

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



Oleh:

HENDWI SAPUTRA

2311102218

IF-11-02

**S1 TEKNIK INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2024**

I. DASAR TEORI

2.1 Struktur Program Go

Dalam kerangka program yang ditulis dalam bahasa pemrograman Go, program utama selalu mempunyai dua komponen berikut:

- Package main merupakan penanda bahwa file ini berisi program utama.
- func main(berisi kode utama dari sebuah program Go.
Komentar, bukan bagian dari kode program, dan dapat ditulis di mana saja di dalam program:
- Satu baris teks yang diawali dengan garis miring ganda ("I") s.d. akhir baris, atau. • Beberapa baris teks yang dimulai dengan pasangan karakter "" dan diakhiri dengan ""

```
package main
import "fmt"
func main() {
    var greetings = "Selamat datang di dunia DAP"
    var a, b int

    fmt.Println(greetings)
    fmt.Scanln(&a, &b)
    fmt.Printf("%v + %v = %v\n", a, b, a+b)
}
```

Koding, Kompilasi, dan Eksekusi Go

Koding

- Tidak berbeda dengan penulisan program sumber dalam bahasa lain, program Go harus dibuat menggunakan penyunting teks dan disimpan dalam format teks, bukan dalam format dokumen (doc, docx, atau lainnya)
- Setiap program go disimpan dalam file teks dengan ekstensi t go, dengan nama bebas. Sebaiknya nama file adalah nama untuk program tersebut,
- Setelah satu program lengkap Go disimpan dalam satu folder tersendiri. Nama folder merupakan nama program tersebut, Karena

itu secara prinsip, satu program Go dapat dipecah dalam beberapa file dengan ekstensi *.go selama disimpan dalam folder yang sama:

Kompilasi

Beberapa bahasa pemrograman dirancang untuk diimplementasikan sebagai interpreter dan lainnya sebagai kompilator. Interpreter akan membaca setiap baris instruksi dan kemudian langsung mengeksekusinya, dengan hanya sedikit pemeriksaan apakah penulisan keseluruhan program sudah benar atau belum. Kompilator akan memeriksa keseluruhan program sumber dan kemudian mengubahnya menjadi program eksekutabel, sehingga konsistensi penulisan (seperti penggunaan tipe data) sudah diperiksa sebelum eksekusi. Selain itu karena program dibuat menjadi eksekutabel lebih dahulu, proses optimasi dapat dilakukan sehingga program menjadi sangat efisien. Catatan : Semua proses terkait bahasa Go dilakukan melalui utilitas Go. Beberapa opsi dengan utilitas Go : Go build : mengkompilasi program sumber yang ada dalam folder menjadi sebuah program

- Go build file.go : mengkompilasi program sumber file.go saja.
- Go fmt : membaca semua program sumber dalam folder dan mereformat penulisannya agar sesuai dengan standar penulisan program sumber Go.
- Go clean : membersihkan file-file dalam folder sehingga tersisa program sumber saja

Variabel

Variabel adalah nama dari suatu lokasi di memori, yang data dengan tipe tertentu dapat disimpan. Nama variabel dimulai dengan huruf dan dapat diikuti dengan sejumlah huruf, angka, atau garisbawah

Notasi tipe dasar	Tipe dalam Go	Keterangan
integer	int int8 int32 //rune int64 uint uint8 //byte uint32 uint64	bergantung platform 8 bit: -128..127 32 bit: -10 ⁹ ..10 ⁹ 64 bit: -10 ¹⁹ ..10 ¹⁹ bergantung platform 0..255 0..4294967295 0..(2 ⁶⁴ -1)
real	float32 float64	32bit: -3.4E+38 .. 3.4E+38 64bit: -1.7E+308 .. 1.7E+308
boolean (atau logikal)	bool	false dan true
karakter	byte	tabel tabel
string	string	

- Tipe data yang umum tersedia adalah integer, real, boolean, karakter, dan string. Lihat tabel berikut ini untuk variasi tipe data yang disediakan dalam bahasa Go.
- Nilai data yang tersimpan dalam variabel dapat diperoleh dengan menyebutkan langsung nama variabelnya. Contoh: Menyebutkan nama found akan mengambil nilai tersimpan dalam memori untuk variabel found, pastinya.
- Informasi alamat atau lokasi dari variabel dapat diperoleh dengan menambahkan prefiks & di depan nama variabel tersebut. Contoh: &found akan mendapatkan alamat memori untuk menyimpan data pada found.
- Jika variabel berisi alamat memori, prefiks * pada variabel tersebut akan memberikan nilai yang tersimpan dalam memori yang lokasinya disimpan dalam variabel tersebut.

Struktur Kontrol Perulangan

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

```

1
2
3
4 }
5 for kondisi {
6     // .. ulangi kode di sini selama kondisi terpenuhi
7     // .. sama seperti "for ; kondisi; {"
8 }
9 for {
10 // .. tanpa kondisi, berarti loop tanpa henti (perlu if-break)
11 }
12 for ndx,
13
14
15 }

```

Bentuk While-Loop Bentuk while-loop memastikan setiap kali memasuki loop, ada kondisi yang harus terpenuhi (benar/true). Inijuga berarti saat keluar dari loop, maka nilai kondisi tersebut pasti salah/false!

	Notasi algoritma	Penulisan dalam bahasa Go
1	e <-	e :=
2	x <-	x :=
3	y <- 0.0	y := 0.0
4	y1 <- x	y1 := x
5	while y1-y > e or y1-y < -e do	for -y > e -y < -e {
6	y <- y1	y = y1

Bentuk Repeat-Until Bentuk repeat-until di perulangan dilakukan terus menerus sampai kondisi keluar terpenuhi. Artinya selama kondisi belum terpenuhi (salah/false) maka perulangan akan terus dilakukan. Pada saat keluar dari loop maka nilai kondisi pasti benar/true!

	Notasi Algoritma	Penulisan dalam bahasa Go
1	repeat	for selesai:=false; !selesai; {
2	.. kode yang diulang	.. kode yang diulang
3	until (kondisi)	selesai = kondisi
4		}
5		for selesai:=false; !selesai;
6		selesai=kondisi {
7		..kode yang diulang
8		}
9		

Struktur Kontrol Percabangan

Untuk analisa kasus, bahasa Go mendukung dua bentuk percabangan, yaitu if-else dan switch-case. 1) Bentuk If-Else 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

	Notasi algoritma	Penulisan dalam bahasa Go
1	if (kondisi) then	if kondisi {
2	.. kode untuk kondisi true	.. kode untuk kondisi true
3	endif	}
4	if (kondisi) then	if kondisi {
5	kode untuk kondisi true	.. kode untuk kondisi true
6	else	else {
7	.. kode untuk kondisi false	kode untuk
8	endif	}
9		if kondisi_1 {
10		.. kode untuk kondisi_1 true
11		} else if kondisi_2 {
12		.. kode untuk kondisi
13		.. dst. dst.
14	else	} else {
15	.. kode jika semua kondisi	kode jika semua kondisi
16	di atas false	di atas false
17	endif	}

Bentuk Switch-Case Dalam bahasa Co ada dua variasi bentuk switchcase. 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

	Notasi algoritma	Penulisan dalam bahasa Go
1	depend on ekspresi	switch ekspresi {
2	nilai_1:	case nilai_1:
3	.. kode jika ekspresi bernilai_1	.. kode jika ekspresi bernilai_1
4	nilai_2:	case nilai_2:
5	.. kode jika ekspresi bernilai_2	.. kode jika ekspresi bernilai_2
6	.. dst. dst.	.. dst. dst.
7	}	default:
8		..kode jika tidak ada nilai
9		..yang cocok dengan ekspresi
10		}
11	depend on (daftar variabel)	switch {
12	kondisi_1:	case kondisi_1:
13	.. kode jika ekspresi_1 true	.. kode jika ekspresi_1 true
14	kondisi_2:	case kondisi_2:
15	.. kode jika ekspresi_2 true	.. kode jika ekspresi _2 true
16	.. dst. dst.	.. dst. dst.
17	}	default:
18		..jika tidak ada ekspresi
19		..yang bernilai true
20		}

II. GUIDED

Source Code + Screenshot hasil program beserta penjelasan

GUIDED I

Source Code

```
package main
```

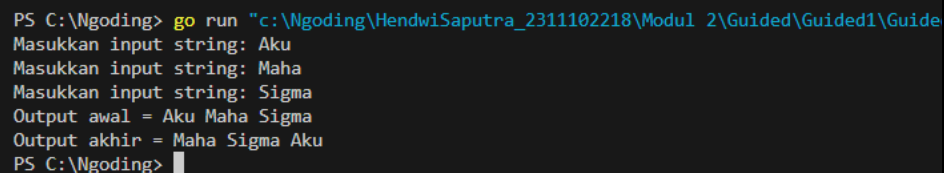
```

import "fmt"

func main() {
    var satu, dua, tiga string
    var temp string
    fmt.Print("Masukkan input string: ")
    fmt.Scanln(&satu)
    fmt.Print("Masukkan input string: ")
    fmt.Scanln(&dua)
    fmt.Print("Masukkan 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)
}

```

Screenshot



```

PS C:\Ngoding> go run "c:\Ngoding\HendwiSaputra_2311102218\Modul 2\Guided\Guided1\Guide
Masukkan input string: Aku
Masukkan input string: Maha
Masukkan input string: Sigma
Output awal = Aku Maha Sigma
Output akhir = Maha Sigma Aku
PS C:\Ngoding>

```

Deskripsi:

Program ini bertujuan untuk menukar nilai tiga string yang dimasukkan oleh pengguna. Pada awalnya, program meminta pengguna untuk memasukkan tiga string, kemudian mencetak string-string tersebut dalam urutan awalnya. Setelah itu, program menukar nilai-nilai string tersebut menggunakan variabel sementara temp, sehingga urutan string berubah. Akhirnya, program mencetak urutan baru dari string-string tersebut.

GUIDED II

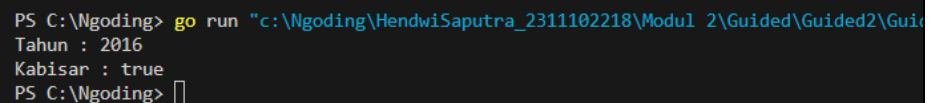
Source Code

```
package main

import "fmt"

func main() {
    var tahun int
    fmt.Print("Tahun : ")
    fmt.Scan(&tahun)
    fmt.Printf("Kabisar : %t \n", (tahun%4 == 0 &&
(tahun%100 != 0 || tahun%400 == 0)))
}
```

Screenshot



```
PS C:\Ngoding> go run "c:\Ngoding\HendwiSaputra_2311102218\Modul 2\Guided\Guided2\Guided2.go"
Tahun : 2016
Kabisar : true
PS C:\Ngoding>
```

Deskripsi:

Program ini adalah aplikasi sederhana dalam bahasa pemrograman Go yang menentukan apakah suatu tahun adalah tahun kabisat atau bukan. Pengguna diminta untuk memasukkan sebuah tahun. Kemudian, program menghitung apakah tahun tersebut memenuhi syarat sebagai tahun kabisat, yaitu tahun yang dapat dibagi habis oleh 4, tetapi tidak oleh 100, kecuali jika tahun tersebut juga dapat dibagi habis oleh 400. Hasil perhitungan ini ditampilkan dalam format boolean (true atau false). Output akan menunjukkan true jika tahun tersebut adalah tahun kabisat, dan false jika tidak.

GUIDED III

Source Code

```
package main

import (
    "fmt"
    "math"
```



```

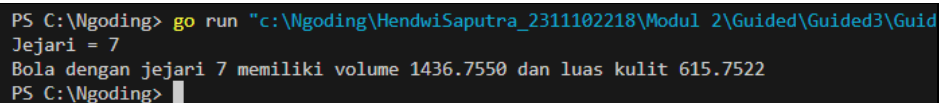
)

func main() {

    var jari, volume, luasKulit float64
    fmt.Print("Jejari = ")
    fmt.Scan(&jari)
    volume = math.Pi * (4.0 / 3.0) * math.Pow(jari, 3)
    luasKulit = 4 * math.Pi * math.Pow(jari, 2)
    fmt.Printf("Bola dengan jejari %v memiliki volume %.4f
dan luas kulit %.4f \n", jari, volume, luasKulit)
}

```

Screenshot



```

PS C:\Ngoding> go run "c:\Ngoding\HendwiSaputra_2311102218\Modul 2\Guided\Guided3\Guided3.go"
Jejari = 7
Bola dengan jejari 7 memiliki volume 1436.7550 dan luas kulit 615.7522
PS C:\Ngoding>

```

Deskripsi:

Program ini adalah aplikasi sederhana dalam bahasa pemrograman Go yang menghitung volume dan luas permukaan bola berdasarkan jejari yang diberikan pengguna. Program meminta input jejari bola, kemudian menggunakan konstanta Pi dari paket math untuk menghitung volume dengan dan luas permukaan dengan rumus. Hasil perhitungan tersebut kemudian ditampilkan dengan presisi empat desimal.

III. UNGUIDED

UNGUIDED I

Source Code

```
package main

import "fmt"

func main() {
    warnaIdeal := []string{"merah", "kuning", "hijau",
"ungu"}
    berhasil := 0

    for i := 1; i < 5; i++ {
        fmt.Print("Percobaan ", i, " : ")
        var warna [4]string
        fmt.Scan(&warna[0], &warna[1], &warna[2],
&warna[3])

        isSesuai := true
        for j := 0; j < len(warna); j++ {
            if warna[j] != warnaIdeal[j] {
                isSesuai = false
                break
            }
        }
        if isSesuai {
            berhasil++
        }
    }

    if berhasil == 4 {
        fmt.Println("BERHASIL: true")
    } else {
        fmt.Println("BERHASIL: false")
    }
}
```

Screenshot

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Ngoding> go run "c:\Ngoding\HendwiSaputra_2311102218\Modul 1\unguided\Unguided1\unguid
Percobaan 1 : merah kuning hijau ungu
Percobaan 2 : merah kuning hijau ungu
Percobaan 3 : merah kuning hijau ungu
Percobaan 4 : merah kuning hijau ungu
BERHASIL: true
PS C:\Ngoding> go run "c:\Ngoding\HendwiSaputra_2311102218\Modul 1\unguided\Unguided1\unguid
Percobaan 1 : merah kuning hijau ungu
Percobaan 2 : merah kuning ungu hijau
Percobaan 3 : merah kuning ungu hijau
Percobaan 4 : merah kuning hijau ungu
Percobaan 5 : merah kuning hijau ungu
BERHASIL: false
```

Deskripsi:

Ini adalah sebuah program yang menerima input berupa warna dari 4 gelas reaksi sebanyak 5 kali percobaan. Kemudian program akan menampilkan **true** apabila urutan warna sesuai dengan informasi yang diberikan pada line sebelumnya, dan **false** untuk urutan warna lainnya. Program akan meminta pengguna untuk memasukkan empat warna sebanyak lima kali. Setiap inputan akan dibandingkan dengan urutan warna ideal. Jika semua warna pada suatu percobaan sesuai dengan urutannya, maka percobaan tersebut dianggap berhasil. Di akhir program, akan ditampilkan hasil akhir berupa pernyataan "BERHASIL: true" jika semua percobaan berhasil, atau "BERHASIL: false" jika ada satu atau lebih percobaan yang gagal.

UNGUIDED II

Source Code

```
package main

import "fmt"

var (
    N, jumlahBunga int
    namaBunga, pita string
)

func main() {

    fmt.Print("Inputkan jumlah bunga: ")
    fmt.Scanln(&N)

    if N <= 0 {
        fmt.Println("Tidak dapat membuat pita bunga.")
    } else {
        for i := 1; i <= N; i++ {
            fmt.Print("Bunga ", i, ": ")
            fmt.Scan(&namaBunga)

            if namaBunga == "Selesai" {
                break
            }
            pita += " " + namaBunga + " -"
            jumlahBunga++
        }

        fmt.Println("Pita:", pita)
        fmt.Print("Bunga: ", jumlahBunga)
    }
}
```

Screenshot

```
PS C:\Ngoding> go run "c:\Ngoding\HendwiSaputra_2311102218\Modul 1\unguided\Unguided2\unguided2.go"
Inputkan jumlah bunga: 4
Bunga 1: Kertas
Bunga 2: MAwar
Bunga 3: Tulip
Bunga 4: Selesai
Pita: Kertas - MAwar - Tulip -
Bunga: 3

PS C:\Ngoding> go run "c:\Ngoding\HendwiSaputra_2311102218\Modul 1\unguided\Unguided2\unguided2.go"
Inputkan jumlah bunga: 0
Tidak dapat membuat pita bunga.
Pita:
Bunga: 0
PS C:\Ngoding>

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS Code + v
PS C:\Ngoding> go run "c:\Ngoding\HendwiSaputra_2311102218\Modul 1\unguided\Unguided2\unguided2.go"
Inputkan jumlah bunga: 3
Bunga 1: Kertas
Bunga 2: MAwar
Bunga 3: Tulip
Pita: Kertas - MAwar - Tulip -
Bunga: 3
PS C:\Ngoding>

PS C:\Ngoding> go run "c:\Ngoding\HendwiSaputra_2311102218\Modul 1\unguided\Unguided2\unguided2.go"
Inputkan jumlah bunga: 1
Bunga 1: Selesai
Pita:
Bunga: 0
PS C:\Ngoding>
```

Deskripsi:

Program ini dirancang untuk membuat sebuah "pita" yang terdiri dari nama-nama bunga. Pengguna diminta untuk menginputkan jumlah bunga yang ingin dimasukkan ke dalam pita. Kemudian, program akan meminta pengguna untuk memasukkan nama setiap bunga satu per satu. Proses input akan terus berulang hingga pengguna memasukkan kata "Selesai". Setiap nama bunga yang dimasukkan akan ditambahkan ke dalam variabel `pita` dan jumlah bunga yang valid akan dihitung. Setelah semua input selesai, program akan menampilkan pita yang telah terbentuk beserta jumlah bunga yang berhasil dimasukkan.

UNGUIDED III

Source Code (Sebelum modifikasi)

```
package main

import "fmt"

var berat_kantong1, berat_kantong2 float32

func main() {
    for {
        fmt.Print("Masukan berat belanja di kedua kantong: ")
        fmt.Scan(&berat_kantong1, &berat_kantong2)

        if berat_kantong1 >= 9 || berat_kantong2 >= 9 {
            fmt.Println("Print selesai.")
        }
    }
}
```

Screenshot (Sebelum modifikasi)



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Ngoding> go run "c:\Ngoding\HendwiSaputra_2311102218\Modul 1\unguided\Unguided3\ungui
Masukan berat belanja di kedua kantong: 5.5 1.0
Masukan berat belanja di kedua kantong: 7.1 8.5
Masukan berat belanja di kedua kantong: 2 6
Masukan berat belanja di kedua kantong: 9 5.8
Print selesai.
```

Deskripsi:

Ini adalah program Pak Andi yang menerima input dua buah bilangan real bilangan 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

Source Code (Setelah modifikasi)

```

package main

import "fmt"

var berat_kantong1, berat_kantong2 float32

func main() {
    for {
        fmt.Print("Masukan berat belanjaan di kedua
kantong: ")
        fmt.Scan(&berat_kantong1, &berat_kantong2)

        if berat_kantong1 < 0 || berat_kantong2 < 0 ||
berat_kantong1+berat_kantong2 > 150 {
            fmt.Println("Print selesai.")
            break
        }

        selisih := berat_kantong1 - berat_kantong2
        if selisih < 0 {
            selisih = -selisih
        }
        fmt.Printf("Sepeda motor Pak Andi akan oleng:
%t\n", selisih >= 9)
    }
}

```

Screenshot (setelah modifikasi)

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Ngoding> go run "c:\Ngoding\HendwiSaputra_2311102218\Modul 1\unguided\Unguided3\unguided3.go"
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
Print selesai.
PS C:\Ngoding>

```

Deskripsi:

Program akan terus meminta pengguna memasukkan berat kedua kantong hingga salah satu kondisi berikut terpenuhi: berat salah satu kantong negatif (input tidak valid), total berat kedua kantong melebihi 150 (beban terlalu berat), atau selisih berat kedua kantong mencapai 9 atau lebih (sepeda motor berpotensi oleng). Setiap kali pengguna memasukkan data, program akan menghitung selisih berat dan memberikan peringatan jika sepeda motor berpotensi tidak stabil.

UNGUIDED IV**Source Code**

```
package main

import (
    "fmt"
)

func main() {
    var K int
    var F float64
```



```

    fmt.Print("Nilai K = ")
    fmt.Scanln(&K)

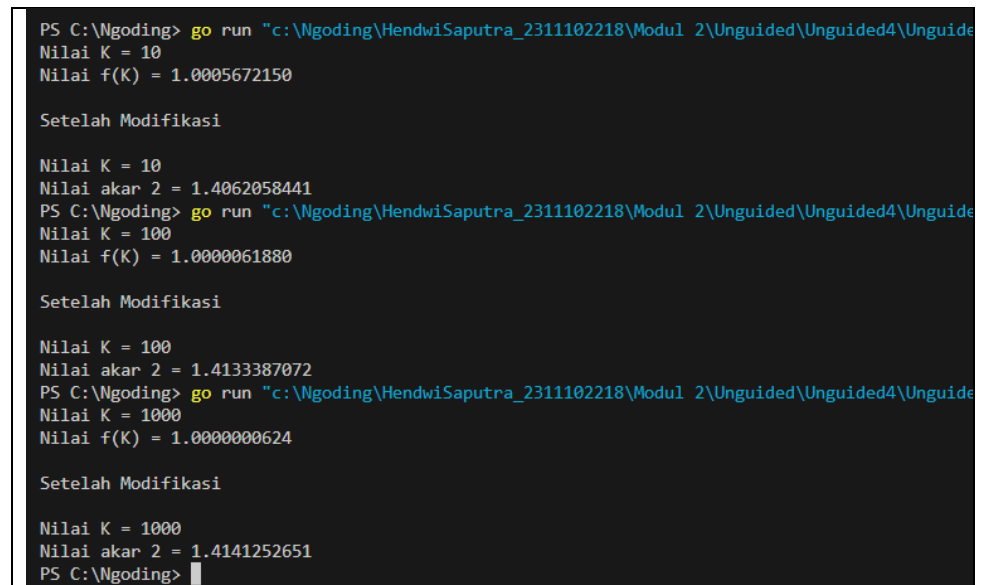
    F = float64((4*K+2)*(4*K+2)) /
float64((4*K+1)*(4*K+3))
    fmt.Printf("Nilai f(K) = %.10f\n", F)

    fmt.Print("\nSetelah Modifikasi\n\n")

    fmt.Print("Nilai K = ")
    fmt.Scanln(&K)
    var faktorAkar float64 = 1
    for i := 0; i <= K; i++ {
        faktorAkar *= float64((4*i+2)*(4*i+2)) /
float64((4*i+1)*(4*i+3))
    }
    fmt.Printf("Nilai akar 2 = %.10f\n", faktorAkar)
}

```

Screenshot



```

PS C:\Ngoding> go run "c:\Ngoding\HendwiSaputra_2311102218\Modul 2\Unguided\Unguided4\Unguide
Nilai K = 10
Nilai f(K) = 1.0005672150

Setelah Modifikasi

Nilai K = 10
Nilai akar 2 = 1.4062058441
PS C:\Ngoding> go run "c:\Ngoding\HendwiSaputra_2311102218\Modul 2\Unguided\Unguided4\Unguide
Nilai K = 100
Nilai f(K) = 1.0000061880

Setelah Modifikasi

Nilai K = 100
Nilai akar 2 = 1.4133387072
PS C:\Ngoding> go run "c:\Ngoding\HendwiSaputra_2311102218\Modul 2\Unguided\Unguided4\Unguide
Nilai K = 1000
Nilai f(K) = 1.0000000624

Setelah Modifikasi

Nilai K = 1000
Nilai akar 2 = 1.4141252651
PS C:\Ngoding>

```

Deskripsi:

Program ini adalah program yang menghitung dua hal: nilai fungsi $f(K)$ dan nilai akar dari 2 yang dimodifikasi. Pada bagian pertama, program meminta input K dari pengguna, kemudian menghitung nilai $f(K)$ berdasarkan rumus yang diberikan, dan mencetak hasilnya. Pada bagian kedua, setelah modifikasi, program kembali meminta input K , kemudian mengguna

kan loop untuk menghitung nilai akar dari 2 berdasarkan formula yang melibatkan K. Hasil dari perhitungan ini kemudian dicetak dengan presisi 10 desimal.

UNGUIDED V

Source Code

```
package main

import "fmt"

var (
    berat_Parsel, beratKg, sisaGram, biayaKg,
    biayaTambahan, total_Biaya int
)

func main() {

    fmt.Print("Berat parsel (gram): ")
    fmt.Scan(&berat_Parsel)

    beratKg = berat_Parsel / 1000
    sisaGram = berat_Parsel % 1000

    biayaKg = beratKg * 10000

    if beratKg > 10 {
        biayaTambahan = 0
    } else {
        if sisaGram >= 500 {
            biayaTambahan = sisaGram * 5
        } else {
            biayaTambahan = sisaGram * 15
        }
    }

    total_Biaya = biayaKg + biayaTambahan

    fmt.Printf("Detail berat: %d kg + %d gr\n", beratKg,
        sisaGram)
    fmt.Printf("Detail biaya: Rp. %d + Rp. %d\n", biayaKg,
        biayaTambahan)
    fmt.Printf("Total biaya: Rp. %d\n", total_Biaya)
}
```

Screenshot

```

PS C:\Ngoding> go run "c:\Ngoding\HendwiSaputra_2311102218\Modul 2\Unguided\Unguided5\Unguided5.go"
Berat parcel (gram): 8500
Detail berat: 8 kg + 500 gr
Detail biaya: Rp. 80000 + Rp. 2500
Total biaya: Rp. 82500
PS C:\Ngoding> go run "c:\Ngoding\HendwiSaputra_2311102218\Modul 2\Unguided\Unguided5\Unguided5.go"
Berat parcel (gram): 9250
Detail berat: 9 kg + 250 gr
Detail biaya: Rp. 90000 + Rp. 3750
Total biaya: Rp. 93750
PS C:\Ngoding> go run "c:\Ngoding\HendwiSaputra_2311102218\Modul 2\Unguided\Unguided5\Unguided5.go"
Berat parcel (gram): 11750
Detail berat: 11 kg + 750 gr
Detail biaya: Rp. 110000 + Rp. 0
Total biaya: Rp. 110000
PS C:\Ngoding>

```

Deskripsi:

Program ini dirancang untuk menghitung biaya pengiriman suatu paket. Pengguna diminta memasukkan berat paket dalam gram, lalu program akan secara otomatis menghitung biaya pengiriman berdasarkan berat tersebut. Program ini membedakan biaya berdasarkan kilogram dan gram sisa. Setiap kilogram memiliki biaya tetap, sedangkan gram sisa dikenakan biaya tambahan yang berbeda-beda tergantung pada beratnya. Setelah perhitungan selesai, program akan menampilkan rincian biaya, termasuk biaya dasar, biaya tambahan, dan total biaya yang harus dibayar.

UNGUIDED VI

Source Code (sebelum perbaikan)

```
package main
```

```

import "fmt"

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

```

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 if nam <= 40 {
        nmK = "E"
    }

    fmt.Println("Nilai mata kuliah: ", nmK)
}

```

Screenshot

```

HendwiSaputra_2311102218\Modul 2\Unguided\Unguided6\Unguided6.go:23:9: cannot use "C" (untyped string constant) as float64 value in assignment
HendwiSaputra_2311102218\Modul 2\Unguided\Unguided6\Unguided6.go:26:9: cannot use "D" (untyped string constant) as float64 value in assignment
HendwiSaputra_2311102218\Modul 2\Unguided\Unguided6\Unguided6.go:28:9: cannot use "E" (untyped string constant) as float64 value in assignment
PS C:\Ngoding> go run "c:\Ngoding\HendwiSaputra_2311102218\Modul 2\Unguided\Unguided6\Unguided6.go"
# command-line-arguments
HendwiSaputra_2311102218\Modul 2\Unguided\Unguided6\Unguided6.go:11:9: cannot use "A" (untyped string constant) as float64 value in assignment
HendwiSaputra_2311102218\Modul 2\Unguided\Unguided6\Unguided6.go:14:9: cannot use "AB" (untyped string constant) as float64 value in assignment
HendwiSaputra_2311102218\Modul 2\Unguided\Unguided6\Unguided6.go:17:9: cannot use "B" (untyped string constant) as float64 value in assignment
HendwiSaputra_2311102218\Modul 2\Unguided\Unguided6\Unguided6.go:20:9: cannot use "BC" (untyped string constant) as float64 value in assignment
HendwiSaputra_2311102218\Modul 2\Unguided\Unguided6\Unguided6.go:23:9: cannot use "C" (untyped string constant) as float64 value in assignment
HendwiSaputra_2311102218\Modul 2\Unguided\Unguided6\Unguided6.go:26:9: cannot use "D" (untyped string constant) as float64 value in assignment
HendwiSaputra_2311102218\Modul 2\Unguided\Unguided6\Unguided6.go:28:9: cannot use "E" (untyped string constant) as float64 value in assignment
PS C:\Ngoding>

```

Deskripsi:

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

Seharusnya jika kita memasukkan nam 80.1 seharusnya program akan menampilkan nilai A namun, program tidak mengeksekusi sesuai spesifikasi

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

```

Pada if nam > 80 {
    nam = "A"
}

```

Seharusnya nam = "A" menjadi nmK = "A", sebab var nam menampung float sehingga tidak bisa digunakan atau error

3. Perbaiki program tersebut! Ujilah dengan masukan: 93.5; 70.6; dan 49.5. Seharusnya keluaran yang diperoleh adalah 'A', 'B', dan 'D'.

```

PS C:\Ngoding> go run "c:\Ngoding\HendwiSaputra_2311102218\Modul 2\Unguided\Unguided6\Unguided6.go"
Nilai akhir mata kuliah: 93.5
Nilai mata kuliah: A
PS C:\Ngoding> go run "c:\Ngoding\HendwiSaputra_2311102218\Modul 2\Unguided\Unguided6\Unguided6.go"
Nilai akhir mata kuliah: 70.6
Nilai mata kuliah: B
PS C:\Ngoding> go run "c:\Ngoding\HendwiSaputra_2311102218\Modul 2\Unguided\Unguided6\Unguided6.go"
Nilai akhir mata kuliah: 49.5
Nilai mata kuliah: D
PS C:\Ngoding>

```

UNGUIDED VII

Source Code

```
package main

import (
    "fmt"
    "math"
)

func cariFaktor(n int) []int {
    var faktor []int
    for i := 1; i <= n; i++ {
        if n%i == 0 {
            faktor = append(faktor, i)
        }
    }
    return faktor
}

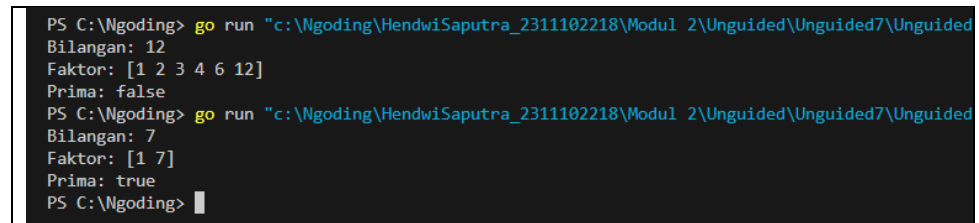
func isPrima(n int) bool {
    if n <= 1 {
        return false
    }
    for i := 2; i <= int(math.Sqrt(float64(n))); i++ {
        if n%i == 0 {
            return false
        }
    }
    return true
}

func main() {
    var angka int
    fmt.Print("Masukkan bilangan: ")
    fmt.Scan(&angka)

    faktor := cariFaktor(angka)
    fmt.Println("Faktor:", faktor)

    if isPrima(angka) {
        fmt.Println("Prima: true")
    } else {
        fmt.Println("Prima: false")
    }
}
```

Screenshot



```
PS C:\Ngoding> go run "c:\Ngoding\HendwiSaputra_2311102218\Modul 2\Unguided\Unguided7\Unguided
Bilangan: 12
Faktor: [1 2 3 4 6 12]
Prima: false
PS C:\Ngoding> go run "c:\Ngoding\HendwiSaputra_2311102218\Modul 2\Unguided\Unguided7\Unguided
Bilangan: 7
Faktor: [1 7]
Prima: true
PS C:\Ngoding>
```

Deskripsi:

Program ini adalah aplikasi sederhana yang ditulis dalam bahasa pemrograman Go untuk menentukan faktor dari suatu bilangan dan memeriksa apakah bilangan tersebut adalah bilangan prima. Fungsi `cariFaktor` menerima bilangan bulat `n` dan mengembalikan slice yang berisi semua faktor `n`. Fungsi `isPrima` memeriksa apakah bilangan `n` adalah bilangan prima dengan memeriksa apakah `n` dapat dibagi habis oleh bilangan lain selain 1 dan dirinya sendiri. Dalam fungsi `main`, program meminta pengguna untuk memasukkan sebuah angka, kemudian memanggil fungsi `cariFaktor` untuk menemukan faktornya dan mencetaknya. Program juga menggunakan fungsi `isPrima` untuk memeriksa apakah bilangan tersebut adalah prima, kemudian mencetak hasil pemeriksaan tersebut.