

**LAPORAN PRAKTIKUM  
PEMROGRAMAN BERORIENTASI OBJEK**

**MODUL II  
REVIEW STRUKTUR KONTROL**



Oleh:

NAMA: Didik Weka Pratama

NIM: 2311102285

KELAS: S1 IF-11-07

**S1 TEKNIK INFORMATIKA  
INSTITUT TEKNOLOGI TELKOM PURWOKERTO  
2024**

## I. DASAR TEORI

Struktur kontrol dalam pemrograman adalah elemen penting yang mengarahkan alur eksekusi program berdasarkan kondisi tertentu. Dalam bahasa Go (Golang), struktur kontrol memberikan cara untuk membuat keputusan, mengulangi tindakan, dan mengelola alur program secara efisien. Berikut adalah beberapa struktur kontrol dasar dalam Go:

### 1. Struktur Kontrol Seleksi

- **If Statement:**

- Digunakan untuk mengeksekusi blok kode jika suatu kondisi terpenuhi.

- Sintaks:

```
if kondisi {  
    // Eksekusi jika kondisi benar  
}
```

- **If-Else Statement:**

- Menyediakan alternatif eksekusi jika kondisi tidak terpenuhi.

- Sintaks:

```
if kondisi {  
    // Eksekusi jika kondisi benar  
} else {  
    // Eksekusi jika kondisi salah  
}
```

- **If-Else If-Else Statement:**

- Memungkinkan pemeriksaan beberapa kondisi.

- Sintaks:

```
if kondisi1 {  
    // Eksekusi jika kondisi1 benar  
} else if kondisi2 {  
    // Eksekusi jika kondisi2 benar  
} else {  
    // Eksekusi jika semua kondisi salah  
}
```

- **Switch Statement:**

- Memilih satu dari beberapa blok kode untuk dieksekusi berdasarkan nilai ekspresi.
- Sintaks:

```
switch ekspresi {  
  case nilai1:  
    // Eksekusi jika ekspresi == nilai1  
  case nilai2:  
    // Eksekusi jika ekspresi == nilai2  
  default:  
    // Eksekusi jika tidak ada kasus yang cocok  
}
```

## 2. Struktur Kontrol Perulangan

- **For Loop:**

- Satu-satunya struktur perulangan dalam Go, digunakan untuk berbagai jenis iterasi.
- Sintaks:

```
for init; kondisi; pembaruan {  
  // Blok kode yang diulang  
}
```

- **For Range Loop:**

- Digunakan untuk mengiterasi elemen dalam array, slice, map, atau channel.
- Sintaks:

```
for indeks, nilai := range koleksi {  
  // Akses setiap elemen  
}
```

- **While Loop (Simulasi):**

- Tidak ada while loop eksplisit di Go, tetapi dapat disimulasikan dengan for loop.
- Sintaks:

```
for kondisi {  
  // Blok kode yang diulang  
}
```

- **Infinite Loop:**

- Loop yang berjalan tanpa batas, dihentikan dengan break.
- Sintaks:

```
for {  
  // Eksekusi terus menerus  
  if someCondition {  
    break  
  }  
}
```

### 3. Struktur Kontrol

- **Break and Continue Branch:**

"Break" adalah istilah yang digunakan untuk keluar dari perselisihan. terus digunakan untuk melanjutkan ke iterasi berikutnya.

- **Fallthrough, atau Switch:**

Meskipun kondisi case saat ini terpenuhi, switch melakukan fallthrough untuk melanjutkan eksekusi ke case berikutnya.

### 4. Penerapan Sistem Kontrol

- Efisiensi: Struktur kontrol menjamin bahwa program hanya menjalankan kode yang diperlukan.
- Keputusan dinamis memungkinkan pengambilan keputusan berdasarkan kondisi runtime.
- Pemeliharaan: Kode dengan struktur kontrol yang baik lebih mudah dipahami dan digunakan.

Untuk menulis program yang efektif dan efisien dalam Go, pengembang harus memahami struktur kontrol karena dirancang untuk menjadi mudah dan efektif.

## II. GUIDED

### 1. Source Code

```
package main

import "fmt"

func main() {

    var nama string = "Didik Weka Pratama"

    var umur int = 20

    var tinggi float64 = 175.5

    var isSunny bool = false

    var inisial rune = 'A'


    fmt.Println("Nama:", nama)

    fmt.Println("Umur:", umur)

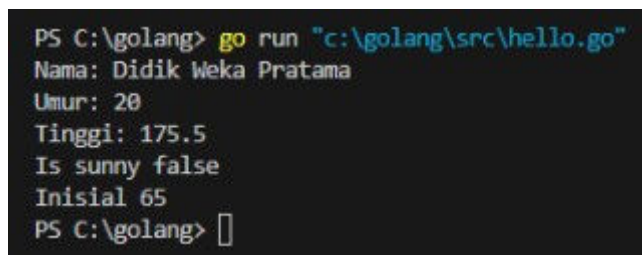
    fmt.Println("Tinggi:", tinggi)

    fmt.Println("Is sunny", isSunny)

    fmt.Println("Inisial", inisial)


}
```

### Screenshot Output



```
PS C:\golang> go run "c:\golang\src\hello.go"
Nama: Didik Weka Pratama
Umur: 20
Tinggi: 175.5
Is sunny false
Inisial 65
PS C:\golang> 
```

## Deskripsi Program

Program Go yang diberikan mendefinisikan paket main dan termasuk fungsi main, yang merupakan titik masuk dari program. Di dalam fungsi main, beberapa variabel dideklarasikan dengan tipe data yang berbeda dan diinisialisasi dengan nilai-nilai tertentu:

-*nama* adalah variabel string yang diinisialisasi dengan nilai "Didik Weka Pratama".

-*umur* adalah variabel integer yang diinisialisasi dengan nilai 20.

-*tinggi* adalah variabel float64 yang diinisialisasi dengan nilai 175.5.

-*isSunny* adalah variabel boolean yang diinisialisasi dengan nilai false.

-*inisial* adalah variabel rune (alias untuk int32) yang diinisialisasi dengan nilai 'A'.

Program kemudian menggunakan fungsi `fmt.Println` untuk mencetak deskripsi dan nilai dari setiap variabel ke output standar.

## 2.Source Code

```
package main

import "fmt"

func main() {
    //Tahun kabisat

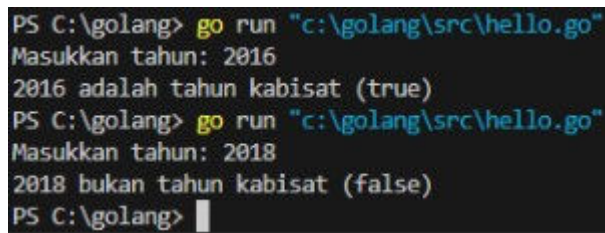
    var year int

    fmt.Print("Masukkan tahun: ")

    fmt.Scanf("%d", &year)

    if (year%400 == 0) || (year%4 == 0 && year%100 != 0) {
        fmt.Println(year, "adalah tahun kabisat (true)")
    } else {
        fmt.Println(year, "bukan tahun kabisat (false)")
    }
}
```

## SCREENSHOOT OUTPUT



```
PS C:\golang> go run "c:\golang\src\hello.go"
Masukkan tahun: 2016
2016 adalah tahun kabisat (true)
PS C:\golang> go run "c:\golang\src\hello.go"
Masukkan tahun: 2018
2018 bukan tahun kabisat (false)
PS C:\golang> 
```

## Deskripsi program

Program Go yang diberikan mendefinisikan paket **main** dan termasuk fungsi **main**, yang merupakan titik masuk dari program. Berikut adalah deskripsi langkah-langkah yang dilakukan oleh program:

1. Mendeklarasikan variabel **year** dengan tipe data **int**.
2. Menampilkan pesan "Masukkan tahun: " ke layar, meminta pengguna untuk memasukkan sebuah tahun.
3. Membaca input dari pengguna menggunakan **fmt.Scanf** untuk memasukkan nilai tahun ke dalam variable **year**.
4. Memeriksa apakah tahun yang dimasukkan adalah tahun kabisat menggunakan kondisi logika:
  - o Jika year habis dibagi 400 atau year habis dibagi 4 tetapi tidak habis dibagi 100, maka tahun tersebut adalah tahun kabisat.
5. Jika kondisi tahun kabisat terpenuhi, program akan mencetak pesan "<year> adalah tahun kabisat (true)".
6. Jika kondisi tahun kabisat tidak terpenuhi, program akan mencetak pesan "<year> bukan tahun kabisat (false)".

Program ini berfungsi sebagai alat sederhana untuk memeriksa apakah suatu tahun adalah tahun kabisat berdasarkan aturan umum:

- Tahun yang habis dibagi 4 adalah tahun kabisat, kecuali jika tahun tersebut juga habis dibagi 100.
- Namun, jika tahun tersebut habis dibagi 400, maka tetap dianggap sebagai tahun kabisat.

### 3. SOURCE CODE

```
package main

import "fmt"

func main() {

    //Temperatur

    var temperaturCelsius float64

    fmt.Print("Temperatur Celsius: ")

    fmt.Scanln(&temperaturCelsius)


    var temperaturFahrenheit float64 = (temperaturCelsius * 9 / 5) + 32

    var temperaturReamur float64 = temperaturCelsius * 4 / 5

    var temperaturKelvin float64 = temperaturCelsius + 273.15

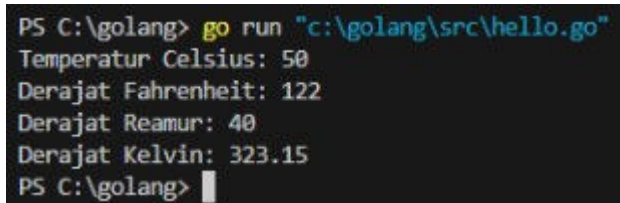

    fmt.Printf("Derajat Fahrenheit: %.0f\n", temperaturFahrenheit)

    fmt.Printf("Derajat Reamur: %.0f\n", temperaturReamur)

    fmt.Printf("Derajat Kelvin: %.2f\n", temperaturKelvin)


}
```

### SCREENSHOOT OUTPUT



```
PS C:\golang> go run "c:\golang\src\hello.go"
Temperatur Celsius: 50
Derajat Fahrenheit: 122
Derajat Reamur: 40
Derajat Kelvin: 323.15
PS C:\golang>
```

### Deskripsi program



Program Go yang diberikan mendefinisikan paket **main** dan termasuk fungsi **main**, yang merupakan titik masuk dari program. Berikut adalah deskripsi langkah-langkah yang dilakukan oleh program:

1. Mendeklarasikan variabel **temperaturCelsius** dengan tipe data **float64**.
2. Menampilkan pesan "Temperatur Celsius: " ke layar, meminta pengguna untuk memasukkan suhu dalam derajat Celsius.
3. Membaca input dari pengguna menggunakan **fmt.Scanln** untuk memasukkan nilai suhu ke dalam variabel **temperaturCelsius**.
4. Menghitung suhu dalam derajat Fahrenheit dengan rumus **(temperaturCelsius \* 9 / 5) + 32** dan menyimpannya dalam variabel **temperaturFahrenheit**.
5. Menghitung suhu dalam derajat Reamur dengan rumus **temperaturCelsius \* 4 / 5** dan menyimpannya dalam variabel **temperaturReamur**.
6. Menghitung suhu dalam derajat Kelvin dengan rumus **temperaturCelsius + 273.15** dan menyimpannya dalam variabel **temperaturKelvin**.
7. Mencetak hasil konversi suhu ke layar menggunakan **fmt.Printf** dengan format yang sesuai:
  - "Derajat Fahrenheit: %.0f\n" untuk menampilkan suhu dalam derajat Fahrenheit tanpa desimal.
  - "Derajat Reamur: %.0f\n" untuk menampilkan suhu dalam derajat Reamur tanpa desimal.
  - "Derajat Kelvin: %.2f\n" untuk menampilkan suhu dalam derajat Kelvin dengan dua angka desimal.

Program ini berfungsi sebagai alat sederhana untuk mengonversi suhu dari derajat Celsius ke derajat Fahrenheit, Reamur, dan Kelvin.

#### 4. SOURCE CODE

```
package main

import "fmt"

func main() {
    var (
        satu, dua, tiga string
        temp string
    )

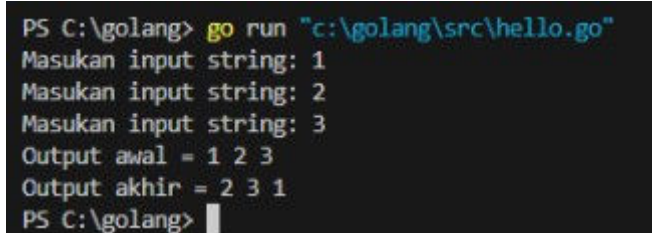
    fmt.Print("Masukan input string: ")
    fmt.Scanln(&satu)
    fmt.Print("Masukan input string: ")
    fmt.Scanln(&dua)
    fmt.Print("Masukan input string: ")
    fmt.Scanln(&tiga)

    fmt.Println("Output awal = " + satu + " " + dua + " " + tiga)

    temp = satu
    satu = dua
    dua = tiga
    tiga = temp

    fmt.Println("Output akhir = " + satu + " " + dua + " " + tiga)
}
```

## SCREENSHOOT OUTPUT



```
PS C:\golang> go run "c:\golang\src\hello.go"
Masukan input string: 1
Masukan input string: 2
Masukan input string: 3
Output awal = 1 2 3
Output akhir = 2 3 1
PS C:\golang> 
```

## Deskripsi Program

Program ini ditulis dalam bahasa pemrograman Go yang berfungsi untuk menerima tiga input string dari pengguna, kemudian menampilkan urutan awal string tersebut, melakukan pertukaran posisi, dan akhirnya menampilkan urutan string setelah ditukar. Program dimulai dengan mengimpor package `fmt` yang digunakan untuk menangani input dan output. Di dalam fungsi utama (`main`), tiga variabel string bernama `satu`, `dua`, dan `tiga` dideklarasikan untuk menyimpan input dari pengguna, serta satu variabel `temp` yang berfungsi sebagai variabel sementara dalam proses pertukaran nilai.

Program akan meminta pengguna memasukkan tiga string secara berurutan dan menyimpannya dalam variabel `satu`, `dua`, dan `tiga`. Setelah itu, program menampilkan urutan string awal berdasarkan input yang diberikan. Selanjutnya, proses pertukaran dilakukan, di mana nilai `satu` diisi dengan nilai `dua`, nilai `dua` diisi dengan nilai `tiga`, dan nilai `tiga` diisi dengan nilai awal `satu` yang disimpan dalam variabel `temp`. Terakhir, program menampilkan urutan string yang sudah diubah setelah proses pertukaran selesai. Dengan demikian, program ini menunjukkan bagaimana urutan input dapat dimanipulasi melalui pertukaran nilai variabel dengan algoritma swap sederhana.

### **III. UNGUIDED**

#### **2B.**

##### **No 1. Source Code**

```
package main

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

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

    scanner := bufio.NewScanner(os.Stdin)

    success := true

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

        scanner.Scan()

        colors := strings.Fields(scanner.Text())
```

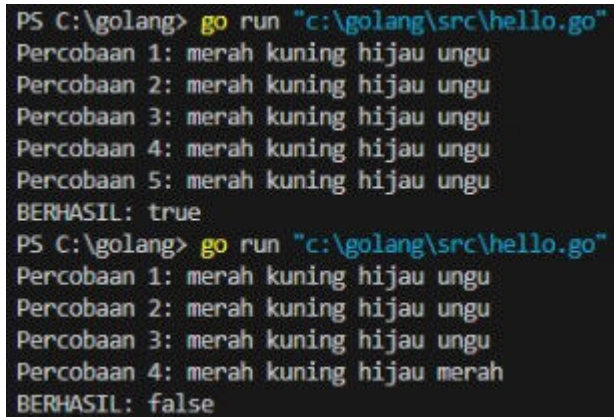
```
if len(colors) != len(expectedColors) {  
    success = false  
    break  
}
```

```
for j, color := range colors {  
    if color != expectedColors[j] {  
        success = false  
        break  
    }  
}
```

```
if !success {  
    break  
}  
}
```

```
fmt.Printf("BERHASIL: %v\n", success)  
}
```

## Screenshoot Output



```
PS C:\golang> go run "c:\golang\src\hello.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:\golang> go run "c:\golang\src\hello.go"
Percobaan 1: merah kuning hijau ungu
Percobaan 2: merah kuning hijau ungu
Percobaan 3: merah kuning hijau ungu
Percobaan 4: merah kuning hijau merah
BERHASIL: false
```

## Deskripsi Program

### 1. Import Package:

- bufio: Untuk membaca input dari pengguna.
- fmt: Untuk mencetak output.
- os: Untuk menangani input/output standar.
- strings: Untuk memanipulasi string, seperti memisahkan input menjadi slice string.

### 2. Deklarasi Variabel:

- expectedColors: Slice dari string yang berisi urutan warna yang diharapkan.

### 3. Membaca Input:

- Menggunakan bufio.NewScanner untuk membaca input dari pengguna.

### 4. Memeriksa Percobaan:

- Loop selama lima kali untuk setiap percobaan.
- Memeriksa apakah setiap urutan warna sesuai dengan yang diharapkan.
- Jika ditemukan perbedaan pada urutan warna, ubah success menjadi false dan keluar dari loop.

## 5. Menampilkan Hasil:

- Mencetak BERHASIL: true jika semua percobaan sesuai, atau BERHASIL: false jika ada percobaan yang tidak sesuai.

## No 2. Source Code

```
package main
```

```
import (  
    "fmt"  
    "strings"  
)
```

```
func main() {  
    var n int  
  
    var bunga, pita string  
  
    var count int  
  
    fmt.Print("Masukkan jumlah bunga (N): ")  
  
    fmt.Scan(&n)  
  
    if n == 0 {  
        fmt.Println("Pita: ")  
  
        fmt.Println("Bunga: 0")  
  
        return  
    }  
}
```

```
for count = 1; count <= n; count++ {  
    fmt.Printf("Bunga %d: ", count)  
    fmt.Scan(&bunga)  
  
    if strings.ToUpper(bunga) == "SELESAI" {  
        count--  
        break  
    }  
  
    if count > 1 {  
        pita += " - "  
    }  
    pita += bunga  
}  
  
fmt.Println("Pita:", pita)  
fmt.Printf("Bunga: %d\n", count)  
}
```

## Screenshot Output



```

PS C:\golang> go run "c:\golang\src\hello.go"
Masukkan jumlah bunga (N): 3
Bunga 1: selesai
Pita:
Bunga: 0
PS C:\golang> go run "c:\golang\src\hello.go"
Masukkan jumlah bunga (N): 4
Bunga 1: kertas
Bunga 2: mawar
Bunga 3: tulip
Bunga 4: selesai
Pita: kertas - mawar - tulip
Bunga: 3
PS C:\golang>

```

### Deskripsi Program

Program ini merupakan aplikasi untuk memasukkan nama bunga dan menampilkan rangkaian bunga dalam bentuk pita. Pengguna diminta untuk menentukan jumlah bunga yang ingin dimasukkan terlebih dahulu. Kemudian, program akan meminta input nama bunga satu per satu, yang akan dirangkai dalam format pita dengan pemisah berupa " - " di antara setiap nama bunga. Jika pengguna mengetik "SELESAI", proses input bunga dihentikan sebelum mencapai jumlah yang diminta. Pada akhir program, pita yang berisi rangkaian bunga ditampilkan bersama jumlah bunga yang berhasil diinput.

### No 3. Source Code

```

package main

import (
    "fmt"
    "math"
)

func main() {
    var beratKantong1, beratKantong2, totalBerat float64

```

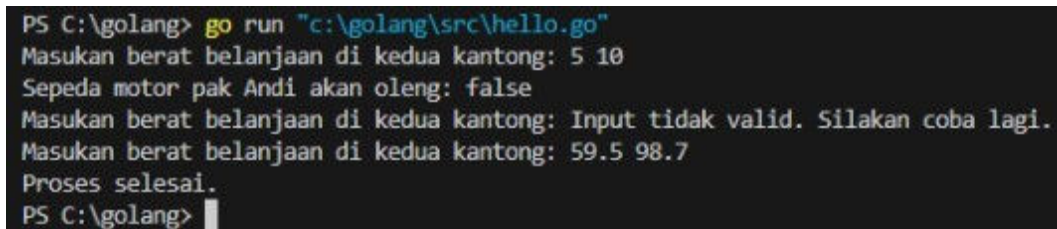
```
for {  
    fmt.Print("Masukan berat belanjaan di kedua kantong: ")  
    _, err := fmt.Scanf("%f %f", &beratKantong1, &beratKantong2)  
  
    if err != nil {  
        fmt.Println("Input tidak valid. Silakan coba lagi.")  
        continue  
    }  
  
    if beratKantong1 < 0 || beratKantong2 < 0 {  
        fmt.Println("Proses selesai.")  
        break  
    }  
  
    totalBerat = beratKantong1 + beratKantong2  
  
    if totalBerat > 150 {  
        fmt.Println("Proses selesai.")  
        break  
    }  
  
    selisih := math.Abs(beratKantong1 - beratKantong2)  
  
    if selisih >= 9 {  
        fmt.Println("Sepeda motor pak Andi akan oleng: true")  
    }  
}
```

```

    } else {
        fmt.Println("Sepeda motor pak Andi akan oleng: false")
    }
}
}
}

```

### Screenshoot Output



```

PS C:\golang> go run "c:\golang\src\hello.go"
Masukan berat belanjaan di kedua kantong: 5 10
Sepeda motor pak Andi akan oleng: false
Masukan berat belanjaan di kedua kantong: Input tidak valid. Silakan coba lagi.
Masukan berat belanjaan di kedua kantong: 59.5 98.7
Proses selesai.
PS C:\golang>

```

### Deskripsi Program

Program ini berfungsi untuk mengecek kestabilan sepeda motor Pak Andi berdasarkan berat belanjaan di dua kantong. Pengguna diminta memasukkan berat belanjaan pada dua kantong secara berulang. Program akan menghitung total berat belanjaan dan selisih berat antara kedua kantong. Jika total berat lebih dari 150 kg atau salah satu kantong memiliki berat negatif, program akan berhenti dengan menampilkan pesan "Proses selesai". Selain itu, jika selisih berat antara kantong lebih dari atau sama dengan 9 kg, program akan mengeluarkan peringatan bahwa sepeda motor akan oleng (true), jika tidak, motor dinyatakan stabil (false).

### No 4. Source Code

```
package main
```

```

import (
    "fmt"
    "math"
)

```

```

func f(k int) float64 {
    numerator := math.Pow(float64(4*k+2), 2)
    denominator := float64((4*k + 1) * (4*k + 3))
    return numerator / denominator
}

```

```

func sqrt2Approximation(K int) float64 {
    product := 1.0
    for k := 0; k <= K; k++ {
        product *= f(k)
    }
    return product
}

```

```

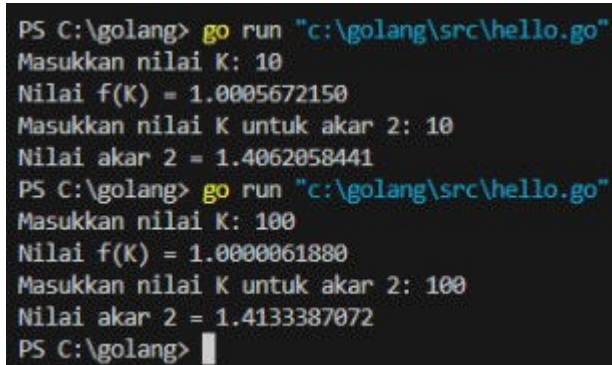
func main() {
    var K int

    fmt.Print("Masukkan nilai K: ")
    fmt.Scan(&K)
    fmt.Printf("Nilai f(K) = %.10f\n", f(K))

    fmt.Print("Masukkan nilai K untuk akar 2: ")
    fmt.Scan(&K)
    fmt.Printf("Nilai akar 2 = %.10f\n", sqrt2Approximation(K))
}

```

## Screenshoot Output



```
PS C:\golang> go run "c:\golang\src\hello.go"
Masukkan nilai K: 10
Nilai f(K) = 1.0005672150
Masukkan nilai K untuk akar 2: 10
Nilai akar 2 = 1.4062058441
PS C:\golang> go run "c:\golang\src\hello.go"
Masukkan nilai K: 100
Nilai f(K) = 1.0000061880
Masukkan nilai K untuk akar 2: 100
Nilai akar 2 = 1.4133387072
PS C:\golang> 
```

## Deskripsi Program

Program ini digunakan untuk menghitung nilai fungsi tertentu dan menghampiri nilai akar 2 menggunakan metode perkalian dengan jumlah iterasi yang ditentukan oleh pengguna (nilai K). Fungsi  $f(k)$  dihitung berdasarkan rumus yang melibatkan perhitungan kuadrat dan pembagian. Program pertama-tama meminta pengguna memasukkan nilai K untuk menghitung dan menampilkan hasil dari fungsi  $f(K)$ . Selanjutnya, pengguna diminta memasukkan nilai K lain untuk menghitung pendekatan nilai akar 2 dengan mengalikan hasil dari fungsi  $f(k)$  dari 0 hingga K. Program kemudian menampilkan hasil pendekatan akar 2 dengan presisi hingga 10 desimal.

## 2C

### No 1. Source Code

```
package main
```

```
import "fmt"
```

```
func main() {
```

```
    var berat, kg, gr, biayaKirim, tambahanBiaya, totalBiaya int
```

```
    fmt.Print("Berat parsel (gram): ")
```

```
    fmt.Scan(&berat)
```

```
    kg = berat / 1000
```

```
    gr = berat % 1000
```

```
    if kg > 10 {
```

```
        tambahanBiaya = 0
```

```
    } else if gr >= 500 {
```

```
        tambahanBiaya = gr * 5
```

```
    } else {
```

```
        tambahanBiaya = gr * 15
```

```
    }
```

```
    biayaKirim = kg * 10000
```

```
    totalBiaya = biayaKirim + tambahanBiaya
```

```

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

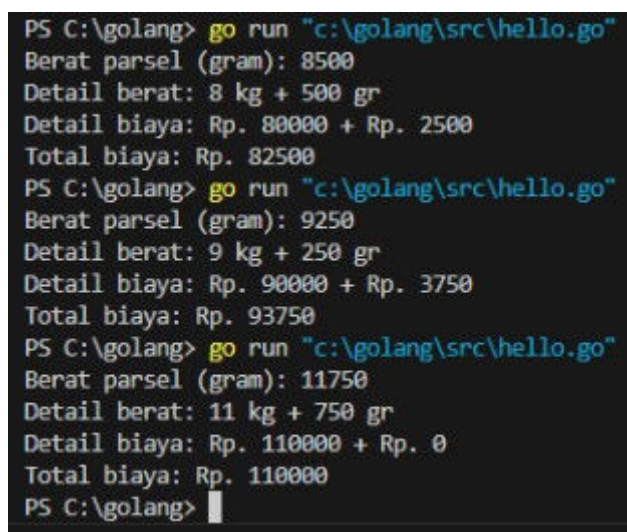
    fmt.Printf("Detail biaya: Rp. %d + Rp. %d\n", biayaKirim, tambahanBiaya)

    fmt.Printf("Total biaya: Rp. %d\n", totalBiaya)

}

```

### Screenshoot Output



```

PS C:\golang> go run "c:\golang\src\hello.go"
Berat parcel (gram): 8500
Detail berat: 8 kg + 500 gr
Detail biaya: Rp. 80000 + Rp. 2500
Total biaya: Rp. 82500
PS C:\golang> go run "c:\golang\src\hello.go"
Berat parcel (gram): 9250
Detail berat: 9 kg + 250 gr
Detail biaya: Rp. 90000 + Rp. 3750
Total biaya: Rp. 93750
PS C:\golang> go run "c:\golang\src\hello.go"
Berat parcel (gram): 11750
Detail berat: 11 kg + 750 gr
Detail biaya: Rp. 110000 + Rp. 0
Total biaya: Rp. 110000
PS C:\golang>

```

### Deskripsi Program

Program ini menghitung total biaya pengiriman parcel berdasarkan beratnya dalam gram. Pengguna diminta untuk memasukkan berat parcel, yang kemudian akan diubah menjadi kilogram (kg) dan gram (gr) sisa. Biaya pengiriman dasar dihitung sebesar Rp 10.000 per kilogram. Untuk sisa berat di bawah 1 kg, biaya tambahan dihitung berdasarkan apakah sisa berat lebih dari 500 gram (dikalikan 5) atau kurang (dikalikan 15). Program kemudian menampilkan rincian berat parcel dalam kg dan gr, biaya pengiriman berdasarkan kg, biaya tambahan berdasarkan sisa gram, serta total biaya pengiriman.

### No 2. Source Code

```
package main
```

```
import "fmt"
```

```
func main() {  
    var nam float64  
  
    var nmk string  
  
    fmt.Print("Nilai akhir mata kuliah: ")  
  
    fmt.Scanln(&nam)  
  
    if nam > 80 {  
        nmk = "A"  
    } else if nam > 72.5 {  
        nmk = "AB"  
    } else if nam > 65 {  
        nmk = "B"  
    } else if nam > 57.5 {  
        nmk = "BC"  
    } else if nam > 50 {  
        nmk = "C"  
    } else if nam > 40 {  
        nmk = "D"  
    } else {  
        nmk = "E"  
    }  
  
    fmt.Println("Nilai mata kuliah: ", nmk)  
}
```

### **Screenshoot Output**



```

PS C:\golang> go run "c:\golang\src\hello.go"
Nilai akhir mata kuliah: 80.1
Nilai mata kuliah: A
PS C:\golang> go run "c:\golang\src\hello.go"
Nilai akhir mata kuliah: 93.5
Nilai mata kuliah: A
PS C:\golang> go run "c:\golang\src\hello.go"
Nilai akhir mata kuliah: 70.6
Nilai mata kuliah: B
PS C:\golang> go run "c:\golang\src\hello.go"
Nilai akhir mata kuliah: 49.5
Nilai mata kuliah: D
PS C:\golang>

```

### Deskripsi Program

Program ini berfungsi untuk menentukan nilai mata kuliah berdasarkan nilai akhir yang dimasukkan oleh pengguna. Setelah pengguna memasukkan nilai akhir, program akan mengklasifikasikan nilai tersebut ke dalam kategori huruf (A, AB, B, BC, C, D, atau E) menggunakan beberapa kondisi. Jika nilai akhir lebih dari 80, pengguna akan mendapatkan nilai "A", dan seterusnya mengikuti interval nilai yang sudah ditentukan. Program kemudian menampilkan nilai huruf yang sesuai dengan nilai akhir tersebut.

### No 3. Source Code

```
package main
```

```
import "fmt"
```

```

func findFactors(b int) []int {
    var factors []int
    for i := 1; i <= b; i++ {
        if b%i == 0 {
            factors = append(factors, i)
        }
    }
}

```

```

    return factors
}

func isPrime(b int) bool {
    factors := findFactors(b)
    return len(factors) == 2
}

func main() {
    var b int

    fmt.Print("Bilangan: ")
    fmt.Scan(&b)

    factors := findFactors(b)
    fmt.Print("Faktor: ")
    for _, factor := range factors {
        fmt.Printf("%d ", factor)
    }
    fmt.Println()

    primeStatus := isPrime(b)
    fmt.Printf("Prima: %t\n", primeStatus)
}

```

### **Screenshoot Output**

```
PS C:\golang> go run "c:\golang\src\hello.go"
Bilangan: 12
Faktor: 1 2 3 4 6 12
Prima: false
PS C:\golang> go run "c:\golang\src\hello.go"
Bilangan: 7
Faktor: 1 7
Prima: true
PS C:\golang> 
```

### Deskripsi Program

Program ini digunakan untuk mencari faktor-faktor dari sebuah bilangan dan menentukan apakah bilangan tersebut merupakan bilangan prima. Pertama, pengguna diminta memasukkan sebuah bilangan. Program kemudian menghitung dan menampilkan semua faktor dari bilangan tersebut dengan memeriksa angka-angka yang dapat membagi habis bilangan tersebut. Selain itu, program juga mengecek apakah bilangan tersebut merupakan bilangan prima dengan memverifikasi apakah jumlah faktornya hanya dua, yaitu 1 dan bilangan itu sendiri. Hasilnya akan ditampilkan dengan format "Prima: true" jika bilangan prima, atau "Prima: false" jika bukan.