

**LAPORAN PRAKTIKUM  
ALGORITME DAN PEMEROGRAMAN  
MODUL 2  
REVIEW STRUKTUR KONTROL**



Oleh:

ERVAN HAPIZ

2311102206

IF – 11- 02

**S1 TEKNIK INFORMATIKA  
INSTITUT TEKNOLOGI TELKOM 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 ("//") 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 .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 intruksi 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 nya 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 <sup>9</sup> ..10 <sup>9</sup> 64 bit: -10 <sup>19</sup> ..10 <sup>19</sup> bergantung platform 0..255 0..4294967295 0..(2 <sup>64</sup> -1)
real	float32 float64	32bit: -3.4E+38 .. 3.4E+38 64bit: -1.7E+308 .. 1.7E+308
boolean (atau logikal)	<b>bool</b>	<b>false</b> dan <b>true</b>
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 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

	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

### 1. Guided 1 (2A.1)

#### Source Code

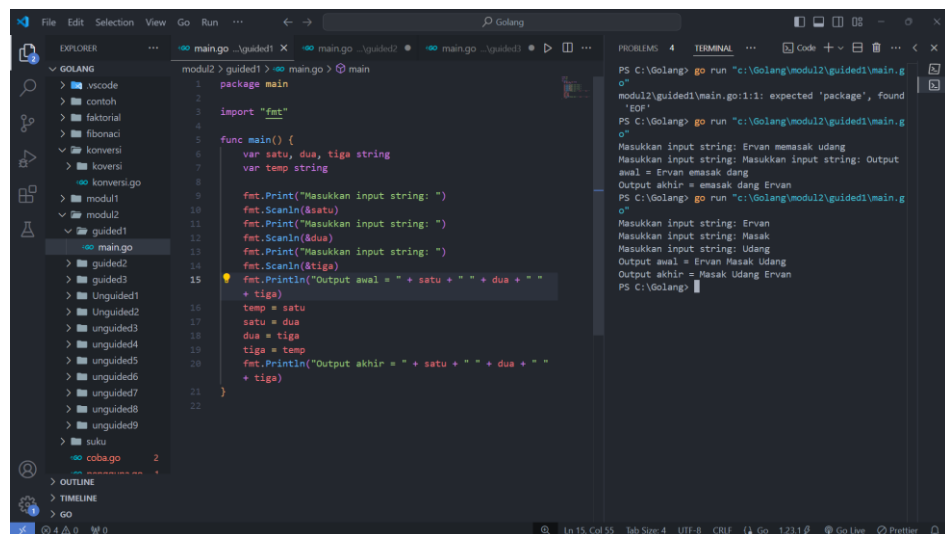
```
package main

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



## Deskripsi

Program ini adalah program mengubah nilai dari suatu variable atau menukan nilai. Pertama terdapat deklarasi variable satu dua dan tiga untuk inputtan dan variable temp untuk menukar nilai variable. Setelah menerima inputan program akan menampillkan inputan awal. Kemudian mulai menukar, nilai varibel satu = dau , dua = tiga dan tiga= satu. Kemudian output akhir adalah nilai variable yang sudah ditukar.

## 2. Guided 2

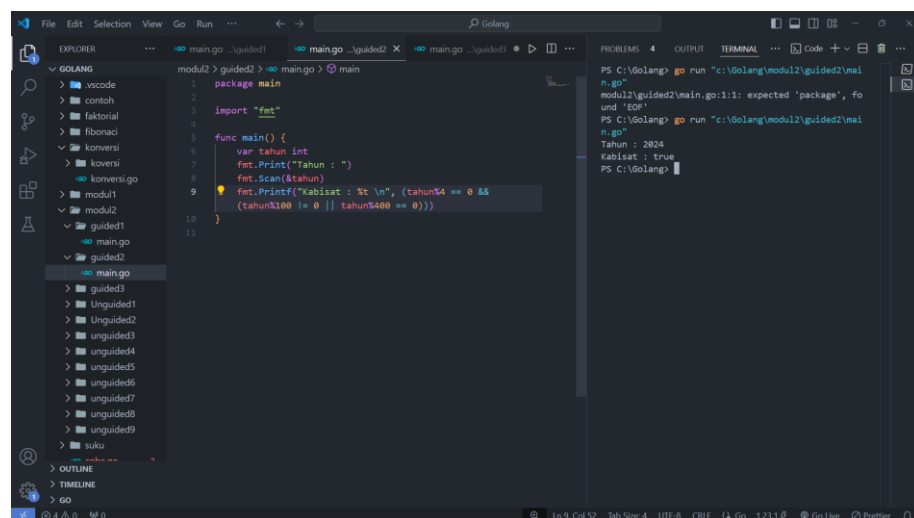
Source Code

```
package main

import "fmt"

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

## Screenshot



## Deskripsi

Program ini adalah untuk melihat apakah tahun tersebut adalah tahun kabisat. Inisiasi variable tahun dengan tipe data int kemudian user akan menginput nilai tahun . program kemudian akan menentukan jika tahun habis dibagi 4 dan tahun habis dibagi 100 atau tahun habis dibagi 400 maka kondisi akan true



### 3. Guided 3

#### Source Code

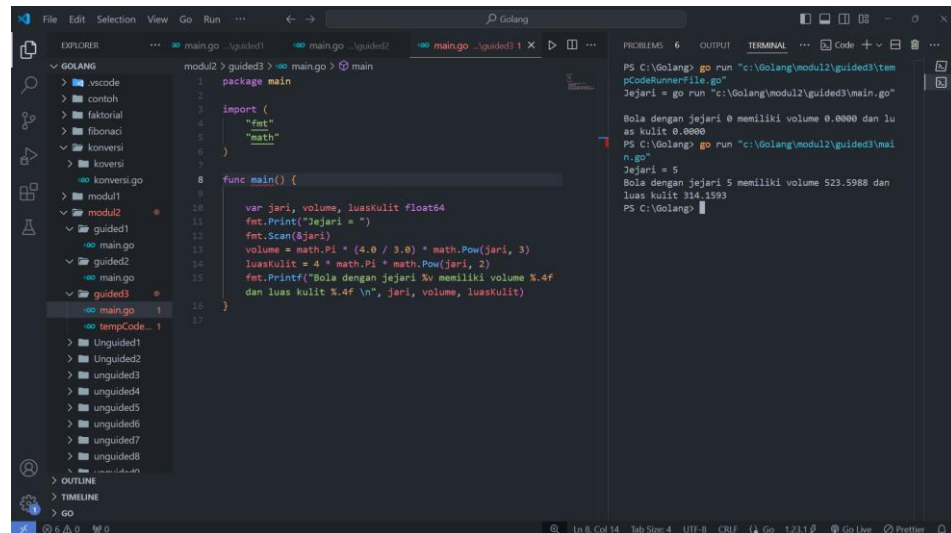
```
package main

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



#### Deskripsi

Program ini adalah program untuk mencari volume dan luas kulit dari sebuah bola. Pertama adalah deklarasi variabel jari, volume dan luas kulit dengan tipe data float. Kemudian program menerima input berupa jari. Kemudian jari akan dimasukkan ke dalam rumus volume dan

luaskulit. Setelah itu program akan menampilkan hasil berupa volume dan luaskulit

### III. UNGUIDED

#### 1. Unguided 1 (2A no 4)

##### Source Code

```
package main

import "fmt"

func Konversi(n float64) {
    var fahrenheit, kelvin, reamur float64

    fahrenheit = n*9.0/5.0 + 32
    kelvin = n + 273.15
    reamur = n * 4.0 / 5.0

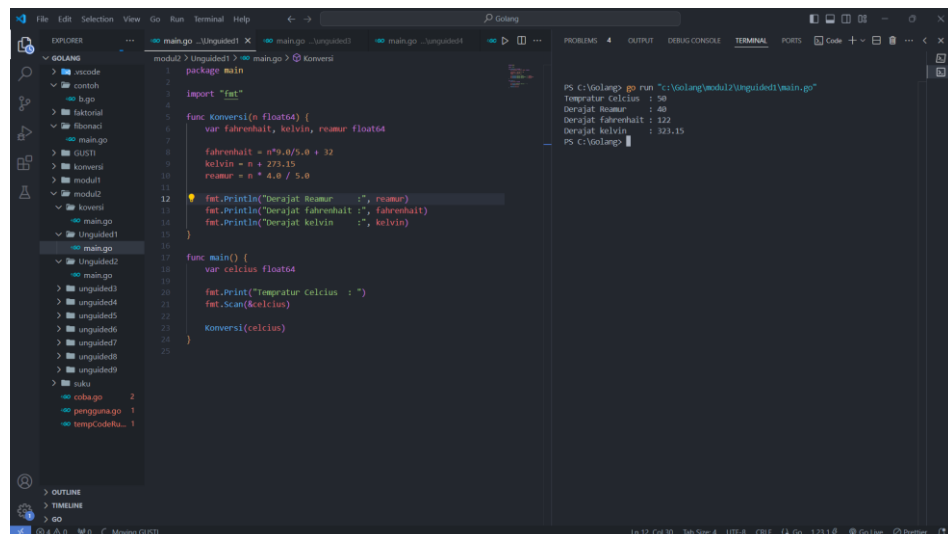
    fmt.Println("Derajat Reamur      :", reamur)
    fmt.Println("Derajat fahrenheit :", fahrenheit)
    fmt.Println("Derajat kelvin       :", kelvin)
}

func main() {
    var celcius float64

    fmt.Print("Tempratur Celcius : ")
    fmt.Scan(&celcius)

    Konversi(celcius)
}
```

##### Screenshot



## Deskripsi

Program ini adalah program untuk mengkonversi suhu dalam celcius ke fahrenheit, kelvin, dan reamur. Pada program terdapat fungsi ( func konversi) untuk mengkonversi dengan parameter n dan tipe data float. Kemudian dalam fungsi terdapat rumus untuk mengubah nilai celcius. Pada func main terdapat deklarasi variable celcius dengan tipe data float, kemudian user menginput nilai celcius. Pemanggilan func konversi dengan parameter celcius untuk mengkonversi. Kemudian program akan menampilkan hasil konversi.

## 2. Unguided 2 (2A no5)

### Source Code

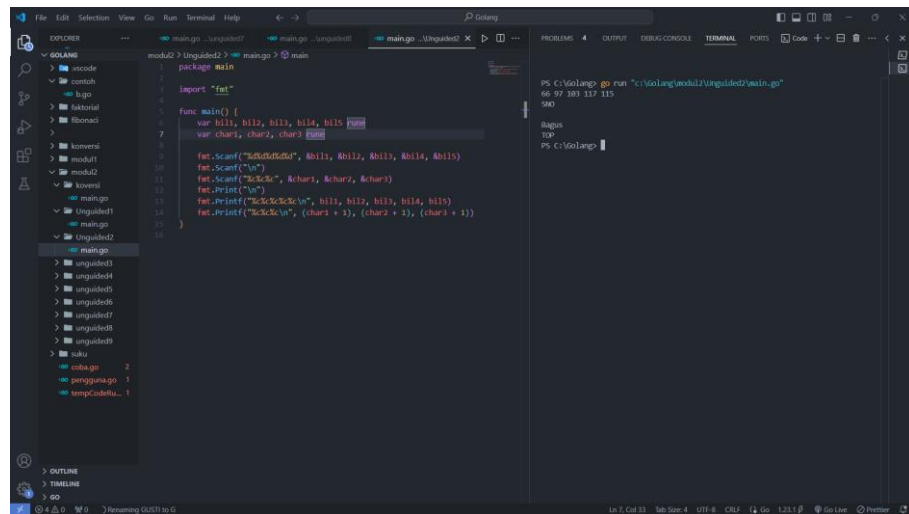
```
package main

import "fmt"

func main() {
    var bil1, bil2, bil3, bil4, bil5 rune
    var char1, char2, char3 rune

    fmt.Scanf("%d%d%d%d%d", &bil1, &bil2, &bil3,
&bil4, &bil5)
    fmt.Scanf("\n")
    fmt.Scanf("%c%c%c", &char1, &char2, &char3)
    fmt.Print("\n")
    fmt.Printf("%c%c%c%c\n", bil1, bil2, bil3,
bil4, bil5)
    fmt.Printf("%c%c%c\n", (char1 + 1), (char2 +
1), (char3 + 1))
}
```

### Screenshot



## Deskripsi

Program ini adalah program untuk mengkonversi atau mengubah nilai integer ke dalam bentuk karakter sesuai dengan table ASCII. Pada program terdapat deklarasi variable bil1, bil2, bil3, bil4, bil5 dengan tipe data rune ( tipe data rune adalah tipe data untuk Unicode ) dan terdapat variable char1, char2, char3 dengan tipe data rune yang akan kita gunakan menginput sebuah karakter. Pertama program akan menerima inputan dari user. Inputan pertama adalah 5 buah bilangan bulat dan inputan kedua adalah 3 buah karakter. Kemudian program akan menampilkan 5 buah karakter yang didapat dari mengubah inputan user( hal tersebut terjadi karena dalam pemanggilan variable di fmt.printf terdapat %c yang digunakan untuk mengubah nilai integer ke karakter dengan tipe data Unicode). Kemudian output kedua adalah 3 karakter yang sudah di ubah dengan menambah satu karakter misal inputan A maka akan menampilkan satu karakter setelahnya yaitu B. output terakhir adalah 5 karakter dan 3 karakter

### 3. Unguided 3 (2B no 1)

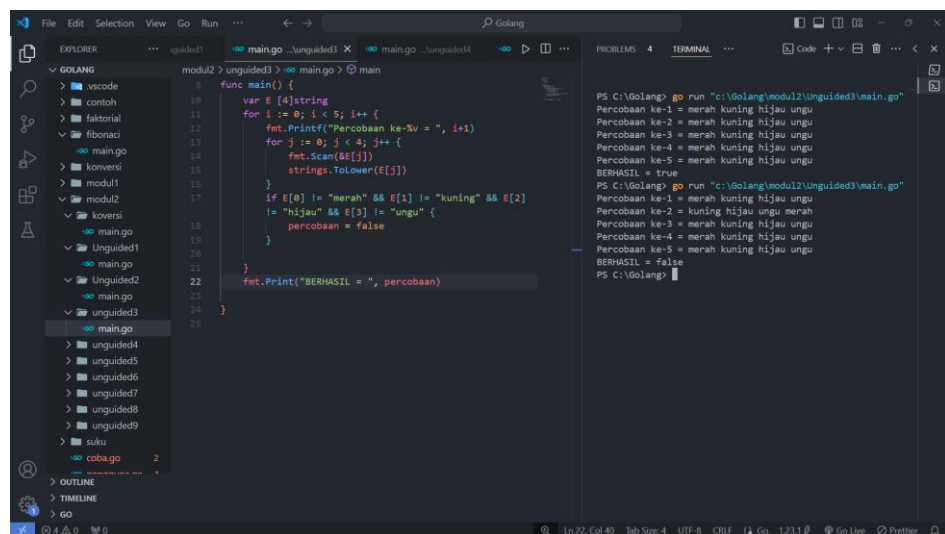
#### Source Code

```
package main

import (
    "fmt"
    "strings"
)

func main() {
    var percobaan = true
    var E [4]string
    for i := 0; i < 5; i++ {
        fmt.Printf("Percobaan ke-%v = ", i+1)
        for j := 0; j < 4; j++ {
            fmt.Scan(&E[j])
            strings.ToLower(E[j])
        }
        if E[0] != "merah" && E[1] != "kuning" &&
E[2] != "hijau" && E[3] != "ungu" {
            percobaan = false
        }
    }
    fmt.Print("BERHASIL = ", percobaan)
}
```

#### Screenshot



## Deskripsi

Program ini adalah program untuk menentukan apakah percobaan yang dilakukan oleh siswa berhasil atau tidak. Pertama terdapat deklarasi variable E yang merupakan array dengan size 4 dengan tipe data string dan terdapat juga variable percobaan dengan tipe data Boolean. Kemudian terdapat perulangan for dengan kondisi dari 0 sampai 5 (karena siswa melakukan percobaan sebanyak 5 kali ). Lalu program akan melakukan perulangan lagi dengan kondisi 0 sampai 4 yang digunakan untuk menginput nilai variable E dari index 0 sampai 3. Fungsi strings.ToLower() untuk mengubah hurup kapital ke hurup kecil. Setelah inputan semua selesai akan terjadi kondisi if untuk menentukan apakah percobaan berhasil atau tidak. Kondisi if akan memeriksa apakah E[0] bukan merah, E[1] bukan kuning, E[2] bukan hijau, E[3] bukan ungu di setiap percobaan artinya program akan memeriksa sebanyak 5 kali. Ketika kondisi itu terpenuhi maka variable percobaan akan bernilai false atau percobaan gagal dan Ketika kondisi if tidak terpenuhi maka percobaan berhasil.

## 4. Unguided 4 (2B no 2)

### Source Code

```
package main

import (
    "fmt"
)

func main() {
    var E int

    fmt.Print("Banyak bunga : ")
    fmt.Scan(&E)

    bunga := make([]string, E)
    var pita string
    if E <= 0 {
        fmt.Print("Pita : ", pita)
    } else {

        for i := 0; i < E; i++ {
            fmt.Printf("Bunga %v : ", i+1)
            fmt.Scan(&bunga[i])
        }
        for i := 0; i < E; i++ {
            pita += bunga[i] + "-"
        }
        fmt.Printf("Pita : %v ", pita)
    }
    //MODIFIKASI
```

```

        fmt.Print("\n\nAFTER MODIFIKASI\n\n")

        var pital string
        var Temp string
        var R = 0
        fmt.Printf("Bunga %v : ", R+1)
        fmt.Scan(&Temp)
        if Temp == "SELESAI" {
            fmt.Printf("Pita : %v \n", pital)
            fmt.Println("Bunga :", R)
        } else {
            for Temp != "SELESAI" {

                if Temp != "SELESAI" {
                    pital += Temp + "-"
                }
                fmt.Printf("Bunga %v : ", R+2)
                fmt.Scan(&Temp)
                R++
            }
            fmt.Printf("Pita : %v \n", pital)
            fmt.Println("Bunga :", R)
        }
    }
}

```

## Screenshot

The screenshot shows a Go IDE with the following components:

- EXPLORER:** A file tree on the left showing a project structure with folders like 'main.go', 'modul2', and 'unguided4'.
- EDITOR:** The main window displaying the Go code. The code is a function `main()` that prints the number of flowers, scans for flower names, and concatenates them into a string 'Pita' until the user enters 'SELESAI'. It then prints the final 'Pita' string and the total number of flowers.
- TERMINAL:** The output window on the right showing the execution results:
 

```

PS C:\Golang> go run "c:\Golang\modul2\unguided4\main.go"
Banyak bunga : 3
Bunga 1 : mawar
Bunga 2 : tulip
Bunga 3 : matahari
Pita : mawar-tulip-matahari-

AFTER MODIFIKASI

Bunga 1 : mawar
Bunga 2 : matahari
Bunga 3 : tulip
Bunga 4 : kamboja
Bunga 5 : SELESAI
Pita : mawar-matahari-tulip-kamboja-
Bunga : 4
PS C:\Golang>
      
```

## Deskripsi



Program ini adalah program untuk menginput nama-nama bunga yang kemudian pada output akan ditampilkan namun dipisah oleh “-“. Pada program terdapat sourcecode yang sebelum dan sesudah modifikasi. Dimana sebelum modifikasi terdapat deklarasi variable E untuk menentukan jumlah bunga yang akan diinput. Program kemudian akan menerima inputan dan mengisi nilai dari variable E. setelah itu terdapat deklarasi untuk membuka array dengan nama bunga ( bunga := make([]string, E)) yang memiliki size E dengan tipe data string. Terdapat juga variable pita dengan tipe data string yang akan digunakan untuk menggabungkan nilai dari array. Kemudian terdapat percabangan if dengan kondisi ketika inputan  $E \leq 0$ . Ketika terpenuhi maka program akan menampilkan output pita : kosong karena belum ada inputan bunga dari user. Ketika kondisi if tidak terpenuhi maka program akan menjalankan perulangan yang digunakan untuk menginput nilai dari array bunga. Perulangan akan berlangsung sebanyak E jumlah bunga. Kemudian terdapat perulangan untuk menggabungkan nilai bunga menjadi satu string yang dipisahkan “-“. Dan program akan menampilkan output nama bunga yang diinput.

Setelah modifikasi program akan melakukan input bunga langsung tanpa menginput jumlah bunga terlebih dahulu akan berhenti ketika user menginput SELESAI. Pertama terdapat deklarasi variabel pita1 (untuk membedakan dari sebelum modifikasi) dan variabel temp untuk menginput nama bunga. Variabel R untuk menghitung jumlah bunga yang diinput. If temp == SELESAI maka program akan menampilkan pita kosong dan bunga 0. Kemudian else akan berjalan seperti sebelum modifikasi namun tidak menggunakan array . kemudian sama seperti sebelumnya hanya pada tampilan output terdapat jumlah bunga yang sudah diinput.

## 5. Unguided 5 (2B no3)

### Source Code

```
package main

import "fmt"

func main() {
    var berat_kiri, berat_kanan float64
    var kondisi = false
    var kondisi1 = false

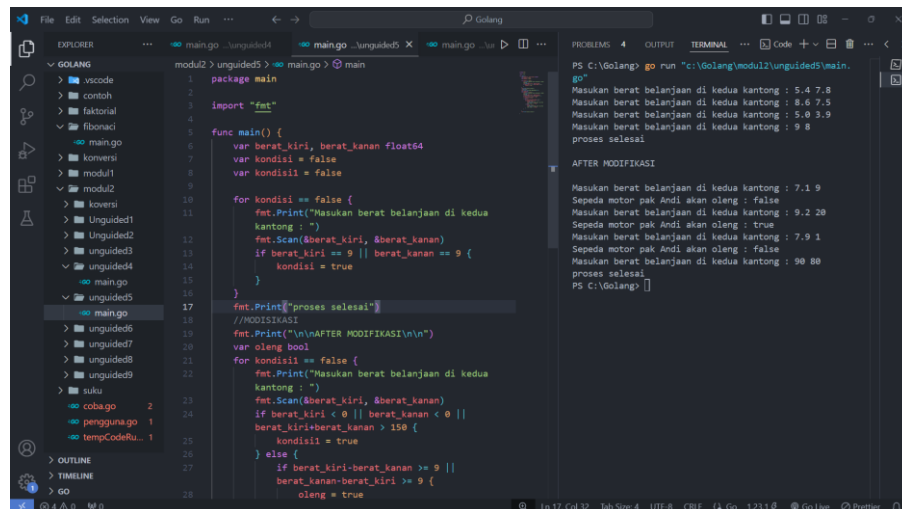
    for kondisi == false {
```

```

        fmt.Print("Masukan berat belanjaan di kedua
kantong : ")
        fmt.Scan(&berat_kiri, &berat_kanan)
        if berat_kiri == 9 || berat_kanan == 9 {
            kondisi = true
        }
    }
    fmt.Print("proses selesai")
    //MODISIKASI
    fmt.Print("\n\nAFTER MODIFIKASI\n\n")
    var oleng bool
    for kondisil == false {
        fmt.Print("Masukan berat belanjaan di kedua
kantong : ")
        fmt.Scan(&berat_kiri, &berat_kanan)
        if berat_kiri < 0 || berat_kanan < 0 ||
berat_kiri+berat_kanan > 150 {
            kondisil = true
        } else {
            if berat_kiri-berat_kanan >= 9 ||
berat_kanan-berat_kiri >= 9 {
                oleng = true
                fmt.Println("Sepeda motor pak Andi
akan oleng :", oleng)
            } else {
                oleng = false
                fmt.Println("Sepeda motor pak Andi
akan oleng :", oleng)
            }
        }
    }
    fmt.Print("proses selesai")
}

```

**Screenshot**



```
package main
import "fmt"

func main() {
    var berat_kiri, berat_kanan float64
    var kondisi = false
    var kondisi1 = false

    for kondisi == false {
        fmt.Println("Masukan berat belanja di kedua kantong : ")
        fmt.Scan(&berat_kiri, &berat_kanan)
        if berat_kiri == 9 || berat_kanan == 9 {
            kondisi = true
        }
    }
    fmt.Println("proses selesai")

    //MODIFIKASI
    fmt.Println("\n\nAFTER MODIFIKASI\n\n")
    var oleng bool
    for kondisi1 == false {
        fmt.Println("Masukan berat belanja di kedua kantong : ")
        fmt.Scan(&berat_kiri, &berat_kanan)
        if berat_kiri < 0 || berat_kanan < 0 ||
            berat_kiri+berat_kanan > 150 {
            kondisi1 = true
        } else {
            if berat_kiri-berat_kanan >= 9 ||
                berat_kanan-berat_kiri >= 9 {
                oleng = true
            }
        }
    }
}
```

PS C:\Golang> go run "c:\Golang\modul2\unguided5\main.go"

Masukan berat belanja di kedua kantong : 5.4 7.8  
Masukan berat belanja di kedua kantong : 8.6 7.5  
Masukan berat belanja di kedua kantong : 5.0 3.9  
Masukan berat belanja di kedua kantong : 9 8  
proses selesai

AFTER MODIFIKASI

Masukan berat belanja di kedua kantong : 7.1 9  
Sepeda motor pak Andi akan oleng : false  
Masukan berat belanja di kedua kantong : 9.2 20  
Sepeda motor pak Andi akan oleng : true  
Masukan berat belanja di kedua kantong : 7.9 1  
Sepeda motor pak Andi akan oleng : false  
Masukan berat belanja di kedua kantong : 90 80  
proses selesai  
PS C:\Golang>

## Deskripsi

Program ini adalah program untuk memasukan barang di kanan dan kiri motor. Program akan selesai menerima input Ketika salah satu berat 9. Pertama terdapat deklarasi variable berat\_kiri dan berat\_kanan dengan tipe data float. Variable kondisi dan kondisi1 (untuk modifikasi) dengan tipe data Boolean. Tedapat kondisi perulangan for untuk memeriksa kondisi == false kemududiaan program menerima inputan untuk mengisi berat kiri dan kanan. Dalam perulangan terdapat If dengan kondisi berat\_kiri ==9 atau berat\_kanan ==9 maka nilai var kondisi akan true yang membuat perulangan terhenti dan menampilkan program selesai.

Modifikasi program adalah mengubah batas inputan yang sebelumnya program akan selesai ketyika ada yang 9, pada modifikasi program akan berhenti Ketika inputan berjumlah 150 atau salah satu inputan bernilai negative. Dan menambahkan tampilan Ketika berat yang diinput memiliki selisih 9 maka motor pak Andi akan oleng. Terdapat tambahan variable oleng tipe data bool. Kemudian program akan melakukan perulang seperti sebelumnya. Bedanya terdapat pada kondisi if Dimana if pertama untuk mengecek apaka ada inputan bernilai negative atau kedua inputan berjumlah 150. Yang Ketika terpenuhi kondisi akan bernilai true dan perulang selesai. Kemudian Ketika tidak terpenuhi maka akan terjadi percabangan lagi untuk menentukan apakah motor pak Andi akan oleng atau tidak. If selisih sama dengan 9 maka oleng bernilai true

## 6. Unguided 6 (2B no 4)

### Source Code

```
package main

import (
    "fmt"
)

func main() {
    var k int
    var F float64

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

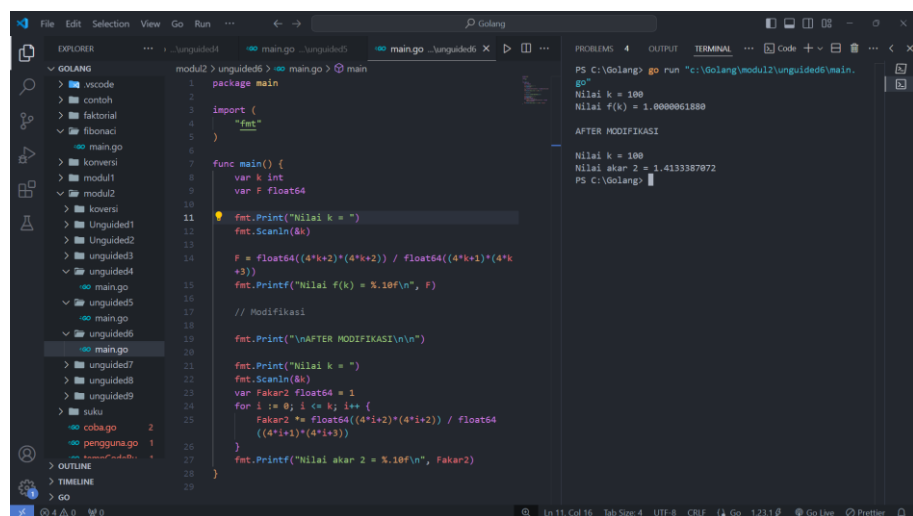
    F = float64((4*k+2)*(4*k+2)) /
float64((4*k+1)*(4*k+3))
    fmt.Printf("Nilai f(k) = %.10f\n", F)

    // Modifikasi

    fmt.Print("\nAFTER MODIFIKASI\n\n")

    fmt.Print("Nilai k = ")
    fmt.Scanln(&k)
    var Fakar2 float64 = 1
    for i := 0; i <= k; i++ {
        Fakar2 *= float64((4*i+2)*(4*i+2)) /
float64((4*i+1)*(4*i+3))
    }
    fmt.Printf("Nilai akar 2 = %.10f\n", Fakar2)
}
```

### Screenshot



## Deskripsi

Program ini adalah program untuk menghitung nilai dari fungsi k. Terdapat deklarasi variable k dengan tipe integer dan f tipe data float. Program akan menerima inputan untuk nilai k yang akan dimasukan kedalam rumus fungsi k. kemudian varibel f akan menampung hasil dari perhitungan rumus. Kemudian program akan menampilkan nilai dari f.

Modifikasi dari program adalah mencari nilai dari akar dua dengan menggunakan fungsi k. dengan menambahkan variable akar2 dengan nilai 1. Kemudian inputan k akan digunakan sebagai batas dari perulangan karena rumus yang digunakan terdapat sigma. Pada perulangan akan terjadi perkalian hasil mulai dari k =0 hingg yang di tentukan kemudian disimpan di variable akar2. Kemudian tampilan output berupa nilai kahir dari akar2

## 7. Unguided 7 (2C no 1)

### Source Code

```
package main

import "fmt"

func main() {
    var berat, kg, gr int
    var biaya_kg, biaya_gr, total int

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

    kg = berat / 1000
    gr = berat % 1000

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

    biaya_kg = kg * 10000
    if gr < 500 {
        biaya_gr = gr * 15
    } else {
        biaya_gr = gr * 5
    }

    if kg > 10 {
        total = biaya_kg
    } else {
        total = biaya_kg + biaya_gr
    }

    fmt.Printf("Detai biaya : RP.%v + RP.%v\n",
biaya_kg, biaya_gr)
    fmt.Printf("Total biaya : RP.%v", total)
```

```
}
```

## Screenshot

```
modul2 > unguided7 > main.go > main
import "fmt"

func main() {
    var berat, kg, gr int
    var biaya_kg, biaya_gr, total int

    fmt.Print("Berat parsal (gram) : ")
    fmt.Scan(&berat)

    kg = berat / 1000
    gr = berat % 1000

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

    biaya_kg = kg * 10000

    if gr < 500 {
        biaya_gr = gr * 15
    } else {
        biaya_gr = gr * 5
    }

    if kg > 10 {
        total = biaya_kg
    } else {
        total = biaya_kg + biaya_gr
    }

    fmt.Printf("Detail biaya : RP.%v + RP.%v\n", biaya_kg, biaya_gr)
    fmt.Printf("Total biaya : RP.%v", total)
}
```

```
PS C:\Golang> go run "c:\Golang\modul2\unguided7\main.go"
Berat parsal (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\modul2\unguided7\main.go"
Berat parsal (gram) : 11750
Detail berat : 11 kg + 750 gr
Detail biaya : RP.110000 + RP.3750
Total biaya : RP.110000
PS C:\Golang>
```

## Deskripsi

Program ini adalah program untuk perhitungan biaya kirim berdasarkan berat parsal. Pertama adalah deklarasi variable berat, kg, gr, biaya\_kg, biaya\_kg, total dengan tipe data int. program akan menerima inputan dari user yaitu berat dengan satuan gram. Kemudian berat tadi akan dipisahkan antara satuan kg dan gram dengan berat dibagi 1000 untuk menentukan satuan kg dan berat modulus 1000 untuk menentukan berat dalam gram. Kemudian akan ditampilkan berat detail yaitu dalam kg + dalam gram. Selanjutnya adalah penentuan biaya, untuk satuan kg dikenakan 10000/kg jadi berat dalam kg akan dikalikan dengan 10000. Kemudian berat satuan gram jikan lebih atau sama dengan 500 akan dikenakan 5 per gram sedangkan jika dibawah 500 akan dikenakan 15 per gram. Kemudian perhitungan berat total, jika berat dalam satuan kg lebih dari 10 maka total biaya sama dengan biaya\_kg, namun jika tidak maka total biaya adalah biaya\_kg+ biaya\_gr. Kemudian tampilan out berupa detail biaya dan biaya total.

### 8. Unguided 3

- A. Jika nam diberikan 80.1 maka keluaran dari program adalah tidak ada karena 80.1 masuk kedalam if nam > 80 namun kode yang dieksekusi tidak benar Dimana dalam if var nam = "A" yang seharusnya itu adalah inisiasi dari variable nmk, namun tidak

diinisiasikan dengan benar sehingga eksekusi dari soal tidak sesuai

- B.** Kasus yang ada adalah if- else if yang tidak maksimal, tidak terdapat blok else if Ketika kondisi if tidak memenuhi. Urutan harusnya adalah pertama user menginput nilai nam, kemudian dilihat kondisi if else, Dimana kondisi memenuhi maka akan dikategorikan dengan nmk yang sesuai dan terakhir tampilan output seperti sourcecode dibawah ini

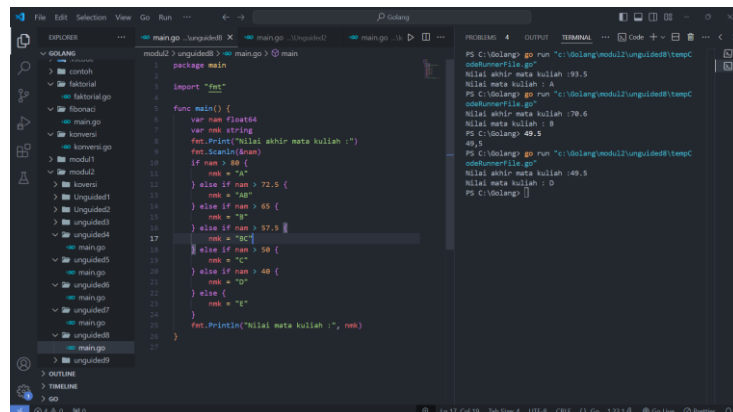
**Source Code**

```
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)
}
```

**C.**



## Deskripsi

Program ini adalah program untuk menginput nilai mahasiswa dan dikategorikan sesuai nilainya. Deklarasi variable nam untuk inputan dan nmk untuk kategori. Program menerima inputan kemudian program akan melakukan pengecekan kondisi misalkan nam 70 maka akan dicek  $70 > 65$  maka  $nmk = B$  dan tampilan output akan menampilkan B

## 9. Unguided 9

### Source Code

```

package main

import "fmt"

func main() {
    var bil int

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

    fmt.Print("Faktor : ")
    for i := 1; i <= bil; i++ {
        if bil%i == 0 {
            fmt.Printf("%d ", i)
        }
    }

    //menentukan bilangan prima
    var prima bool

    for i := 2; i*i <= bil; i++ {
        if bil%i == 0 {
            prima = false
        } else {

```



```

        prima = true
    }
}

fmt.Print("\nPrima : ", prima)
}

```

## Screenshot

```

package main

import "fmt"

func main() {
    var bil int

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

    fmt.Print("Faktor : ")
    for i := 1; i <= bil; i++ {
        if bil%i == 0 {
            fmt.Printf("%d ", i)
        }
    }

    //menentukan bilangan prima
    var prima bool
    for i := 2; i*i <= bil; i++ {
        if bil%i == 0 {
            prima = false
        } else {
            prima = true
        }
    }

    fmt.Print("\nPrima : ", prima)
}

```

PS C:\Golang> go run "C:\Golang\modul2\unguided9\main.go"

Bilangan : 16  
Faktor : 1 2 4 8 16  
Prima : false  
PS C:\Golang>

## Deskripsi

Program ini adalah program untuk menentukan factor dari nilai yang diinputkan dan menentukan apakah bilangan tersebut bilangan prima atau tidak. Pertama terdapat deklarasi variabel `bil` untuk inputan. Kemudian program akan menerima inputan dari user. Terdapat perulangan untuk menentukan factor bilangan tersebut. Perulangan akan berjalan sebanyak bilangan yang diinput. Kemudian dalam perulangan terdapat `if bil % i` maka akan ditampilkan semua `i` yang habis membagi `bil` dan program akan menampilkan factor. Kemudian untuk menentukan bilangan prima atau tidak. Program akan melakukan perulangan dimulai dari `i = 2` hingga `i*i` sampai dengan `bil`. Dalam perulangan terdapat kondisi `if` jika `bil` habis dibagi `i` maka `prima = false` dan ketika tidak terpenuhi maka `prima = true`. dan output akan menampilkan bilangan tersebut `prima true` atau `false`.