LAPORAN PRAKTIKUM
ALGORITMA PEMROGRAMAN 2
MODUL II
REVIEW STRUKTUR KONTROL

Disusun Oleh:
Haifa Zahra Azzimmi
2311102163

S1 IF 11 05

PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2024

MODUL II

I. Dasar Teori

1. Struktur Kontrol Perulansgan

   Go hanya mempunyai kata kunci for untuk semua jenis perulangan yang kita pelajari dalam notasi algoritma. Dua bentuk yang kita gunakan di sini adalah struktur while-loop dan repeat-until. Bentuk perulangan di go dapat dilihat gambar dibawah ini

```
for inisialisasi; kondisi; update {
    // .. for-loop ala C
    // .. ke-3 bagian opsional, tetapi ";" tetap harus ada
}
```

```
for kondisi {
    // .. ulangi kode di sini selama kondisi terpenuhi
    // .. sama seperti "for ; kondisi; {"
}
```

```
for {
    // .. tanpa kondisi, berarti loop tanpa henti (perlu if-break)
}
```

```
for ndx, var := range slice/array {
    // .. iterator mengunjungi seluruh isi slice/array
    // .. pada setiap iterasi ndx diset indeks dan var diisi nilainya
}
```

   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

2. Struktur Kontrol Percabangan

   Untuk analisa kasus, bahasa Go mendukung dua bentuk percabangan, yaitu if-else dan switch-case. 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. 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-else if-..-else-endif.

II. Guided

- Guided 1

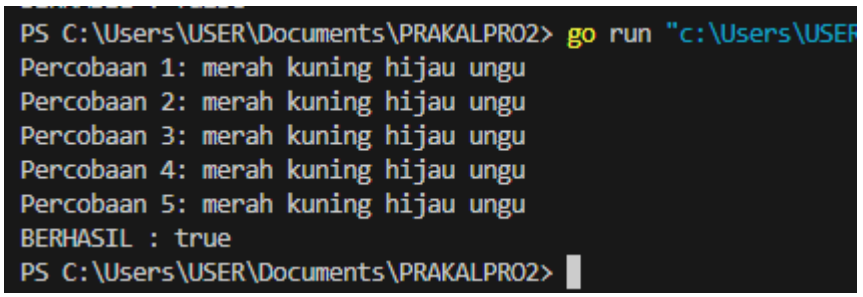
Source Code:

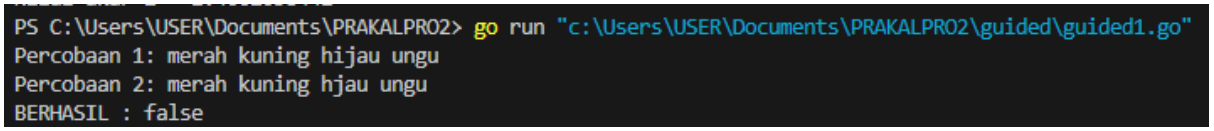
```
package main

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

func main() {
	correctOrder := []string{"merah", "kuning", "hijau", "ungu"}

	reader := bufio.NewReader(os.Stdin)
	success := true

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

		input, _ := reader.ReadString('\n')
		input = strings.TrimSpace(input)

		colors := strings.Split(input, " ")

		for j := 0; j < 4; j++ {
			if colors[j] != correctOrder[j] {
				success = false
				break
			}
		}

		if !success {
			break
		}
	}

	if success {
		fmt.Println("BERHASIL : true")
	} else {
		fmt.Println("BERHASIL : false")
	}
}
```

Penjelasan:

Kode di atas merupakan program Go yang meminta pengguna untuk memasukkan urutan warna sebanyak lima kali, dan kemudian membandingkan urutan warna yang dimasukkan dengan urutan warna yang benar, yaitu "merah", "kuning", "hijau", dan "ungu". Pengguna diminta untuk memasukkan urutan warna dalam satu baris yang dipisahkan oleh spasi, lalu input tersebut akan dipisahkan menjadi array string. Program akan membandingkan setiap warna dari input pengguna dengan urutan warna yang benar. Jika urutan warna pada percobaan tidak sesuai dengan urutan yang benar, maka program akan berhenti dan mencetak "BERHASIL : false". Namun, jika semua percobaan sesuai dengan urutan yang benar, program akan mencetak "BERHASIL : true".

Output:





- Guided 2

Source Code:

```
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)
}
```

Penjelasan:

Kode di atas merupakan program sederhana dalam bahasa Go yang meminta pengguna untuk memasukkan lima angka dan kemudian menjumlahkannya. program ini menghitung penjumlahan dari lima angka yang dimasukkan oleh pengguna

Output:

```
1 2 3 4 5
Hasil Penjumlahan  1 2 3 4 5 adalah =  15
```

- Guided 3

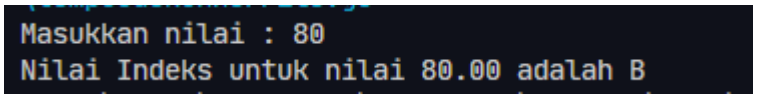
Source Code:

```
package main

import "fmt"

func main() {
	var nam float32
	var nmk string

	fmt.Print("Masukkan 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)
}
```

Penjelasan:

Kode di atas adalah program Go yang menerima input nilai berupa angka desimal (float32) dan mengonversinya menjadi nilai indeks huruf (A hingga F) berdasarkan rentang nilai tertentu.

Output:



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

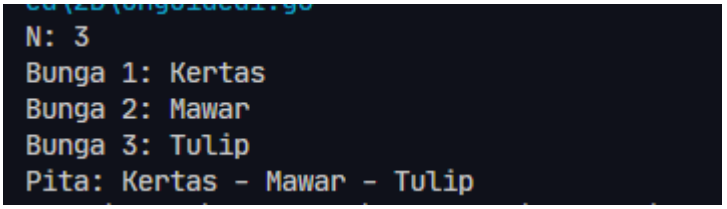
Source Code:

```
package main

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

func main() {
	var jumlahBunga int
	pita := ""
	scanner := bufio.NewScanner(os.Stdin)

	fmt.Print("N: ")
	fmt.Scanln(&jumlahBunga)

	for i := 1; i <= jumlahBunga; i++ {
		fmt.Printf("Bunga %d: ", i)
		scanner.Scan()
		bunga := scanner.Text()

		if pita == "" {
			pita = bunga
		} else {
			pita += " - " + bunga
		}
	}

	fmt.Printf("Pita: %s\n", pita)
}
```

Penjelasan:

Kode di atas bekerja dengan meminta pengguna untuk memasukkan jumlah bunga yang akan dimasukkan, kemudian memulai perulangan sebanyak jumlah tersebut untuk meminta nama bunga satu per satu. Nama-nama bunga tersebut disimpan dalam variabel `pita` yang diinisialisasi sebagai string kosong. Nama bunga pertama langsung disimpan di variabel `pita`, sedangkan nama-nama bunga berikutnya ditambahkan ke dalam variabel tersebut dengan tanda pemisah " - ". Setelah semua nama bunga diinput, program akan mencetak hasilnya dalam bentuk "Pita" yang berisi nama-nama bunga yang digabungkan dengan tanda pemisah tersebut.

Output:



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. 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.
```

Source Code:

```
package main

import (
        "fmt"
```

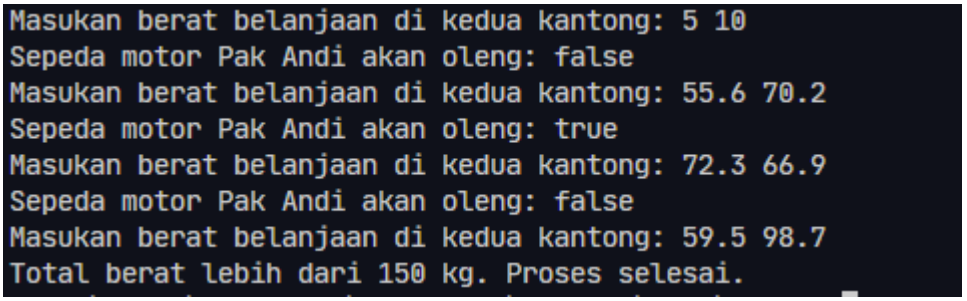
```
)

func main() {
	var beratKantong1, beratKantong2 float64
	var totalBerat float64
	selisih := 0.0
	const batasTotalBerat = 150.0

	for {
		fmt.Print("Masukan berat belanjaan di kedua kantong: ")
		fmt.Scan(&beratKantong1, &beratKantong2)

		totalBerat = beratKantong1 + beratKantong2

		if beratKantong1 < 0 || beratKantong2 < 0 {
			fmt.Println("Proses selesai.")
			break
		}

		if totalBerat > batasTotalBerat {
			fmt.Println("Total berat lebih dari 150 kg. Proses selesai.")
			break
		}

		if beratKantong1 > beratKantong2 {
			selisih = beratKantong1 - beratKantong2
		} else {
			selisih = beratKantong2 - beratKantong1
		}

		if selisih >= 9 {
			fmt.Println("Sepeda motor Pak Andi akan oleng: true")
		} else {
			fmt.Println("Sepeda motor Pak Andi akan oleng: false")
		}
	}
}
```

Penjelasan:

Kode di atas adalah program Go yang meminta pengguna untuk memasukkan berat belanjaan dalam dua kantong secara berulang hingga pengguna memasukkan berat negatif atau total berat melebihi 150 kg. Di dalam loop, program pertama-tama menjumlahkan berat kedua kantong. Jika salah satu berat negatif, program mencetak pesan "Proses selesai" dan berhenti. Jika total berat melebihi 150 kg, program juga

mencetak pesan dan berhenti. Selanjutnya, program menghitung selisih berat antara kedua kantong dan menentukan apakah selisih tersebut lebih besar atau sama dengan 9 kg. Jika ya, program mencetak "Sepeda motor Pak Andi akan oleng: true", sebaliknya mencetak "Sepeda motor Pak Andi akan oleng: false". Dengan demikian, program ini membantu memantau berat belanjaan dan potensi ketidakstabilan pada sepeda motor berdasarkan berat kantong.

Output:



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):

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

Source Code:

```
package main

import (
	"fmt"
	"math"
)

func hitungFK(K float64) float64 {
	atas := (4*K + 2) * (4*K + 2)
	bawah := (4*K + 1) * (4*K + 3)
	return atas / bawah
}

func hitungAkar2() float64 {
	return math.Sqrt(2)
```

```
}

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

	nilaiFK := hitungFK(K)
	fmt.Printf("Nilai f(K) = %.10f\n", nilaiFK)

	fmt.Println("\nMenghitung akar 2 dengan presisi hingga 10 digit desimal:")
	akar2 := hitungAkar2()
	fmt.Printf("Nilai akar 2 = %.10f\n", akar2)

	fmt.Println("\nHasil iterasi beberapa nilai K:")
	for i := 1; i <= 3; i++ {
		nilaiFK = hitungFK(float64(i))
		fmt.Printf("Nilai K = %d, Nilai f(K) = %.10f\n", i, nilaiFK)
	}
}
```

Penjelasan:

Kode di atas adalah program Go yang melakukan dua perhitungan matematis: menghitung nilai fungsi \(f(K) \) berdasarkan nilai yang dimasukkan oleh pengguna dan menghitung akar kuadrat dari 2 dengan presisi hingga 10 digit desimal. Pertama, program meminta pengguna untuk memasukkan nilai \(K \), kemudian menggunakan fungsi `hitungFK(K)` untuk menghitung nilai \(f(K) \) berdasarkan rumus yang diberikan. Hasilnya dicetak dengan format 10 digit desimal. Selanjutnya, program menghitung dan mencetak nilai akar kuadrat dari 2 menggunakan fungsi `hitungAkar2()` dan juga mencetak nilai \(f(K) \) untuk \(K = 1, 2, 3 \) dalam sebuah iterasi. Dengan demikian, program ini menyediakan hasil perhitungan yang jelas dan terstruktur..

Output:

```
Masukkan nilai K: 100
Nilai f(K) = 1.0000061880

Menghitung akar 2 dengan presisi hingga 10 digit desimal:
Nilai akar 2 = 1.4142135624

Hasil iterasi beberapa nilai K:
Nilai K = 1, Nilai f(K) = 1.0285714286
Nilai K = 2, Nilai f(K) = 1.0101010101
Nilai K = 3, Nilai f(K) = 1.0051282051
```

4. PT POS membutuhkan aplikasi perhitungan biaya kirim berdasarkan berat parsel. Maka, buatlah program BiayaPos untuk menghitung biaya pengiriman tersebut dengan ketentuan sebagai berikut! Dari berat parsel (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):

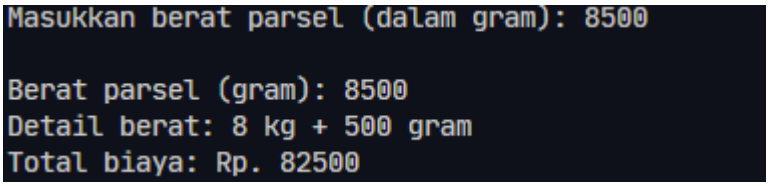
   Source Code:

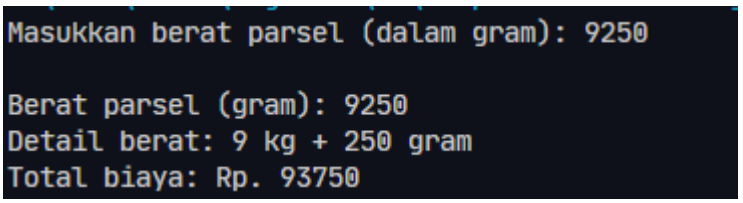
```
package main

import (
	"fmt"
)

func main() {
	var beratParsel, totalBiaya float64
	var kg, gram int

	fmt.Print("Masukkan berat parsel (dalam gram): ")
	fmt.Scan(&beratParsel)

	kg = int(beratParsel) / 1000
	gram = int(beratParsel) % 1000

	totalBiaya = float64(kg) * 10000

	if gram >= 500 {
		totalBiaya += 2500
	} else if gram > 0 {
		totalBiaya += float64(gram) * 15
	}

	if kg >= 10 {
		totalBiaya = 100000
	}

	fmt.Printf("\nBerat parsel (gram): %.0f\n", beratParsel)
	fmt.Printf("Detail berat: %d kg + %d gram\n", kg, gram)
	fmt.Printf("Total biaya: Rp. %.0f\n", totalBiaya)
}
```

Penjelasan:

Kode di atas adalah program Go yang menghitung biaya pengiriman berdasarkan berat parsel yang dimasukkan pengguna dalam satuan gram. Pertama, berat parsel diubah menjadi kilogram dan gram menggunakan operasi pembagian dan modulus. Biaya dasar ditetapkan sebesar Rp 10.000 per kilogram, dan jika ada sisa gram, tambahan biaya dikenakan: Rp 2.500 untuk lebih dari 500 gram, atau Rp 15 per gram untuk kurang dari 500 gram. Jika berat parsel 10 kg atau lebih, biaya pengiriman ditetapkan pada Rp 100.000. Program kemudian menampilkan berat parsel serta rincian biaya.

Output:





5. Buatlah program yang menerima input sebuah bilangan bulat b dan b > 1. Program harus dapat mencari dan menampilkan semua faktor dari bilangan tersebut! 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):

   Source Code:

```
package main

import "fmt"

func main() {
        var bil1, bil2, i int
        var prima1, prima2 bool
```

```
	fmt.Print("Bilangan 1: ")
	fmt.Scanln(&bil1)

	fmt.Print("Faktor: ")
	prima1 = true
	for i = 1; i <= bil1; i++ {
		if bil1%i == 0 {
			fmt.Print(i, " ")
			if i != 1 && i != bil1 {
				prima1 = false
			}
		}
	}
	fmt.Println()

	fmt.Print("Bilangan 2: ")
	fmt.Scanln(&bil2)

	fmt.Print("Faktor: ")
	prima2 = true
	for i = 1; i <= bil2; i++ {
		if bil2%i == 0 {
			fmt.Print(i, " ")
			if i != 1 && i != bil2 {
				prima2 = false
			}
		}
	}
	fmt.Println()

	if bil1 == 1 {
		prima1 = false
	}
	if bil2 == 1 {
		prima2 = false
	}

	fmt.Println("Prima 1:", prima1)
	fmt.Println("Prima 2:", prima2)
}
```

Penjelasan:

Kode ini meminta pengguna untuk memasukkan dua bilangan, lalu menampilkan faktor-faktor dari masing-masing bilangan serta menentukan apakah bilangan tersebut adalah bilangan prima. Program memeriksa faktor bilangan dengan iterasi dari 1

hingga bilangan tersebut, mencetak setiap faktor, dan menetapkan variabel boolean `prima1` dan `prima2` berdasarkan apakah bilangan hanya memiliki faktor 1 dan dirinya sendiri. Jika ya, bilangan tersebut dianggap prima, kecuali jika bilangan adalah 1, yang secara eksplisit dianggap bukan bilangan prima. Hasilnya berupa apakah bilangan pertama dan kedua adalah prima.

Output: