

**LAPORAN PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN 2  
MODUL 2  
REVIEW STRUKTUR KONTROL**



Oleh:

HAKAN ISMAIL AFNAN

103112400038

12 IF 01

**S1 TEKNIK INFORMATIKA  
TELKOM UNIVERSITY PURWOKERTO  
2025**

## I. DASAR TEORI

### 1. Struktur Program Bahasa Go

Bahasa pemrograman Go, memiliki struktur program yang berbeda, di mana setiap program pertama terdiri dari dua komponen penting, yaitu:

- **package main:** Menandakan bahwa file ini adalah program utama yang dapat dieksekusi.
- **func main():** Fungsi utama tempat eksekusi program dimulai. Selain itu, komentar dalam Go dapat dibuat dengan dua cara:
  - `//` untuk komentar satu baris.
  - `/*` untuk komentar lebih dari satu baris.

### 2. Koding, Kompilasi, dan Eksekusi Program Go

- Program Go harus disimpan dalam file berekstensi **.go**.
- Kompilasi dilakukan menggunakan perintah **go build**.
- Setelah dikompilasi, program dapat dieksekusi langsung melalui terminal.
- Bahasa Go adalah bahasa yang dikompilasi, sehingga program harus melewati tahap kompilasi sebelum dijalankan.

### 3. Tipe Data dan Variabel dalam Go

Variabel dalam Go harus dideklarasikan sebelum digunakan dan memiliki tipe data yang ketat. Beberapa tipe data yang umum digunakan antara lain:

#### 3.1 Tipe Data dalam Go dan Fungsinya

1. **Integer (int, int8, int16, int32, int64, uint, uint8, uint16, uint32, uint64)**
  - Digunakan untuk menyimpan bilangan bulat.
  - Contoh penggunaan:
    - `var angka int = 10`
    - `var kecil int8 = 127`
    - `var besar int64 = 1000000000`
2. **Float (float32, float64)**
  - Digunakan untuk menyimpan bilangan desimal.
  - Contoh penggunaan:
    - `var suhu float64 = 36.5`
3. **Boolean (bool)**
  - Digunakan untuk menyimpan nilai **true** atau **false**.
  - Contoh penggunaan:
    - `var aktif bool = true`

#### 4. Karakter (byte, rune)

- byte adalah alias dari uint8 dan digunakan untuk menyimpan karakter dalam format ASCII.
- rune adalah alias dari int32 dan digunakan untuk menyimpan karakter Unicode.
- Contoh penggunaan:
- `var huruf byte = 'A' // ASCII`

`var simbol rune = '€' // Unicode`

#### 5. String

- Digunakan untuk menyimpan teks.
- Contoh penggunaan:

`var nama string = "Go Language"`

- String bisa diakses sebagai array karakter:

`fmt.Println(nama[0]) // Menampilkan karakter pertama`

### 3.2 Deklarasi dan Inisialisasi Variabel dalam Go

Variabel harus dideklarasikan sebelum digunakan. Ada beberapa cara untuk mendeklarasikan variabel:

- **Deklarasi tanpa inisialisasi**

`var a int // Nilai default adalah 0`

- **Deklarasi dengan inisialisasi**

- `var b int = 10`

`var c = 20 // Go secara otomatis menentukan tipe data`

- **Deklarasi menggunakan shorthand :=**

`d := 30 // Sama dengan var d int = 30`

### 3.3 Konversi Tipe Data

Konversi tipe data dalam Go dilakukan secara eksplisit dengan menuliskan tipe yang diinginkan:

`var x int = 10`

`var y float64 = float64(x) // Konversi dari int ke float64`

`var z string = strconv.Itoa(x) // Konversi dari int ke string`

#### 4. Struktur Kontrol dalam Go

Struktur kontrol dalam Go terdiri dari dua kategori utama:

##### 1. Perulangan (Looping):

- **For-loop** digunakan untuk mengulang eksekusi kode selama suatu kondisi terpenuhi.
- **While-loop** dalam Go diimplementasikan dengan `for` tanpa inisialisasi dan pembaruan.
- **Repeat-until** dapat diimplementasikan menggunakan `for` dengan kondisi penghentian di dalam loop.

##### 2. Percabangan (Branching):

- **If-else** digunakan untuk menjalankan kode berdasarkan suatu kondisi.
- **Switch-case** memungkinkan eksekusi kode berdasarkan nilai tertentu.

#### 5. Konstanta Simbolik

Konstanta dalam Go dideklarasikan menggunakan kata kunci **const**.

Contoh:

```
const PI = 3.1415926535
```

```
const MARKER = "AKHIR"
```

#### 6. Operasi Dasar dalam Go

Operasi yang dapat dilakukan pada tipe data di Go antara lain:

- **Aritmatika:** `+`, `-`, `*`, `/`, `%`
- **Logika:** `&&`, `||`, `!`
- **Perbandingan:** `==`, `!=`, `<`, `>`, `<=`, `>=`
- **Bitwise:** `&`, `|`, `^`, `<<`, `>>`

#### 7. Implementasi dalam Praktikum

Modul ini berisi berbagai latihan untuk mengimplementasikan konsep dasar pemrograman dalam Go, seperti:

- Membuat program untuk membaca dan menampilkan data.
- Menentukan apakah suatu tahun merupakan tahun kabisat.
- Menghitung volume dan luas bola.
- Mengonversi temperatur dari Celsius ke Fahrenheit, Reamur, dan Kelvin.
- Menampilkan faktor dari suatu bilangan dan menentukan apakah bilangan tersebut prima.

## II. GUIDED

Source Code + Screenshot hasil program beserta penjelasan

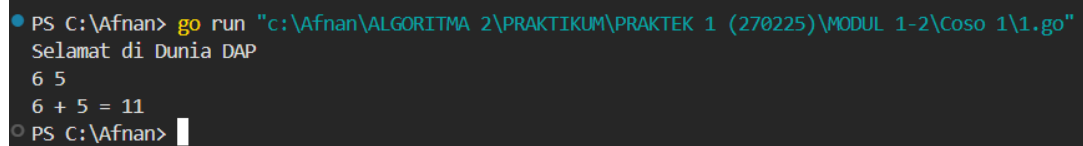
### Contoh 1

```
package main

import "fmt"

func main() {
    var greetings = "Selamat di Dunia DAP"
    var a, b int
    fmt.Println(greetings)
    fmt.Scan(&a, &b)
    fmt.Printf("%v + %v = %v\n", a, b, a+b)
}
```

### Screenshots Output



```
PS C:\Afnan> go run "c:\Afnan\ALGORITMA 2\PRAKTIKUM\PRAKTEK 1 (270225)\MODUL 1-2\Coso 1\1.go"
Selamat di Dunia DAP
6 5
6 + 5 = 11
PS C:\Afnan>
```

// Foto hasil dari menjalankan code

Deskripsi:

Tujuan dari kode tersebut adalah untuk membaca dua angka dari pengguna, menjumlahkannya, dan menampilkan hasilnya ke layar. Program ini juga menampilkan pesan selamat datang sebelum meminta input dari pengguna.

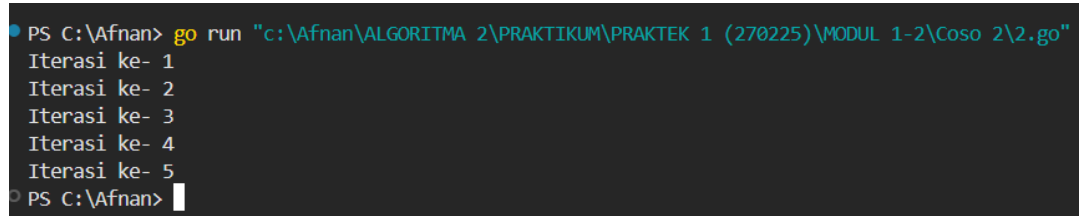
## Contoh 2

```
package main

import "fmt"

func main() {
    for i := 1; i <= 5; i++ {
        fmt.Println("Iterasi ke-", i)
    }
}
```

### Screenshots Output



```
PS C:\Afnan> go run "c:\Afnan\ALGORITMA 2\PRAKTIKUM\PRAKTEK 1 (270225)\MODUL 1-2\Coso 2\2.go"
Iterasi ke- 1
Iterasi ke- 2
Iterasi ke- 3
Iterasi ke- 4
Iterasi ke- 5
PS C:\Afnan>
```

// Foto hasil dari menjalankan code

### Deskripsi:

Tujuan dari program ini adalah untuk melakukan iterasi sebanyak 5 kali menggunakan perulangan for, lalu mencetak teks "Iterasi ke- 1" di setiap iterasi.

### Contoh 3

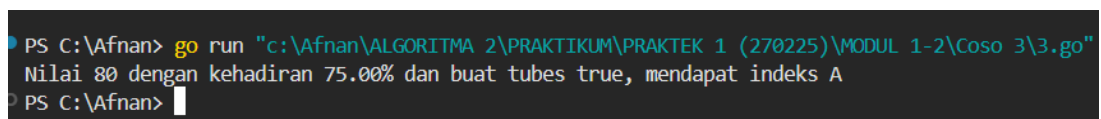
```
package main

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 %d dengan kehadiran %.2f%% dan buat tubes %t, mendapat indeks %s\n", nilai, pctHadir*100, adaTubes, indeks)
}
```

### Screenshots Output



```
PS C:\Afnan> go run "c:\Afnan\ALGORITMA 2\PRAKTIKUM\PRAKTEK 1 (270225)\MODUL 1-2\Coso 3\3.go"
Nilai 80 dengan kehadiran 75.00% dan buat tubes true, mendapat indeks A
PS C:\Afnan>
```

// Foto hasil dari menjalankan code

Deskripsi:

### Tujuan Program

Program ini bertujuan untuk menentukan **indeks nilai** berdasarkan nilai akademik, persentase kehadiran, dan keberadaan tugas besar (tubes).

### Cara Kerja Program

#### 1. Menentukan variabel awal:

- o nilai (80): Menyimpan nilai akademik.

- pctHadir (0.75): Menyimpan persentase kehadiran (dalam desimal).
- adaTubes (true): Menyimpan status apakah ada tugas besar atau tidak.
- indeks: Variabel kosong yang nantinya akan diisi berdasarkan kondisi tertentu.

2. **Menentukan indeks berdasarkan kondisi:**

- Jika nilai > 75 **dan** ada tugas besar (adaTubes == true), maka indeks = "A".
- Jika nilai > 65, maka indeks = "B".
- Jika nilai > 50 **dan** kehadiran lebih dari 70% (pctHadir > 0.7), maka indeks = "C".
- Jika tidak memenuhi syarat di atas, maka indeks = "F".

3. **Menampilkan hasil akhir** dengan mencetak nilai, persentase kehadiran (dikonversi ke persen), status tugas besar, dan indeks yang diperoleh.



### III. UNGUIDED

Source Code + Screenshot hasil program beserta penjelasan

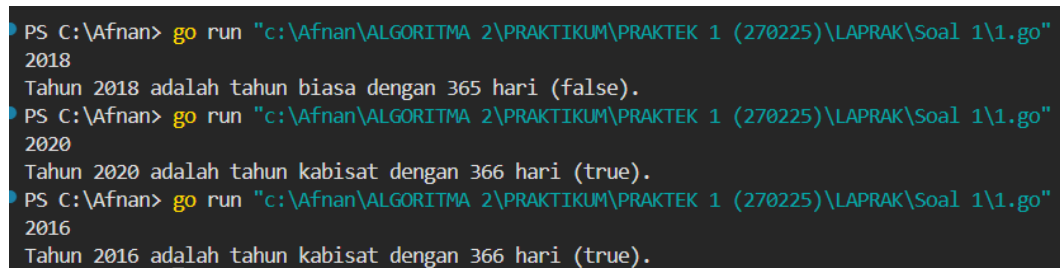
#### Soal 1

```
//HAKAN ISMAIL AFNAN
//103112400038
package main

import (
    "fmt"
)

func main() {
    var tahun int
    fmt.Print("Masukkan tahun yang ingin diperiksa: ")
    fmt.Scanln(&tahun)
    switch {
    case (tahun%4 == 0 && tahun%100 != 0) || (tahun%400 == 0):
        fmt.Printf("%d merupakan tahun kabisat dengan 366 hari.\n", tahun)
    default:
        fmt.Printf("%d bukan tahun kabisat, hanya memiliki 365 hari.\n", tahun)
    }
}
```

#### Screenshots Output



```
PS C:\Afnan> go run "c:\Afnan\ALGORITMA 2\PRAKTIKUM\PRAKTEK 1 (270225)\LAPRAK\Soal 1\1.go"
2018
Tahun 2018 adalah tahun biasa dengan 365 hari (false).
PS C:\Afnan> go run "c:\Afnan\ALGORITMA 2\PRAKTIKUM\PRAKTEK 1 (270225)\LAPRAK\Soal 1\1.go"
2020
Tahun 2020 adalah tahun kabisat dengan 366 hari (true).
PS C:\Afnan> go run "c:\Afnan\ALGORITMA 2\PRAKTIKUM\PRAKTEK 1 (270225)\LAPRAK\Soal 1\1.go"
2016
Tahun 2016 adalah tahun kabisat dengan 366 hari (true).
```

// Foto hasil dari menjalankan code

Deskripsi:

#### Tujuan Program

Program ini bertujuan untuk **menentukan apakah suatu tahun merupakan tahun kabisat atau bukan** berdasarkan aturan penanggalan **Gregorian**.

---

## Cara Kerja Program

### 1. Meminta Input Tahun dari Pengguna

- Pengguna diminta memasukkan tahun yang ingin diperiksa.

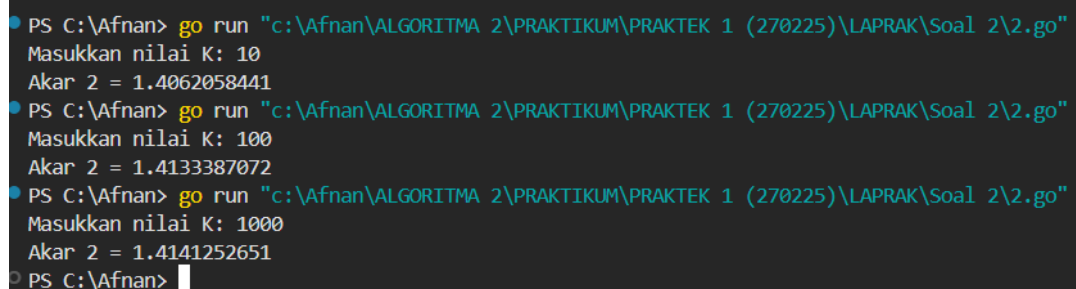
### 2. Menentukan Apakah Tahun Kabisat atau Bukan

- Program menggunakan struktur **switch case** dengan kondisi berikut:
  - **Tahun kabisat** terjadi jika:
    - Tahun habis dibagi **4**, tetapi **tidak habis dibagi 100**, atau
    - Tahun habis dibagi **400**.
  - Jika salah satu kondisi di atas terpenuhi, maka tahun dianggap **kabisat** (memiliki **366 hari**).
  - Jika tidak, maka tahun dianggap **bukan tahun kabisat** (memiliki **365 hari**).

## Soal 2

```
//HAKAN ISMAIL AFNAN
//103112400038
package main
import (
    "fmt"
    "math"
)
func hitungAkar2(K int) float64 {
    var hasil float64 = 1.0
    for k := 0; k <= K; k++ {
        nom := math.Pow(float64(4*k+2), 2)
        den := float64((4*k + 1) * (4*k + 3))
        hasil *= nom / den
    }
    akar2 := hasil
    return akar2
}
func main() {
    var K int
    fmt.Print("Masukkan nilai K: ")
    fmt.Scan(&K)
    akar2 := hitungAkar2(K)
    fmt.Printf("Akar 2 = %.10f\n", akar2)
}
```

## Screenshots Output



```
PS C:\Afnan> go run "c:\Afnan\ALGORITMA 2\PRAKTIKUM\PRAKTEK 1 (270225)\LAPRAK\Soal 2\2.go"
Masukkan nilai K: 10
Akar 2 = 1.4062058441
PS C:\Afnan> go run "c:\Afnan\ALGORITMA 2\PRAKTIKUM\PRAKTEK 1 (270225)\LAPRAK\Soal 2\2.go"
Masukkan nilai K: 100
Akar 2 = 1.4133387072
PS C:\Afnan> go run "c:\Afnan\ALGORITMA 2\PRAKTIKUM\PRAKTEK 1 (270225)\LAPRAK\Soal 2\2.go"
Masukkan nilai K: 1000
Akar 2 = 1.4141252651
PS C:\Afnan>
```

// Foto hasil dari menjalankan code

Deskripsi:

**Tujuan Program**

Program ini bertujuan untuk **menghitung nilai aproksimasi dari akar 2 ( $\sqrt{2}$ )** menggunakan metode perkalian pecahan berulang berdasarkan nilai K yang dimasukkan oleh pengguna.

---

## Cara Kerja Program

### 1. Meminta Input dari Pengguna

- Pengguna memasukkan nilai K, yaitu jumlah iterasi yang digunakan untuk menghitung aproksimasi akar 2.

### 2. Menghitung Aproksimasi Akar 2

- Fungsi `hitungAkar2(K)` digunakan untuk menghitung **perkalian pecahan berulang** sebagai pendekatan nilai akar 2.
- Perhitungan dilakukan dengan formula:

$$\text{hasil} = \prod_{k=0}^K \frac{(4k+2)^2}{(4k+1)(4k+3)} \quad \text{hasil} = \prod_{k=0}^K \frac{(4k+2)^2}{(4k+1)(4k+3)}$$

- $\text{nom} = (4*k + 2)^2 \rightarrow$  Menghitung pembilang (numerator).
- $\text{den} = (4*k + 1) * (4*k + 3) \rightarrow$  Menghitung penyebut (denominator).
- Hasil dikalikan secara iteratif ( $\text{hasil} *= \text{nom} / \text{den}$ ).

### 3. Menampilkan Hasil Perhitungan

- Program mencetak nilai aproksimasi akar 2 dengan **presisi 10 desimal**.

### Soal 3

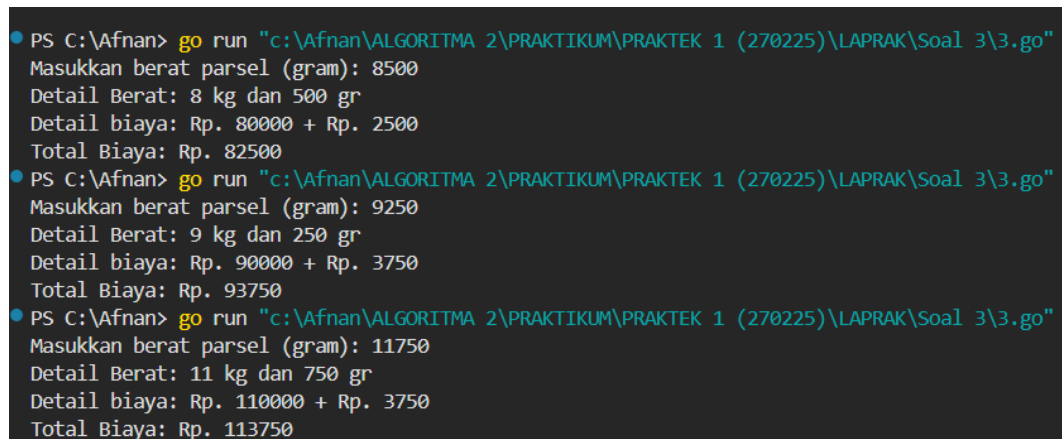
```
//HAKAN ISMAIL AFNAN
//103112400038

package main

import (
    "fmt"
)

func main() {
    var berat, hargaGram int
    fmt.Print("Masukkan berat parsel (gram): ")
    fmt.Scanln(&berat)
    kilogram := berat / 1000
    sisaGram := berat % 1000
    fmt.Printf("Detail Berat: %d kg dan %d gr\n", kilogram, sisaGram)
    hargaKg := kilogram * 10000
    if sisaGram >= 500 {
        hargaGram = sisaGram * 5
    } else {
        hargaGram = sisaGram * 15
    }
    fmt.Printf("Detail biaya: Rp. %d + Rp. %d\n", hargaKg, hargaGram)
    fmt.Printf("Total Biaya: Rp. %d\n", hargaKg+hargaGram)
}
```

### Screenshots Output



```
PS C:\Afnan> go run "c:\Afnan\ALGORITMA 2\PRAKTIKUM\PRAKTEK 1 (270225)\LAPRAK\Soal 3\3.go"
Masukkan berat parsel (gram): 8500
Detail Berat: 8 kg dan 500 gr
Detail biaya: Rp. 80000 + Rp. 2500
Total Biaya: Rp. 82500
PS C:\Afnan> go run "c:\Afnan\ALGORITMA 2\PRAKTIKUM\PRAKTEK 1 (270225)\LAPRAK\Soal 3\3.go"
Masukkan berat parsel (gram): 9250
Detail Berat: 9 kg dan 250 gr
Detail biaya: Rp. 90000 + Rp. 3750
Total Biaya: Rp. 93750
PS C:\Afnan> go run "c:\Afnan\ALGORITMA 2\PRAKTIKUM\PRAKTEK 1 (270225)\LAPRAK\Soal 3\3.go"
Masukkan berat parsel (gram): 11750
Detail Berat: 11 kg dan 750 gr
Detail biaya: Rp. 110000 + Rp. 3750
Total Biaya: Rp. 113750
```

// Foto hasil dari menjalankan code

Deskripsi:

## **Tujuan Program**

**Program ini bertujuan untuk menghitung total biaya pengiriman parcel berdasarkan beratnya dalam gram, dengan tarif yang berbeda untuk berat di atas atau di bawah 500 gram.**

---

## **Cara Kerja Program**

### **1. Meminta Input dari Pengguna**

- **Pengguna diminta memasukkan berat parcel dalam satuan gram (gram).**

### **2. Menghitung Berat dalam Kilogram dan Sisa Gram**

- **Program menghitung jumlah kilogram (kilogram = berat / 1000) dan sisa gram setelah pembagian (sisaGram = berat % 1000).**

### **3. Menghitung Biaya Berdasarkan Berat**

- **Biaya per kilogram ditetapkan sebesar Rp. 10.000.**
- **Biaya untuk sisa gram dihitung dengan aturan:**
  - **Jika  $\geq 500$  gram, tarifnya Rp. 5 per gram.**
  - **Jika  $< 500$  gram, tarifnya Rp. 15 per gram.**

### **4. Menampilkan Rincian Berat dan Biaya**

- **Program mencetak rincian berat dalam kg dan gram.**
- **Menampilkan perhitungan biaya per kilogram dan sisa gram.**
- **Menghitung dan mencetak total biaya pengiriman.**

## KESIMPULAN

Dari berbagai program yang telah dianalisis, dapat disimpulkan bahwa bahasa pemrograman Go (Golang) memiliki struktur yang sederhana namun sangat efisien dalam menangani berbagai skenario pemrograman. Berikut beberapa poin utama yang dapat disimpulkan dari materi yang telah dibahas:

### 1. Struktur Dasar Program dalam Go

- Setiap program utama menggunakan package main dan fungsi utama func main() sebagai titik eksekusi.
- Komentar dalam Go bisa menggunakan // untuk satu baris dan /\* \*/ untuk banyak baris.

### 2. Koding, Kompilasi, dan Eksekusi Program

- Program Go harus disimpan dalam file berekstensi .go.
- Kompilasi dilakukan dengan perintah go build, dan eksekusi dilakukan langsung melalui terminal.

### 3. Tipe Data dan Variabel dalam Go

- Go memiliki tipe data yang ketat seperti int, float, string, bool, byte, dan rune.
- Variabel dapat dideklarasikan secara eksplisit (var x int = 10) atau menggunakan shorthand (x := 10).
- Konversi tipe data dilakukan secara eksplisit menggunakan fungsi bawaan Go.

### 4. Struktur Kontrol dalam Go

- Percabangan menggunakan if-else dan switch-case untuk menangani keputusan berbasis kondisi.
- Perulangan (looping) menggunakan for untuk mengulang eksekusi berdasarkan kondisi tertentu.

## 5. Konstanta Simbolik

- Konstanta dalam Go dideklarasikan dengan `const`, contohnya `const PI = 3.14159`.

## 6. Operasi Dasar dalam Go

- Mendukung operasi aritmatika (+, -, \*, /, %), logika (&&, ||, !), perbandingan (==, !=, <, >, <=, >=), dan bitwise (&, |, ^, <<, >>).

## 7. Implementasi dalam Program Praktis

- Menentukan tahun kabisat berdasarkan aturan kalender Gregorian menggunakan percabangan switch-case.
- Menghitung aproksimasi akar 2 ( $\sqrt{2}$ ) dengan metode perkalian pecahan berulang.
- Menghitung total biaya pengiriman parcel berdasarkan berat dengan sistem tarif variatif menggunakan percabangan if-else.
- Menentukan indeks nilai mahasiswa berdasarkan nilai, kehadiran, dan tugas besar dengan menggunakan if-else.
- Menampilkan teks sederhana dan melakukan operasi aritmatika berdasarkan input pengguna.

## Kesimpulan Akhir

Bahasa pemrograman Go memiliki sintaks yang sederhana namun sangat efisien dan cepat karena berbasis kompilasi. Dengan kontrol alur yang fleksibel serta dukungan tipe data yang ketat, Go sangat cocok untuk membangun berbagai aplikasi, mulai dari program sederhana hingga sistem yang kompleks.



## IV. REFERENSI

### □ Mata kuliah Algoritma Pemrograman 2

- Go MODUL 2. REVIEW STRUKTUR KONTROL, Praktikum Algoritma dan Pemrograman 2

### □ Program dalam Bahasa Go

- Donovan, A. A., & Kernighan, B. W. (2015). *The Go Programming Language*. Addison-Wesley.
- Pike, R. (2012). *Go at Google: Language Design in the Service of Software Engineering*. Communications of the ACM, 57(1), 44-56.

### □ Koding, Kompilasi, dan Eksekusi Program Go

- Golang.org. (2024). *Go Documentation - Command go*. Retrieved from <https://golang.org/doc/>
- Kernighan, B. W., & Ritchie, D. M. (1988). *The C Programming Language (2nd Edition)*. Prentice Hall. (Sebagai referensi bahasa yang menginspirasi Go)

### □ Tipe Data dan Variabel dalam Go

- Donovan, A. A., & Kernighan, B. W. (2015). *The Go Programming Language*. Addison-Wesley.
- Ghaffarian, S. M., & Khalaj, B. H. (2017). *Statically Typed vs Dynamically Typed Languages: A Comparative Analysis*. ACM Computing Surveys, 50(4), 1-38.

### □ Struktur Kontrol dalam Go

- Arora, A., & Mehta, P. (2021). *Programming in Go: Loops, Conditional Statements, and Functions*. Springer.
- Mitzenmacher, M., & Upfal, E. (2017). *Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis*. Cambridge University Press. (Menjelaskan konsep di balik pengendalian aliran program berbasis probabilitas seperti dalam percabangan dan perulangan)

#### □ **Konstanta Simbolik**

- Go Documentation. (2024). *Constants in Go*. Retrieved from <https://golang.org/ref/spec#Constants>

#### □ **Operasi Dasar dalam Go**

- Knuth, D. E. (1997). *The Art of Computer Programming, Volume 1: Fundamental Algorithms*. Addison-Wesley.
- Golang.org. (2024). *Arithmetic and Logical Operators in Go*. Retrieved from <https://golang.org/ref/spec#Operators>

#### □ **Implementasi dalam Praktikum (Studi Kasus)**

- ISO/IEC 9899:2018. *Programming Languages — C Standard*. (Sebagai dasar untuk memahami struktur bahasa yang menginspirasi Go)
- Sedgewick, R., & Wayne, K. (2011). *Algorithms (4th Edition)*. Addison-Wesley. (Referensi untuk algoritma komputasi, termasuk perhitungan akar kuadrat, looping, dan percabangan dalam implementasi program di Go)
- Goldberg, D. (1991). *What Every Computer Scientist Should Know About Floating-Point Arithmetic*. ACM Computing Surveys, 23(1), 5-48. (Menjelaskan bagaimana perhitungan floating-point bekerja dalam bahasa pemrograman seperti Go, terutama dalam aproksimasi akar kuadrat)