

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL 2
REVIEW STRUKTUR KONTROL**



Disusun Oleh :

Shiva Indah Kurnia

2311102035

S1-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

- Keyword ``package``: Setiap file program Go harus memiliki package. Minimal harus ada satu file dengan package main, yang akan dieksekusi pertama kali ketika program dijalankan.
- Keyword ``import``: Digunakan untuk mengimpor package lain ke dalam program agar fungsionalitas dari package tersebut bisa digunakan. Contoh package bawaan Go adalah ``fmt``, yang menyediakan banyak fungsi untuk operasi I/O terkait teks.
- Keyword ``var``: Digunakan untuk mendeklarasikan variabel baru.
- Fungsi ``main()``: Fungsi ini harus ada di dalam file dengan package main dan akan menjadi titik awal eksekusi program.
- Fungsi ``fmt.Println()``: Digunakan untuk menampilkan teks ke layar. Berada di dalam package ``fmt``, jadi package ini harus diimpor terlebih dahulu.
- Fungsi ``fmt.Printf()``: Mirip dengan ``fmt.Println()``, tetapi memberikan fleksibilitas untuk menentukan format output.

B. Tipe Data dan Instruksi Dasar

- Tipe Data Numerik Non-Desimal:
 - ``uint``: untuk bilangan cacah (positif).
 - ``int``: untuk bilangan bulat (positif dan negatif).
- Tipe Data Numerik Desimal:
 - ``float32`` dan ``float64``: Berbeda dalam lebar cakupan nilai desimal.
- Tipe Data Boolean (``bool``): Hanya memiliki dua nilai, yaitu ``true`` dan ``false``. Sering digunakan dalam seleksi kondisi dan perulangan.
- Tipe Data String: Nilainya diapit oleh tanda kutip dua (``"``).
- Konstanta Simbolik: Digunakan untuk memberikan nama pada nilai konstan, seperti PI untuk nilai π .

C. Struktur Kontrol Perulangan

Go hanya memiliki kata kunci `for` untuk perulangan. Ada dua bentuk yang sering digunakan:

1. While-Loop: Memastikan kondisi bernilai benar saat masuk loop, dan kondisi akan bernilai salah saat keluar loop.
2. Repeat-Until: Perulangan berlanjut hingga kondisi terpenuhi (benar).

D. Struktur Kontrol Percabangan

1. If-Else: Go mendukung beberapa bentuk if-else, baik yang bersarang maupun sederhana.
2. Switch-Case: Ada dua variasi switch-case di Go. Pertama, ekspresi ditulis di perintah `switch`, dan nilai di label `case`. Kedua, switch tanpa ekspresi di mana setiap `case` bisa berupa ekspresi boolean, yang menyederhanakan bentuk `if-else`.

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 Go dan bertujuan untuk memahami dasar penggunaan input/output serta variabel.

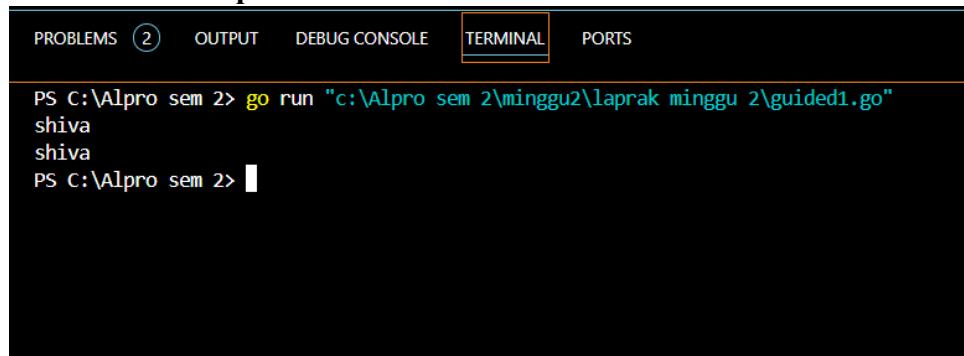
Sourcecode

```
package main

import (
    "fmt"
)

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

Screenshoot Output



```
PS C:\Alpro sem 2> go run "c:\Alpro sem 2\minggu2\laprak minggu 2\guided1.go"
shiva
shiva
PS C:\Alpro sem 2> 
```

Deskripsi Program

Program di atas adalah program sederhana dalam bahasa Go yang menerima input dari pengguna dan menampilkan kembali input tersebut ke layar. Berikut adalah deskripsi singkat dari skrip tersebut:

1. **Package main:** Program berada dalam package `*main*`, yang menandakan bahwa ini adalah file utama yang akan dieksekusi.

2. **Import fmt:** Package `fmt` diimpor untuk memanfaatkan fungsi I/O, seperti membaca input dan menampilkan output.

3. **Variabel nama:** Sebuah variabel bernama `nama` dideklarasikan dengan tipe data `string` untuk menyimpan input dari pengguna.

4. **fmt.Scanln():** Fungsi ini digunakan untuk membaca input dari pengguna dan menyimpannya ke dalam variabel `nama`.

5. **fmt.Println():** Fungsi ini menampilkan nilai dari variabel `nama` ke layar, yang berisi input yang telah diberikan oleh pengguna.

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 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. 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, " ")

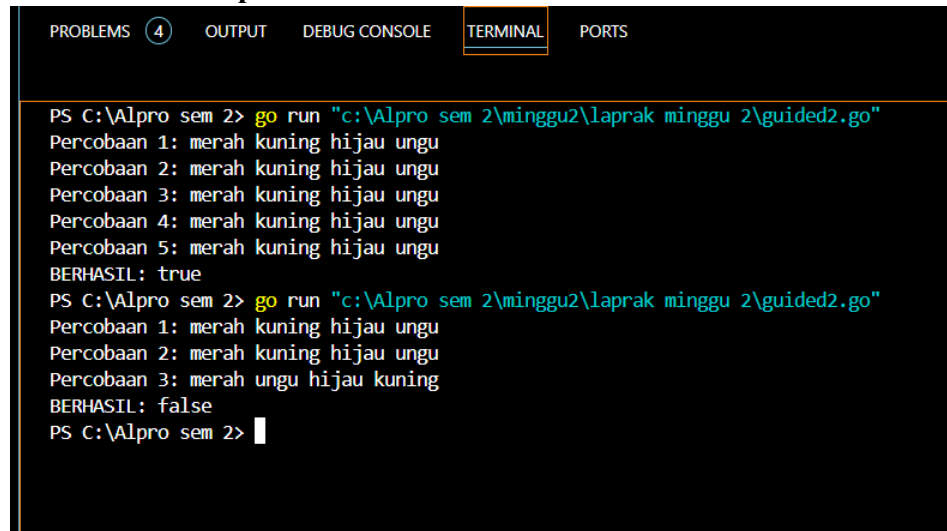
        // Mengecek apakah urutan warna sesuai
        for j := 0; j < 4; j++ {
            if colors[j] != correctOrder[j] {
                success = false
                break
            }
        }

        // 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")
    }
}

```

Screenshoot Output



```
PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Alpro sem 2> go run "c:\Alpro sem 2\minggu2\laprak minggu 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
PS C:\Alpro sem 2> go run "c:\Alpro sem 2\minggu2\laprak minggu 2\guided2.go"
Percobaan 1: merah kuning hijau ungu
Percobaan 2: merah kuning hijau ungu
Percobaan 3: merah ungu hijau kuning
BERHASIL: false
PS C:\Alpro sem 2> |
```

Deskripsi Program

Program di atas adalah program dalam bahasa Go yang memeriksa apakah urutan warna yang diinputkan oleh pengguna sesuai dengan urutan warna yang telah ditentukan. Berikut deskripsi singkatnya:

- 1. Urutan warna yang benar:** Program mendefinisikan urutan warna yang benar dalam slice `correctOrder`, yaitu "merah", "kuning", "hijau", "ungu".
- 2. Input pengguna:** Pengguna diminta untuk memasukkan urutan warna selama 5 kali percobaan. Program menggunakan `bufio.NewReader` untuk membaca input dari pengguna baris demi baris.
- 3. Proses validasi: Pada setiap percobaan:**
 - Program membaca input dari pengguna dan membersihkannya dari karakter newline (`\n`) menggunakan `strings.TrimSpace`.
 - Input tersebut dipisahkan menjadi elemen-elemen warna berdasarkan spasi menggunakan `strings.Split`.
 - Program memeriksa apakah urutan warna yang diinputkan sesuai dengan urutan yang benar (`correctOrder`).
 - Jika pada salah satu percobaan urutan tidak sesuai, program keluar dari loop dan menyatakan percobaan gagal.
- 4. Hasil:** Setelah 5 percobaan atau setelah menemukan ketidaksesuaian, program menampilkan hasil. Jika semua input benar, program mencetak "BERHASIL: true", jika tidak, "BERHASIL: false".

3. Guided – 3 – Soal abcd

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

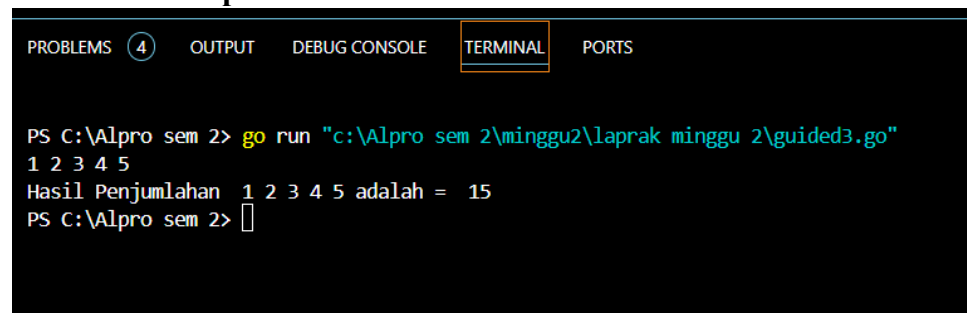
import "fmt"

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)
}
```

Screenshot Output

A screenshot of a terminal window showing the execution of a Go program. The terminal has tabs for PROBLEMS (4), OUTPUT, DEBUG CONSOLE, TERMINAL (selected), and PORTS. The command executed is 'go run "c:\Alpro sem 2\minggu2\laprak minggu 2\guided3.go"'. The output shows the user input '1 2 3 4 5' and the program response 'Hasil Penjumlahan 1 2 3 4 5 adalah = 15'. The prompt 'PS C:\Alpro sem 2>' is visible at the bottom.

```
PROBLEMS (4) OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Alpro sem 2> go run "c:\Alpro sem 2\minggu2\laprak minggu 2\guided3.go"
1 2 3 4 5
Hasil Penjumlahan 1 2 3 4 5 adalah = 15
PS C:\Alpro sem 2> 
```

Deskripsi Program

Program di atas adalah program sederhana dalam bahasa Go yang melakukan penjumlahan lima angka yang dimasukkan oleh pengguna. Berikut adalah deskripsi singkat dari program tersebut:

1. **Deklarasi variabel:** Program mendeklarasikan lima variabel `a`, `b`, `c`, `d`, dan `e` bertipe `int` untuk menyimpan input bilangan dari pengguna, serta variabel `hasil` untuk menyimpan hasil penjumlahan.

2. **Membaca input:** Program menggunakan ``fmt.Scanln()`` untuk membaca lima bilangan yang dimasukkan oleh pengguna dan menyimpannya ke dalam variabel ``a``, ``b``, ``c``, ``d``, dan ``e``.

3. **Proses penjumlahan:** Program menjumlahkan nilai-nilai dari kelima variabel tersebut dan menyimpannya ke dalam variabel ``hasil``.

4. **Menampilkan hasil:** Program mencetak hasil penjumlahan beserta nilai-nilai yang dimasukkan oleh pengguna dengan menggunakan ``fmt.Println()``.

4. Guided 4

Soal Studi Case

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

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)
}

```

Screenshot Output

```

PROBLEMS (6) OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Alpro sem 2> go run "c:\Alpro sem 2\minggu2\laprak minggu 2\guided4.go"
Masukkan nilai: 80.1
Nilai Indeks untuk nilai 80.10 adalah A
PS C:\Alpro sem 2> go run "c:\Alpro sem 2\minggu2\laprak minggu 2\guided4.go"
Masukkan nilai: 93.5
Nilai Indeks untuk nilai 93.50 adalah A
PS C:\Alpro sem 2> go run "c:\Alpro sem 2\minggu2\laprak minggu 2\guided4.go"
Masukkan nilai: 70.6
Nilai Indeks untuk nilai 70.60 adalah B
PS C:\Alpro sem 2> go run "c:\Alpro sem 2\minggu2\laprak minggu 2\guided4.go"
Masukkan nilai: 49.5
Nilai Indeks untuk nilai 49.50 adalah D
PS C:\Alpro sem 2>

```

Deskripsi Program

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

= Program menampilkan hasil eksekusi sesuai dengan spesifikasi yang diminta, di mana nilai 80.1 lebih besar dari 80 sehingga memperoleh nilai A

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

= 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'.

=

```
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)
}

```

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 **arrange** 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".

D. UNGUIDED

1. Unguided1 – 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>	N : <u>0</u>
Bunga 1: <u>Kertas</u>	Pita :
Bunga 2: <u>Mawar</u>	
Bunga 3: <u>Tulip</u>	
Pita: Kertas - Mawar - Tulip -	

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):

Bunga 1: <u>Kertas</u>	Bunga 1: <u>SELESAI</u>
Bunga 2: <u>Mawar</u>	Pita :
Bunga 3: <u>Tulip</u>	Bunga: 0
Bunga 4: <u>SELESAI</u>	
Pita: Kertas - Mawar - Tulip -	
Bunga: 3	

Sourcecode

```
package main

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

func main() {
```

```

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

fmt.Println("Masukkan nama 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))
}

```

Screenshoot Output

```

PS C:\Alpro sem 2> go run "c:\Alpro sem 2\minggu2\laprak minggu 2\unguided1.go"
Masukkan nama bunga, ketik 'SELESAI' untuk selesai:
Bunga: spider lily
Bunga: anggrek
Bunga: mawar
Bunga: SELESAI
Pita: spider lily - anggrek - mawar
Bunga: 3
PS C:\Alpro sem 2>

```

Deskripsi Program

Program di atas adalah program dalam bahasa Go yang meminta pengguna untuk memasukkan nama-nama bunga hingga pengguna mengetik "SELESAI". Berikut deskripsi singkat dari program tersebut:

1. Deklarasi variabel:
Program mendeklarasikan cut 'bunga' untuk menyimpan nama-nama bunga yang dimasukkan oleh pengguna, serta 'scanner' untuk membaca input dari terminal menggunakan 'bufio.NewScanner'.

2. Input bunga:
Program meminta pengguna memasukkan nama bunga secara berulang. Jika pengguna mengetik "SELESAI", program akan berhenti menerima input.
3. Menambahkan ke cut:
Setiap input yang diberikan oleh pengguna akan ditambahkan ke cut 'bunga'.
4. Menampilkan hasil:
 - Program menampilkan daftar bunga yang dimasukkan, dipisahkan dengan tanda strip (" - "), menggunakan 'strings.Join()'.
 - Program juga menampilkan jumlah add up to bunga yang dimasukkan dengan 'len(bunga)'.

2. Unguided 2 – 2B – 3

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

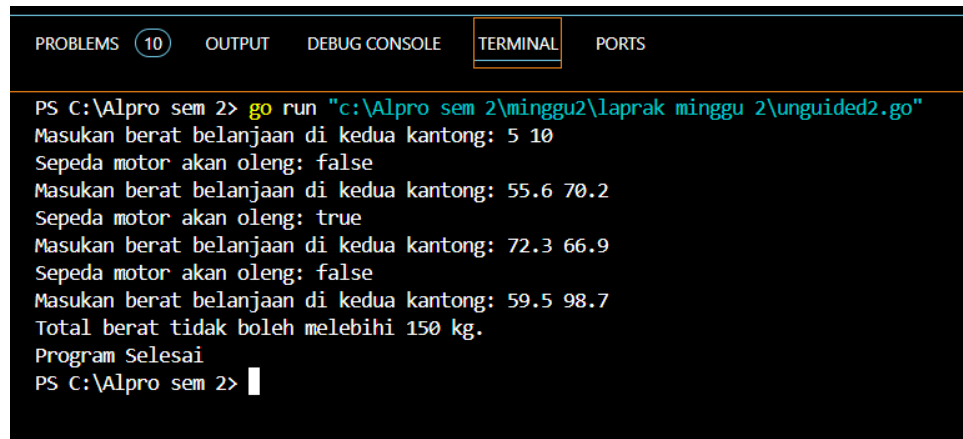
import (
    "fmt"
    "math"
)

func main() {
    var kantong1, kantong2 float64
    for {
        fmt.Println("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")
        }
    }
}
```


Screenshot Output

A screenshot of a terminal window showing the execution of a Go program. The terminal has tabs for PROBLEMS (10), OUTPUT, DEBUG CONSOLE, TERMINAL (selected), and PORTS. The output text is as follows:

```
PS C:\Alpro sem 2> go run "c:\Alpro sem 2\minggu2\laprak minggu 2\unguided2.go"
Masukan berat belanja di kedua kantong: 5 10
Sepeda motor akan oleng: false
Masukan berat belanja di kedua kantong: 55.6 70.2
Sepeda motor akan oleng: true
Masukan berat belanja di kedua kantong: 72.3 66.9
Sepeda motor akan oleng: false
Masukan berat belanja di kedua kantong: 59.5 98.7
Total berat tidak boleh melebihi 150 kg.
Program Selesai
PS C:\Alpro sem 2> |
```

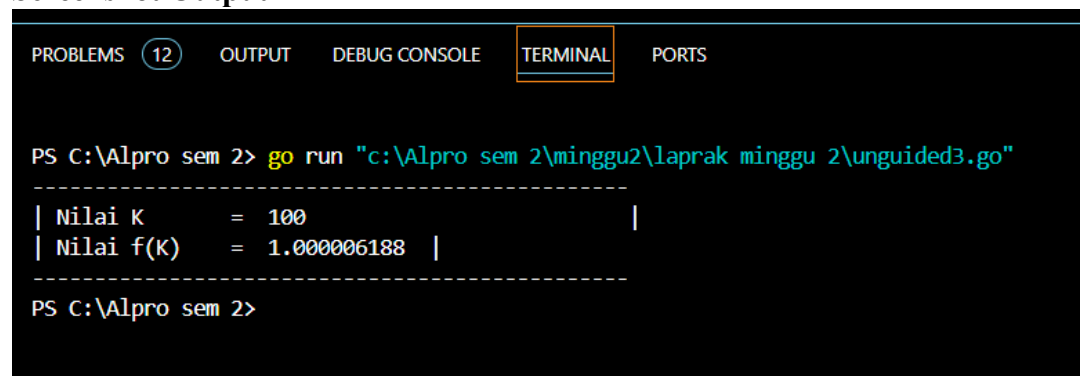
Deskripsi Program

Program di atas adalah program dalam bahasa Go yang mengelola berat belanja di dua kantong dan menentukan apakah sepeda engine akan oleng berdasarkan berat tersebut. Berikut adalah deskripsi singkat dari program tersebut:

1. Deklarasi variabel: Program mendeklarasikan dua variabel 'kantong1' dan 'kantong2' bertipe 'float64' untuk menyimpan berat belanja dari masing-masing kantong.
2. Input pengguna:
Program memasuki circle yang meminta pengguna untuk memasukkan berat belanja di kedua kantong.
3. Perhitungan add up to berat:
Setelah menerima input, program menghitung add up to berat dari kedua kantong.
4. Pemeriksaan kondisi:
 - Jika add up to berat melebihi 150 kg, program memberi peringatan dan mengakhiri program dengan mencetak "Program Selesai".
 - Jika salah satu berat kantong negatif, program memberi peringatan bahwa berat tidak boleh negatif.
 - Jika selisih berat antara kedua kantong kurang dari 9 kg, program menyatakan bahwa "Sepeda engine akan oleng: untrue".
 - Jika tidak memenuhi semua kondisi di atas, program menyatakan bahwa "Sepeda engine akan oleng: genuine".

Cara Kerja Program

1. Program mulai dengan mengimpor paket yang diperlukan (fmt dan math).



Deskripsi Program

Program di atas adalah program sederhana dalam bahasa Go yang menampilkan nilai dari dua variabel: satu numbers dan satu coast. Berikut adalah deskripsi singkat dari program tersebut:

1. Deklarasi variabel:
 - Variabel 'k' bertipe 'int' diinisialisasi dengan nilai 100.
 - Variabel 'f' bertipe 'float64' diinisialisasi dengan nilai 1.0000061880.
2. Output:

Program mencetak tabel yang menampilkan nilai dari kedua variabel tersebut.

 - Menggunakan 'fmt.Println()', program mencetak garis pemisah dan dua baris informasi:
 - Baris pertama menunjukkan nilai 'k'.
 - Baris kedua menunjukkan nilai 'f'.
3. Arrange:

Tabel output memiliki garis level di atas dan di bawah, dengan name dan nilai yang disusun rapi.

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 parcel
    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 parcel
    for i, parcel := range parcels {
        totalBiaya :=
hitungBiayaParcel(parcel["berat"].(int))
        parcels[i]["totalBiaya"] = totalBiaya
        fmt.Printf("Contoh #d\n", i+1)
        fmt.Printf("Berat parcel (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
}

```

Screenshot Output

```

PROBLEMS (14) OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Alpro sem 2> go run "c:\Alpro sem 2\minggu2\laprak minggu 2\unguided4.go"
Contoh #1
Berat parsel (gram): 8500
Detail berat: 8 kg + 500 gr
Total biaya: Rp. 82500
-----
Contoh #2
Berat parsel (gram): 9250
Detail berat: 9 kg + 250 gr
Total biaya: Rp. 91250
-----
Contoh #3
Berat parsel (gram): 11750
Detail berat: 11 kg + 750 gr
Total biaya: Rp. 113750
-----
PS C:\Alpro sem 2>

```

Deskripsi Program

Program di atas adalah program dalam bahasa Go yang menghitung dan menampilkan biaya pengiriman untuk beberapa parsel berdasarkan beratnya. Berikut adalah deskripsi singkat dari program tersebut:

1. Information Parsel:

Program mendeklarasikan cut 'parcels', yang berisi peta (outline) untuk setiap parsel, termasuk berat dalam gram dan rincian berat dalam kilogram dan gram.

2. Menghitung Biaya:

Program menggunakan circle untuk iterasi melalui setiap parcel. Untuk setiap parcel, fungsi 'hitungBiayaParcel' dipanggil dengan berat parcel untuk menghitung add up to biaya berdasarkan beratnya.

3. Output Biaya:

- Program mencetak informasi tentang setiap parcel, termasuk nomor contoh, berat parcel dalam gram, detail berat dalam kilogram dan gram, serta add up to biaya pengiriman.
- Organize output disusun rapi dengan pemisah untuk memisahkan setiap parcel.

5. Fungsi hitungBiayaParcel:

- Fungsi ini menerima berat dalam gram dan menghitung biaya pengiriman.
- Biaya dihitung dengan tarif per kilogram (Rp. 10.000) dan biaya tambahan per gram (Rp. 5).
- Fungsi mengembalikan add up to biaya sebagai nilai numbers.

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: <u>12</u> Faktor: 1 2 3 4 6 12	Bilangan: <u>7</u> 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: <u>12</u> Faktor: 1 2 3 4 6 12 Prima: false	Bilangan: <u>7</u> Faktor: 1 7 Prima: true
---	--

Sourcecode

```
package main

import (
    "fmt"
)

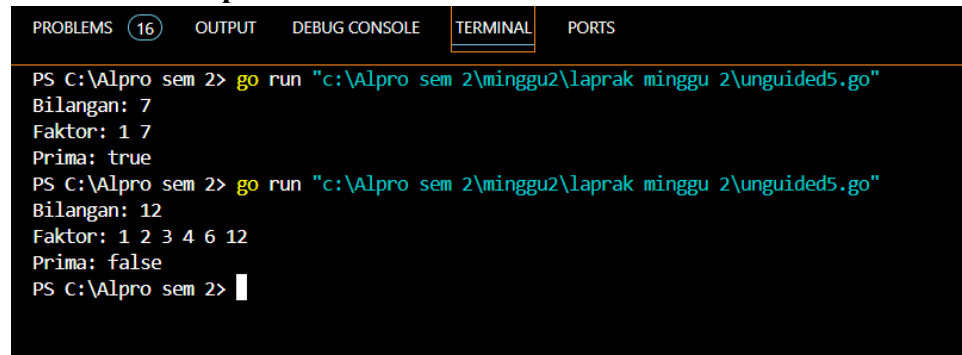
func main() {
    var b int
    fmt.Print("Bilangan: ")
    fmt.Scanln(&b)

    fmt.Print("Faktor: ")
    for i := 1; i <= b; i++ {
        if b%i == 0 {
            fmt.Print(i, " ")
        }
    }
    fmt.Println()

    prima := true
    if b <= 1 {
        prima = false
    } else {
        for i := 2; i*i <= b; i++ {
            if b%i == 0 {
                prima = false
                break
            }
        }
    }

    fmt.Println("Prima:", prima)
}
```

Screenshot Output



```
PS C:\Alpro sem 2> go run "c:\Alpro sem 2\minggu2\laprak minggu 2\unguided5.go"
Bilangan: 7
Faktor: 1 7
Prima: true
PS C:\Alpro sem 2> go run "c:\Alpro sem 2\minggu2\laprak minggu 2\unguided5.go"
Bilangan: 12
Faktor: 1 2 3 4 6 12
Prima: false
PS C:\Alpro sem 2> 
```

Deskripsi Program

Program di atas adalah program dalam bahasa Go yang meminta pengguna untuk memasukkan sebuah bilangan bulat, kemudian menentukan dan menampilkan faktor dari bilangan tersebut serta memeriksa apakah bilangan itu adalah bilangan prima. Berikut adalah deskripsi singkat dari program tersebut:

1. **Input Pengguna:** Program meminta pengguna untuk memasukkan sebuah bilangan bulat melalui input. Nilai ini disimpan dalam variabel `b`.
2. **Menampilkan Faktor:**
 - Program melakukan iterasi dari 1 hingga bilangan `b`.
 - Dalam setiap iterasi, program memeriksa apakah `b` habis dibagi oleh `i`. Jika ya, `i` adalah faktor dari `b` dan akan dicetak ke layar.
3. **Memeriksa Bilangan Prima:**
 - Program menginisialisasi variabel `prima` sebagai `true`.
 - Jika `b` kurang dari atau sama dengan 1, program mengubah `prima` menjadi `false`, karena bilangan negatif dan 0 atau 1 bukan bilangan prima.
 - Untuk bilangan lebih dari 1, program memeriksa apakah `b` dapat dibagi oleh angka dari 2 hingga akar kuadrat dari `b`. Jika ada pembagi, `prima` diubah menjadi `false`.
4. **Output Hasil:** Program menampilkan hasil apakah bilangan `b` adalah bilangan prima atau bukan dengan mencetak nilai dari variabel `prima`.

E. KESIMPULAN

Dalam bahasa pemrograman Go, setiap record harus memiliki sebuah bundle, dengan setidaknya satu record yang menggunakan bundle primary, yang berisi fungsi fundamental() sebagai titik awal eksekusi program. Penggunaan purport diperlukan untuk memuat bundle lain, seperti fmt, yang sering digunakan untuk fungsi input/output teks, termasuk fmt.Println() dan fmt.Printf(). Go mendukung

berbagai jenis tipe information, seperti bilangan cacah (uint), bilangan bulat (int), bilangan desimal (float32, float64), boolean (bool), dan string. Untuk mendeklarasikan variabel baru, digunakan kata kunci var, sedangkan konstanta dapat digunakan untuk menyimpan nilai tetap, seperti π . Pengulangan di Go menggunakan kata kunci for, yang dapat diterapkan dalam bentuk while-loop maupun repeat-until. Untuk struktur percabangan, Go menyediakan bentuk if-else dan switch-case, di mana switch dapat digunakan dengan atau tanpa ekspresi, memberikan fleksibilitas dalam penulisan logika kondisi.

F. REFERENSI

[1] Modul 2 Praktikum Algoritma 2

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