
    var faktor []int // Mendeklarasikan slice untuk
    menyimpan faktor-faktor

    // Melakukan perulangan dari 1 hingga b untuk mencari
    faktor
    for i := 1; i <= b; i++ {
        if b%i == 0 { // Jika b habis dibagi i
            faktor = append(faktor, i) // Menambahkan i
            ke dalam slice faktor
        }
    }
    return faktor // Mengembalikan slice faktor
}

func main() {
    var b int // Mendeklarasikan variabel b bertipe int

    fmt.Print("Masukkan Bilangan: ") // Menampilkan
    pesan untuk meminta input
    fmt.Scan(&b) // Membaca input
    dari pengguna

    if b <= 1 { // Mengecek apakah b kurang dari atau
    sama dengan 1
        fmt.Println("Bilangan harus > 1") // Menampilkan
        pesan kesalahan
        return // Menghentikan eksekusi program
    }

    faktor := cariFaktor(b) // Memanggil fungsi
    cariFaktor untuk mendapatkan faktor-faktor dari b

    // Menampilkan faktor-faktor yang ditemukan
    fmt.Print("Faktor: ")
    for _, f := range faktor { // Melakukan iterasi pada
    slice faktor
        fmt.Printf("%d ", f) // Mencetak setiap faktor
    }
    fmt.Println() // Menampilkan baris baru setelah
    mencetak faktor
}

```

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.

Sourcecode

```
package main

import (
    "fmt"
)

// Fungsi untuk mencari faktor dari bilangan b
func cariFaktor(b int) []int {
    var faktor []int // Mendeklarasikan slice untuk
    menyimpan faktor-faktor

    // Melakukan perulangan dari 1 hingga b untuk mencari
    faktor
    for i := 1; i <= b; i++ {
        if b%i == 0 { // Jika b habis dibagi i
            faktor = append(faktor, i) // Menambahkan i
            ke dalam slice faktor
        }
    }
    return faktor // Mengembalikan slice faktor
}

// Fungsi untuk mengecek apakah bilangan tersebut adalah
prima
func cekPrima(faktor []int) bool {
    return len(faktor) == 2 // Bilangan prima memiliki
    tepat dua faktor: 1 dan bilangan itu sendiri
}

func main() {
    var b int // Mendeklarasikan variabel b bertipe int

    fmt.Print("Masukkan bilangan: ") // Menampilkan
    pesan untuk meminta input
    fmt.Scan(&b) // Membaca input
    dari pengguna

    if b <= 0 { // Mengecek apakah b kurang dari atau
    sama dengan 0
        fmt.Println("Bilangan harus > 0") // Menampilkan
        pesan kesalahan
    }
```

```

        return // Menghentikan eksekusi program
    }

    faktor := cariFaktor(b) // Memanggil fungsi
    cariFaktor untuk mendapatkan faktor-faktor dari b

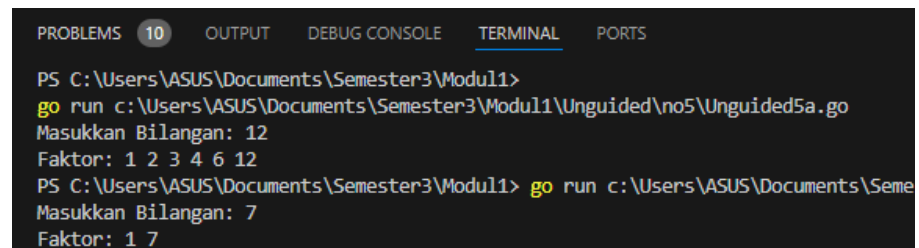
    // Menampilkan faktor-faktor yang ditemukan
    fmt.Print("Faktor: ")
    for _, f := range faktor { // Melakukan iterasi pada
    slice faktor
        fmt.Printf("%d ", f) // Mencetak setiap faktor
    }
    fmt.Println() // Menampilkan baris baru setelah
    mencetak faktor

    // Mengecek apakah bilangan adalah prima
    if cekPrima(faktor) {
        fmt.Println("Prima: true") // Menampilkan true
        jika bilangan adalah prima
    } else {
        fmt.Println("Prima: false") // Menampilkan false
        jika bilangan bukan prima
    }
}

```

Screenshoot Program

Sebelum dimodifikasi

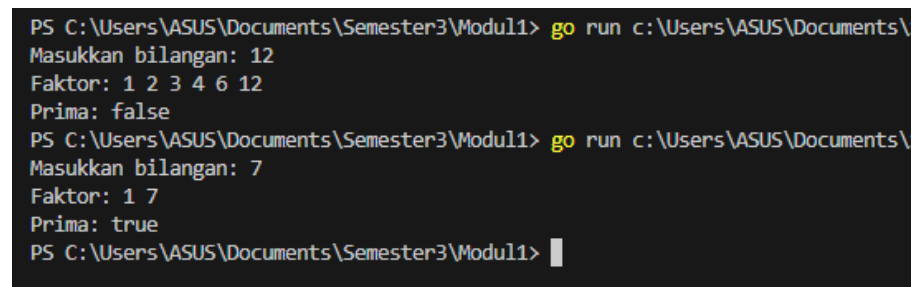


```

PROBLEMS 10 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\ASUS\Documents\Semester3\Modul1> go run c:\Users\ASUS\Documents\Semester3\Modul1\Unguided\Unguided5a.go
Masukkan Bilangan: 12
Faktor: 1 2 3 4 6 12
PS C:\Users\ASUS\Documents\Semester3\Modul1> go run c:\Users\ASUS\Documents\Semester3\Modul1\Unguided\Unguided5a.go
Masukkan Bilangan: 7
Faktor: 1 7

```

Setelah dimodifikasi



```

PS C:\Users\ASUS\Documents\Semester3\Modul1> go run c:\Users\ASUS\Documents\Semester3\Modul1\Unguided\Unguided5a.go
Masukkan bilangan: 12
Faktor: 1 2 3 4 6 12
Prima: false
PS C:\Users\ASUS\Documents\Semester3\Modul1> go run c:\Users\ASUS\Documents\Semester3\Modul1\Unguided\Unguided5a.go
Masukkan bilangan: 7
Faktor: 1 7
Prima: true
PS C:\Users\ASUS\Documents\Semester3\Modul1>

```

Deskripsi Program

Program diatas adalah program yang berfungsi untuk mencari faktor dari sebuah bilangan bulat positif yang dimasukkan pengguna dan memeriksa apakah bilangan tersebut adalah bilangan prima. Program

meminta pengguna untuk memasukkan bilangan, lalu menggunakan fungsi untuk mencari faktor dengan memeriksa setiap bilangan dari 1 hingga bilangan tersebut. Setelah itu, program mencetak semua faktor yang ditemukan dan menentukan bilangan prima berdasarkan jumlah faktornya; jika hanya memiliki dua faktor (1 dan bilangan itu sendiri), program akan mencetak bahwa bilangan tersebut adalah prima (true), sebaliknya akan menyatakan tidak prima (false).

IV. KESIMPULAN

Dalam bahasa pemrograman Go, setiap file harus memiliki sebuah **package**, dengan minimal satu file ber-package *main*, yang mengandung fungsi *main()* sebagai titik eksekusi program. **Import** digunakan untuk memuat package lain, seperti *fmt* yang sering dipakai untuk fungsi I/O teks seperti *fmt.Println()* dan *fmt.Printf()*. Go mendukung berbagai **tipe data**, seperti bilangan cacah (*uint*), bilangan bulat (*int*), desimal (*float32*, *float64*), boolean (*bool*), dan string. Untuk membuat variabel baru, digunakan keyword **var**, sedangkan **konstanta** bisa digunakan untuk menyimpan nilai tetap seperti π . **Perulangan** di Go menggunakan kata kunci *for*, baik dalam bentuk *while-loop* maupun *repeat-until*. Untuk **percabangan**, Go menawarkan bentuk *if-else* dan *switch-case*, di mana *switch* bisa digunakan dengan atau tanpa ekspresi, memperluas fleksibilitas penulisan logika kondisi.

V. REFERENSI

- [1] Novalagung. "Golang Tipe Data." Dasar Pemrograman Golang. Accessed October 6, 2024. <https://dasarpemrogramangolang.novalagung.com/A-tipe-data.html>