**LAPORAN PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL 2**

**REVIEW STRUKTUR KONTROL**



Oleh:

NAMA: Lutfi Shidqi Mardian

NIM: 103112400077

KELAS: IF-12-01

**S1 TEKNIK INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**

**2025**

1. **DASAR TEORI**

Dalam pemrograman, struktur kontrol merupakan salah satu konsep fundamental yang menentukan bagaimana alur eksekusi program dijalankan. Struktur kontrol memungkinkan program untuk mengambil keputusan (percabangan), mengulangi eksekusi kode (perulangan), dan menjalankan blok kode tertentu sesuai dengan kondisi yang diberikan.

**1. Percabangan (Conditional Statements)**

Percabangan digunakan untuk mengeksekusi suatu bagian kode berdasarkan suatu kondisi tertentu. Dalam bahasa Go (Golang), percabangan dapat dilakukan dengan pernyataan if, if-else, dan switch.

* **If Statement:** Digunakan untuk menjalankan blok kode jika suatu kondisi bernilai benar (true).

if nilai > 75 {

fmt.Println("Lulus")

}

* **If-Else Statement:** Digunakan jika terdapat dua kemungkinan kondisi yang harus diproses.

if nilai >= 75 {

fmt.Println("Lulus")

} else {

fmt.Println("Tidak Lulus")

}

* **Switch Statement:** Alternatif dari if-else yang lebih bersih untuk menangani banyak kondisi.

switch nilai {

case 80:

fmt.Println("Bagus")

case 90:

fmt.Println("Sangat Bagus")

default:

fmt.Println("Perlu Perbaikan")

}

**2. Perulangan (Looping)**

Perulangan digunakan untuk mengeksekusi blok kode secara berulang selama kondisi tertentu terpenuhi. Dalam Golang, perulangan dilakukan menggunakan for.

* **For Loop dengan Inisialisasi, Kondisi, dan Iterasi**

for i := 1; i <= 5; i++ {

fmt.Println("Iterasi ke-", i)

}

* **For Loop dengan Kondisi Saja** (Mirip dengan while dalam bahasa lain)

i := 1

for i <= 5 {

fmt.Println("Iterasi ke-", i)

i++

}

* **For Loop Tanpa Kondisi** (Loop tak terbatas, harus memiliki break untuk keluar)

for {

fmt.Println("Loop tanpa henti")

break

}

1. **GUIDED**

**1.**

*package* main

*import* "fmt"

*func* *main*() {

*var* *greetings* *=* "selamat datang di dunia DAP"

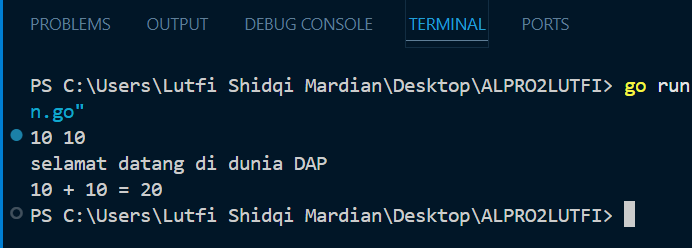
*var* *a*, *b* *int*

*fmt*.*Print*(*greetings*)

*fmt*.*Printf*("%v + %v = %v\n", *a*, *b*, *a+b*)

}

**Output Screenshot:**

****

**Penjelasan:**

Dalam program ini terdapat variabel greetings dengan tipe data string yang langsung di deklarasikan dengan “selamat datang di dunia DAP” sedangkan untuk variabel a dan b bertipe data integer yang harus di input kan oleh pengguna. Ketika dijalankan program akan mencetak keluar variable greetings berupa “selamat datang di dunia DAP” di terminal dan mencetak hasil penjumlahan dari variabel a dan b yang di input oleh pengguna.

**2.**

*package* main

*import* "fmt"

*func* *main*() {

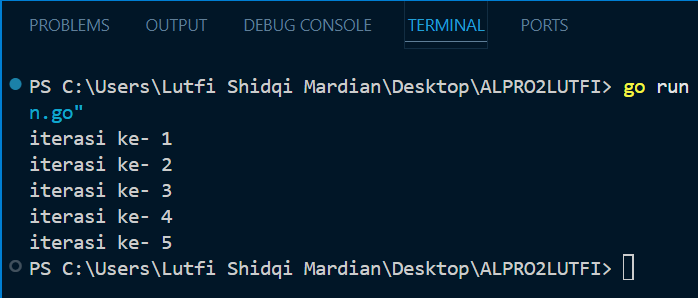
*for* *i* *:=* 1; *i* *<=* 5; *i++* {

*fmt*.*Println*("iterasi ke-", *i*)

    }

}

**Output Screenshot:**

****

**Penjelasan:**

Program ini mencetak kalimat “iterasi ke- i” yang mana I merupakan variabel iterasi yang akan terus bertambah selama dalam kondisi perulangan nya nilai I tersebut masih kurang dari/sama dengan batas perulangannya, disini yang menjadi batas perulangannya adalah 5 yang berarti pencetakan kalimat akan dilakukan sampai iterasi I bertambah 1 kali hingga mencapai 5 lalu perulangannya berhenti.

**3.**

*package* main

*import* "fmt

*func* *main*() {

*nilai* *:=* 80

*pctHadir* *:=* 0.75

*adaTubes* *:=* true

*var* *indeks* *string*

*if* *nilai* *>* 75 *&&* *adaTubes* {

*indeks* *=* "A"

    } *else* *if* *nilai* *>* 65 {

*indeks* *=* "B"

    } *else* *if* *nilai* *>* 50 *&&* *pctHadir* *>* 0.7 {

*indeks* *=* "C"

    } *else* {

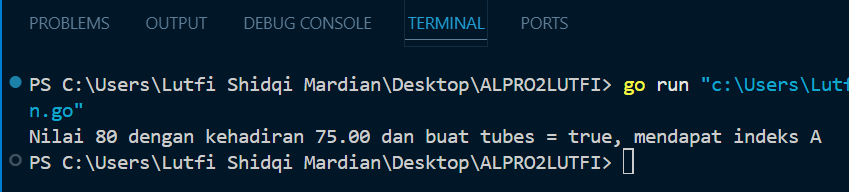
*indeks* *=* "F"

    }

*fmt*.*Printf*("Nilai %v dengan kehadiran %0.2f dan buat tubes= %t, mendapat indeks %s\n", *nilai*, *pctHadir\**100, *adaTubes*, *indeks*)

}

**Output Screenshot:**

****

**Penjelasan:**

Program ini digunakan untuk mengetahui indeks mahasiswa berdasarkan keterangan dari variabel nilai, pctHadir (persentase hadir), dan adaTubes (mengerjakan tugas besar atau tidak), variabel nilai dan adaTubes sendiri sudah di deklarasikan dengan 80;true maka dari itu langsung memenuhi kondisi percabangan Dimana indeksnya adalah “A”. Lalu program pun mencetak “Nilai 80 dengan kehadiran 75.00 dan buat tubes = true, mendapat indeks A” sesuai dengan variabel yang ada dan hasil indeks yang sesuai dengan kondisi yang ada.

1. **UNGUIDED**

**1. 2A Nomor 2**

*package* main

*import* "fmt"

*func* *main*() {

*var* *t* *int*

*var* *ket* *bool*

*fmt*.*Scan*(*&t*)

*if* *t* *%* 400 *==* 0 *||* *t* *%* 4 *==* 0 {

*ket* *=* true

    } *else* *if* *t* *%* 100 *!=* 0 {

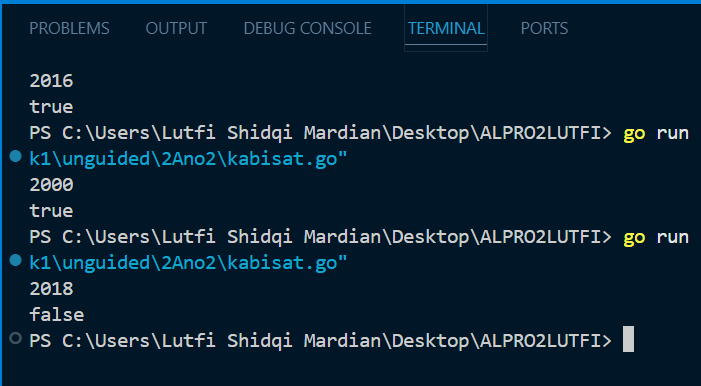
*ket* *=* false

    }

*fmt*.*Print*(*ket*)

}

**Output Screenshot:**

****

**Penjelasan:**

Program ini digunakan untuk mengetahui apakah input variabel t (tahun) dari pengguna adalah tahun kabisat dengan cara mengeceknya melalui beberapa kondisi percabangan yang ada di dalam program tersebut, output berupa keterangan Boolean true atau false tergantung input user.

**2. 2B Nomor 4**

*package* main

*import* "fmt"

*func* *main*() {

*var* *K* *int*

*fmt*.*Scan*(*&K*)

*akr2* *:=* 1.0

*for* *k* *:=* 0; *k* *<=* *K*; *k++* {

*akr2* *\*=* ((4*\*float64*(*k*) *+* 2) *\** (4*\*float64*(*k*) *+* 2)) */*

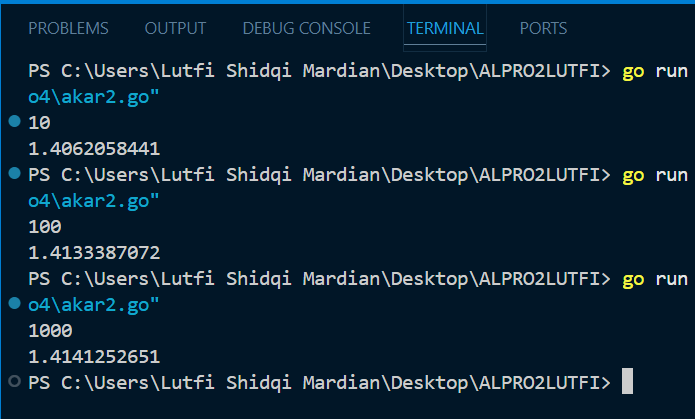
            ((4*\*float64*(*k*) *+* 1) *\** (4*\*float64*(*k*) *+* 3))

    }

*fmt*.*Printf*("%.10f\n", *akr2*)

}

**Output Screenshot:**

****

**Penjelasan:**

Program ini digunakan untuk menghitung nilai dari suatu deret menggunakan perulangan for dengan nilai K sebagai input dari pengguna dan variabel akr2 diinisialisasi dengan 1.0 sebagai nilai awalnya.

Variabel k(kecil) sama seperti variabel i didalam for i := 0; i <= K; i++, yang bertugas sebagai peng iterasian yang terus berjalan sampai terkena batas K yang diinput oleh pengguna, dan pada setiap akhir iterasi akr2 akan dikalikan dengan hasil perhitungan berdasarkan rumus:

(4k + 2)2 / (4k + 1) \* (4k + 3)

Setelah perulangan selesai, program mencetak hasil akhir akr2 dengan presisi 10 angka di belakang desimal.

**3. 2C Nomor 1**

*package* main

*import* "fmt"

*func* *main*() {

*var* *berat*, *kg*, *gr*, *biaya*, *tambahan* *int*

*fmt*.*Scan*(*&berat*)

*kg* *=* *berat* */* 1000

*gr* *=* *berat* *%* 1000

*biaya* *=* *kg* *\** 10000

*if* *gr* *>=* 500 {

*tambahan* *=* *gr* *\** 5

    } *else* *if* *gr* *<* 500 {

*tambahan* *=* *gr* *\** 15

    }

*total* *:=* *biaya* *+* *tambahan*

*if* *kg* *>* 10 {

*fmt*.*Println*("Berat parsel (gram): ", *berat*)

*fmt*.*Printf*("Detail berat : %v kg + %v gr\n", *kg*, *gr*)

*fmt*.*Printf*("Detail biaya: Rp. %v + Rp. %v\n", *biaya*, *tambahan*)

*fmt*.*Printf*("Total biaya: Rp. %v", *biaya*)

    } *else* *if* *kg* *<* 10 {

*fmt*.*Println*("Berat parsel (gram): ", *berat*)

*fmt*.*Printf*("Detail berat : %v kg + %v gr\n", *kg*, *gr*)

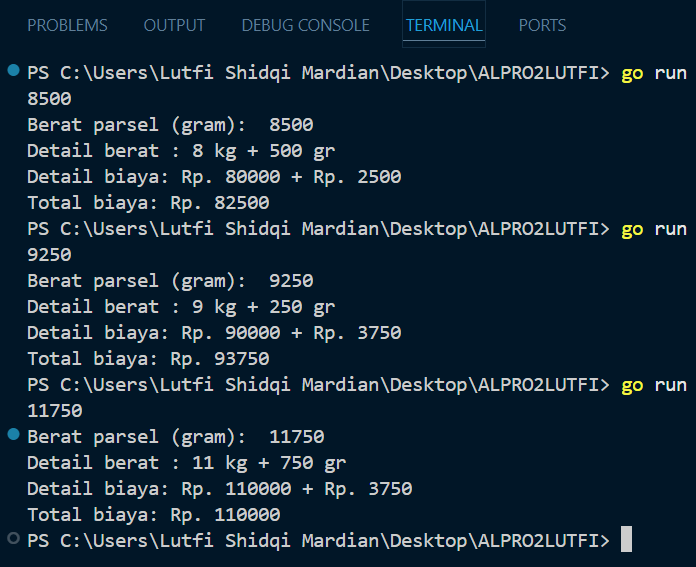
*fmt*.*Printf*("Detail biaya: Rp. %v + Rp. %v\n", *biaya*, *tambahan*)

*fmt*.*Printf*("Total biaya: Rp. %v", *total*)

    }

}

**Output Screenshot:**



**Penjelasan:**

Program ini menghitung biaya pengiriman parsel berdasarkan berat dalam gram yang diinput oleh pengguna. Berat dikonversi menjadi kilogram (kg) dan sisa gram (gr), lalu biaya dasar dihitung dengan tarif Rp. 10.000 per kg. Jika terdapat sisa gram, biaya tambahan ditentukan berdasarkan ketentuan: Rp. 5 per gram untuk sisa ≥ 500 gram, dan Rp. 15 per gram untuk sisa < 500 gram. Total biaya adalah hasil penjumlahan biaya dasar dan biaya tambahan, kecuali jika berat lebih dari 10 kg, di mana biaya hanya dihitung berdasarkan kilogram tanpa tambahan biaya gram. Program kemudian mencetak detail berat dan biaya dengan format yang jelas sesuai dengan kondisi berat barang.

1. **KESIMPULAN**

Dalam praktikum ini telah dilakukan implementasi struktur kontrol dalam bahasa Golang, termasuk percabangan dengan if-else dan switch serta perulangan dengan for loop. Percabangan digunakan untuk mengeksekusi kode berdasarkan kondisi tertentu, seperti menentukan indeks mahasiswa berdasarkan nilai dan kehadiran. Sementara itu, perulangan digunakan untuk mengulang eksekusi kode, seperti mencetak iterasi dan menghitung nilai dari suatu deret.

Setiap program yang dibuat menunjukkan bagaimana struktur kontrol bekerja dalam berbagai skenario, mulai dari pengambilan keputusan hingga perulangan berdasarkan kondisi tertentu. Implementasi program juga menunjukkan bagaimana penggunaan logika dalam pemrograman dapat membantu menyelesaikan permasalahan nyata, seperti menentukan tahun kabisat atau menghitung biaya pengiriman berdasarkan berat barang.

Dari hasil percobaan, dapat disimpulkan bahwa struktur kontrol berperan penting dalam alur eksekusi program. Dengan memahami dan menerapkannya dengan benar, program dapat berjalan lebih efisien dan sesuai dengan kebutuhan logika yang diinginkan.

1. **REFERENSI**
2. Donovan, A. A., & Kernighan, B. W. (2015). *The Go Programming Language*. Addison-Wesley.
3. Official Golang Documentation. (n.d.). *Control Flow Statements*. Retrieved from <https://golang.org/doc/>
4. W3Schools. (n.d.). *Golang If-Else Statements*. Retrieved from https://www.w3schools.com/go/go\_conditions.asp