

**LAPORAN PRAKTIKUM  
ALGORITME DAN PEMEROGRAMAN**

**MODUL 2  
REVIEW STRUKTUR KONTROL**



Oleh:

MUHAMMAD DAFFA AL FAIZ

2311102237

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,
- Setiap 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

- 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

**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!

**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!

### 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

**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

## **II. GUIDED**

### Guided 1

#### Source Code

```

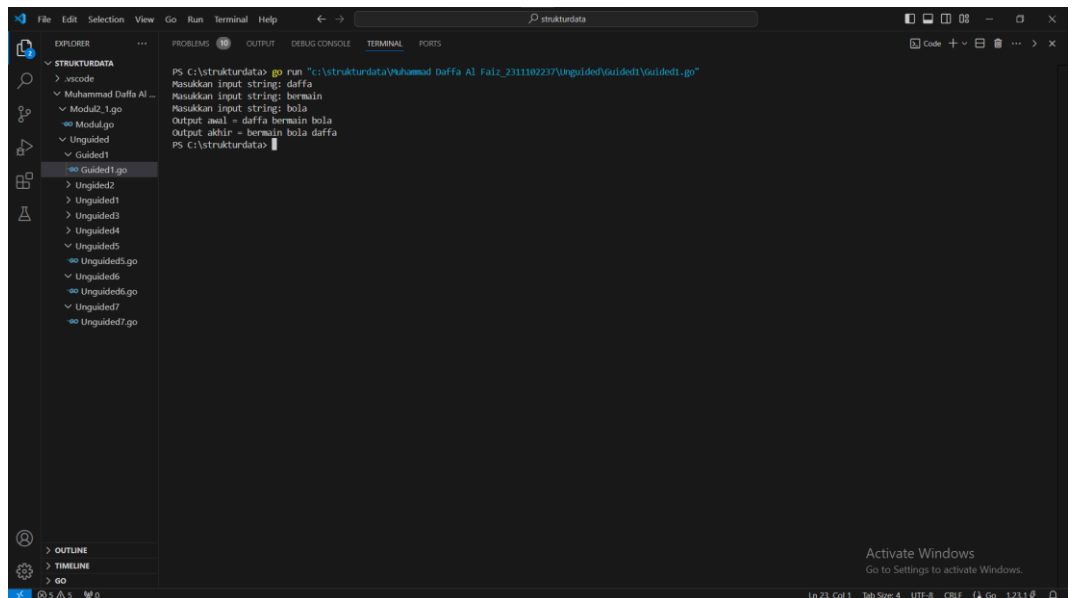
package main

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 mengubah nilai dari suatu variable atau menentukan nilai

## Guided 2

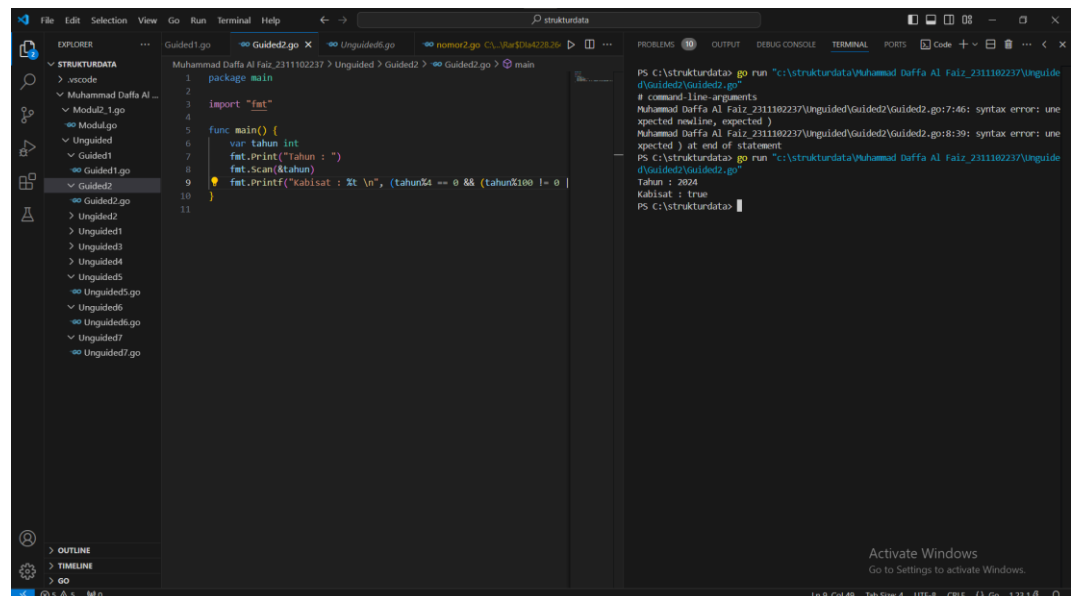
### Source Code

```
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 tahun kabisat.

## Guided 3

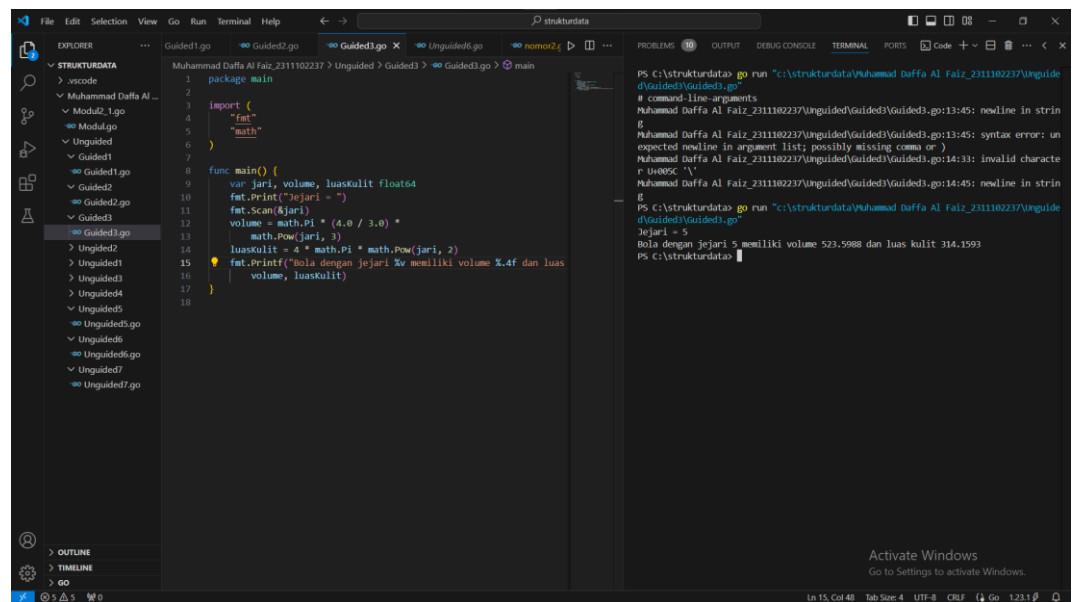
### Source Code

```
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 untuk mencari volume dan luas kulit dari sebuah bola.



### III. UNGUIDED

#### Unguide 1

#### Source Code

```
package main

import (
    "fmt"
    "strings"
)

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

    numExperiments := 5

    for {
        success := true

        for i := 1; i <= numExperiments; i++ {
            var colors [4]string
            fmt.Printf("Percobaan %d: ", i)
            for j := 0; j < 4; j++ {
                fmt.Scan(&colors[j])
            }

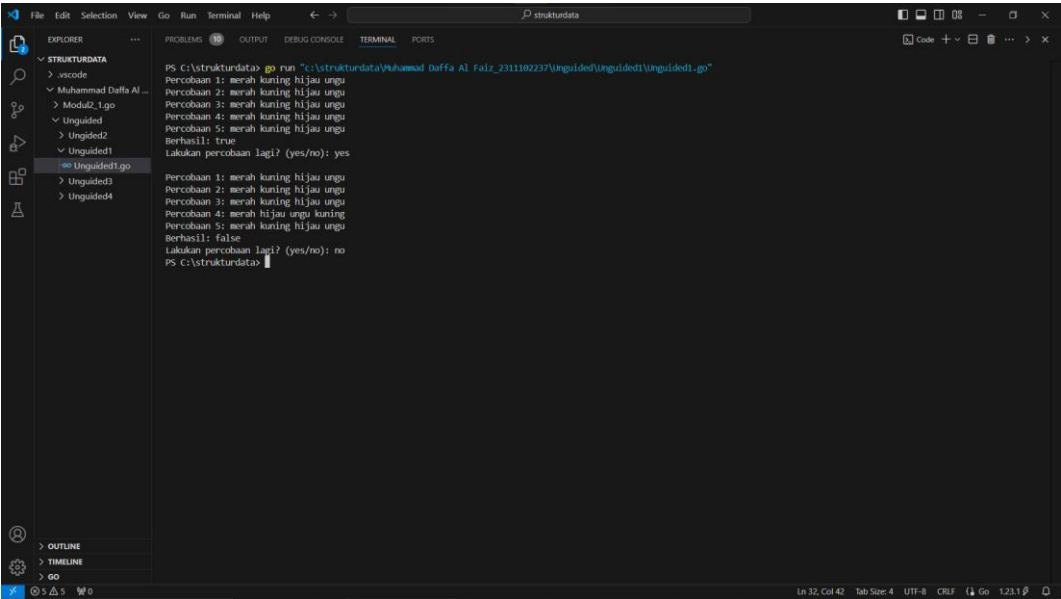
            for j := 0; j < 4; j++ {
                if strings.ToLower(colors[j]) != correctOrder[j] {
                    success = false
                    break
                }
            }
        }

        if success {
            fmt.Println("Berhasil: true")
        } else {
            fmt.Println("Berhasil: false")
        }

        var again string
        fmt.Print("Lakukan percobaan lagi? (yes/no): ")
        fmt.Scan(&again)
        if strings.ToLower(again) != "yes" {
```

```
        break
    }
    fmt.Println() // Add a blank line for readability
}
}
```

Screenshot



Deskripsi

Program ini adalah untuk melakukan percobaan sederhana Dimana untuk memasukan urutan warna tertentu dan Program ini juga mengevaluasi apakah urutan warna yang dimasukan oleh pengguna sesuai dengan urutan yang telah ditentukan sebelumnya.

Unguided 2

Source Code

```
package main

import "fmt"

func main() {
```

```
var N int
fmt.Print("N: ")
fmt.Scanln(&N)
var pita string
var bunga string
var count int

for {
    fmt.Printf("Bunga %d: ", count+1)
    fmt.Scanln(&bunga)

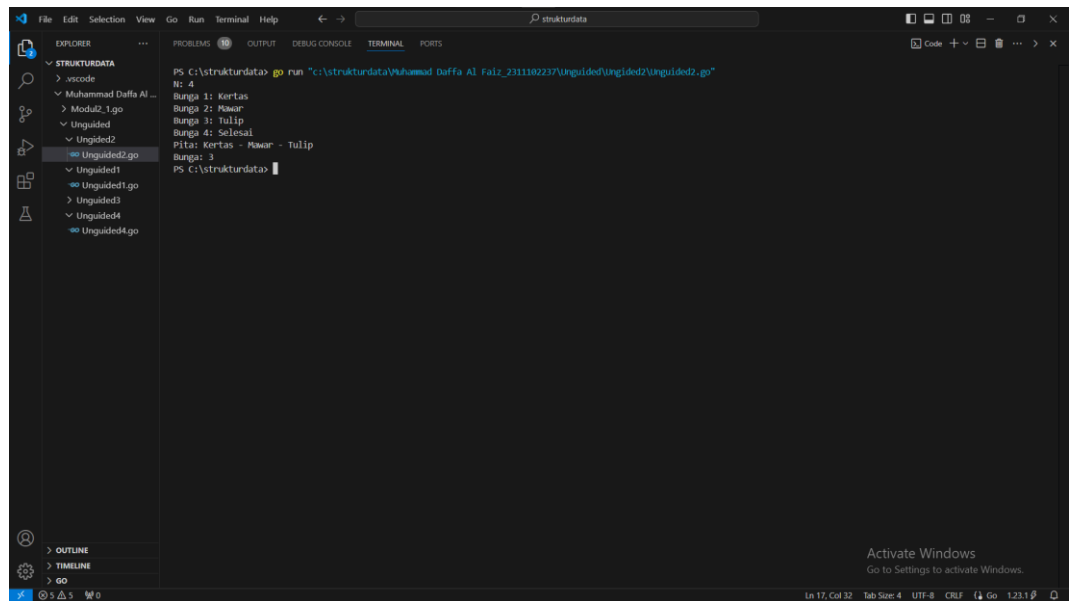
    if bunga == "Selesai" {
        break
    }

    pita += bunga + " - "
    count++
}

// Hapus tanda "-" yang berlebihan di akhir
pita = pita[:len(pita)-3]

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

Screenshot



## Deskripsi

Program ini adalah untuk membuat sebuah program sederhana yang meminta pengguna untuk memasukkan nama bunga sebanyak yang diinginkan sehingga pengguna memasukkan kata “Selesai”.

## Unguided 3

### Source Code

```
package main

import "fmt"

func main() {
    var berat1, berat2 float64

    for {
        fmt.Print("Masukan berat belanjaan di kedua kantong: ")
        fmt.Scan(&berat1, &berat2)

        // Kondisi berhenti:
        if berat1 < 0 || berat2 < 0 || berat1+berat2 > 150 {
            fmt.Println("Selesai.")
            break
        }
    }
}
```

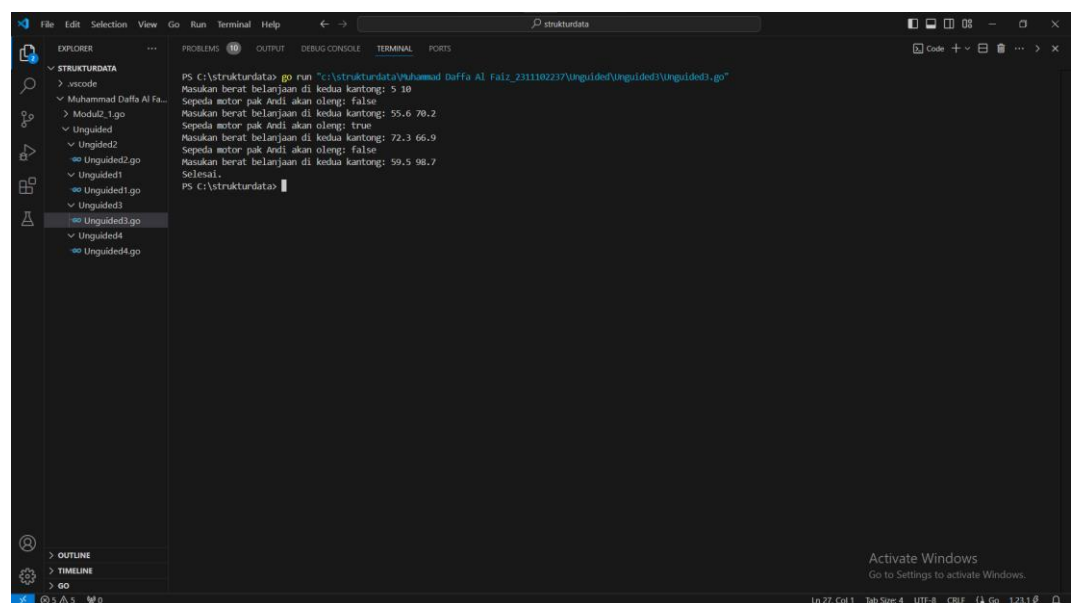
```

    }

    // Hitung selisih berat dan tentukan apakah sepeda akan oleng
    selisih := berat1 - berat2
    if selisih < 0 {
        selisih = -selisih // Ambil nilai absolut selisih
    }
    akanOleng := selisih >= 9
    fmt.Printf("Sepeda motor pak Andi akan oleng: %t\n", akanOleng)
}
}

```

## Screenshot



## Deskripsi

Program ini adalah untuk menentukan apakah sebuah sepeda motor akan oleng berdasarkan berat beban yang diletakan dikedua keranjangnya.

## Unguided 4

## Source Code

```
package main
```

```

import (
    "fmt"
)

func main() {
    var k int
    var F, Fakar2 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)

    // Modification

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

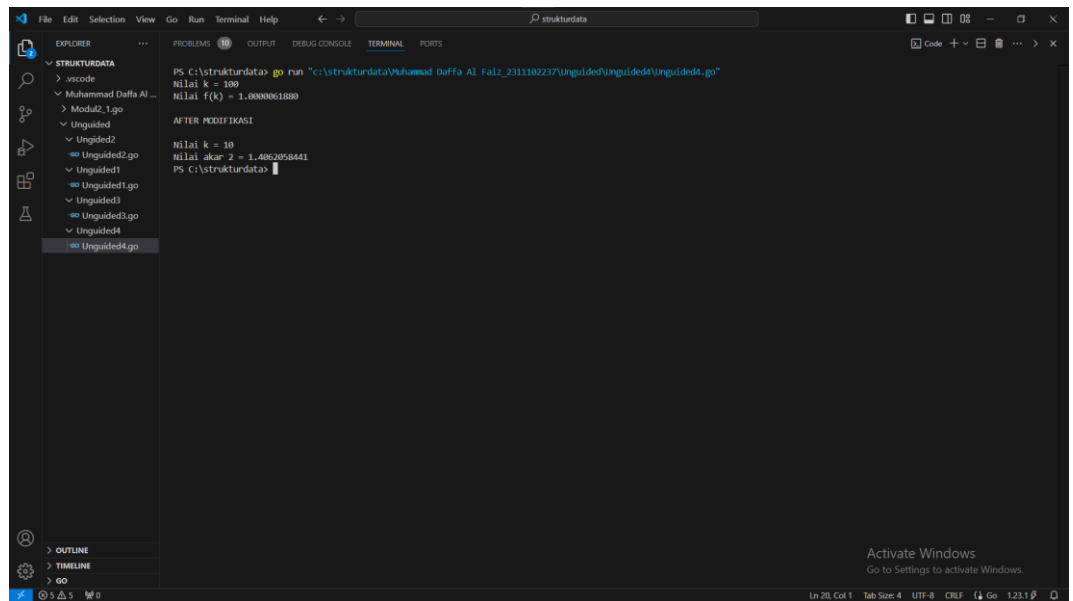
    fmt.Print("Nilai k = ")
    fmt.Scanln(&k)

    Fakar2 = 1
    for i := 0; i <= k; i++ {
        jumlah := float64((4*i+2)*(4*i+2)) / float64((4*i+1)*(4*i+3))
        Fakar2 *= jumlah
    }

    fmt.Printf("Nilai akar 2 = %.10f\n", Fakar2)
}

```

Screenshot



## Deskripsi

Program ini adalah untuk menghitung dan mencetak nilai dari rumus matematika.

## Unguided 5

### Source Code

```
package main

import "fmt"

func main() {
    var beratParsel int
    var beratKg, sisaGram, biayaKg, biayaTambahan, totalBiaya int

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

    beratKg = beratParsel / 1000
    sisaGram = beratParsel % 1000

    biayaKg = beratKg * 10000

    // Kondisi untuk menentukan biaya tambahan
    if sisaGram > 0 {
```

```

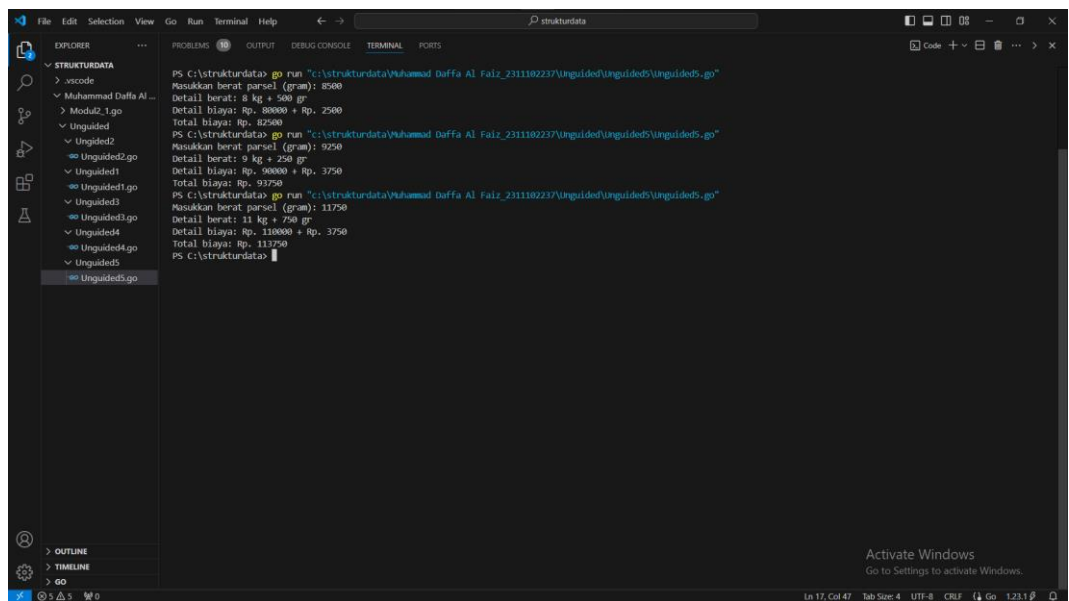
        if sisaGram >= 500 {
            biayaTambahan = sisaGram * 5
        } else {
            biayaTambahan = sisaGram * 15
        }
    }

    totalBiaya = 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", totalBiaya)
}

```

## Screenshot



```

PS C:\strukturdata> go run "C:\strukturdata\Muhammad Daffa Al Faiz_2311180223\Unguided\Unguided5\Unguided5.go"
Masukkan berat parcel (gram): 8500
Detail berat: 2 kg + 500 gr
Detail biaya: Rp. 80000 + Rp. 2500
Total biaya: Rp. 82500

PS C:\strukturdata> go run "C:\strukturdata\Muhammad Daffa Al Faiz_2311180223\Unguided\Unguided5\Unguided5.go"
Masukkan berat parcel (gram): 9250
Detail berat: 9 kg + 250 gr
Detail biaya: Rp. 90000 + Rp. 3750
Total biaya: Rp. 93750

PS C:\strukturdata> go run "C:\strukturdata\Muhammad Daffa Al Faiz_2311180223\Unguided\Unguided5\Unguided5.go"
Masukkan berat parcel (gram): 11750
Detail berat: 11 kg + 750 gr
Detail biaya: Rp. 110000 + Rp. 3750
Total biaya: Rp. 113750

PS C:\strukturdata>

```

## Deskripsi

Program ini adalah untuk menghitung biaya pengiriman suatu parcel berdasarkan beratnya.

## Unguided 6

## Source Code



A. Program tidak menghasilkan output yang diharapkan karena tidak ada kondisi if yang benar-benar terpenuhi untuk nilai 80.1. Semua kondisi if dalam program ini menggunakan operator perbandingan yang ketat (==) sehingga hanya akan benar jika nilai nam persis sama dengan nilai yang ditentukan. Padahal, berdasarkan tabel penilaian, nilai 80.1 seharusnya masuk dalam kategori A karena lebih besar dari 80.

B. Kesalahan program tersebut

Penggunaan operator perbandingan: Penggunaan operator == yang terlalu ketat menyebabkan banyak kasus tidak tertangani

Struktur kontrol: Struktur if bersarang yang terlalu banyak membuat kode kurang efisien dan sulit dibaca

Alur programnya:

Input nilai

Pemeriksaan kondisi

Penentuan nilai huruf

Output

C.

```
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 {
```

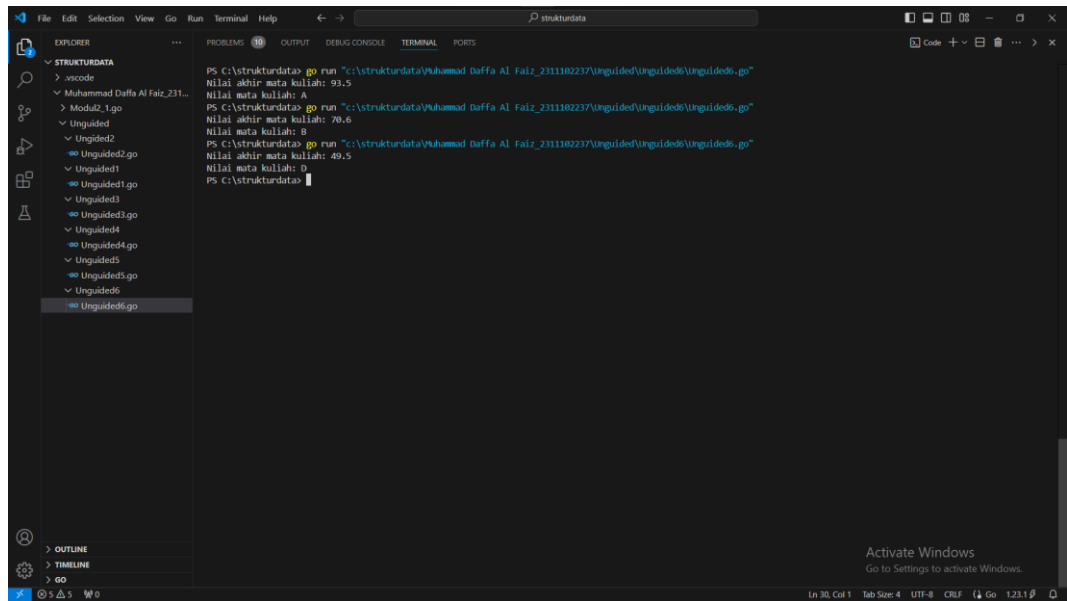
```

        nmk = "E"
    }

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

```

## Screenshot



## Deskripsi

Program ini adalah untuk mengonversi nilai numerik akhir dari suatu mata kuliah menjadi nilai huruf.

## Unguided 7

### 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)
        }
    }
}

```

```

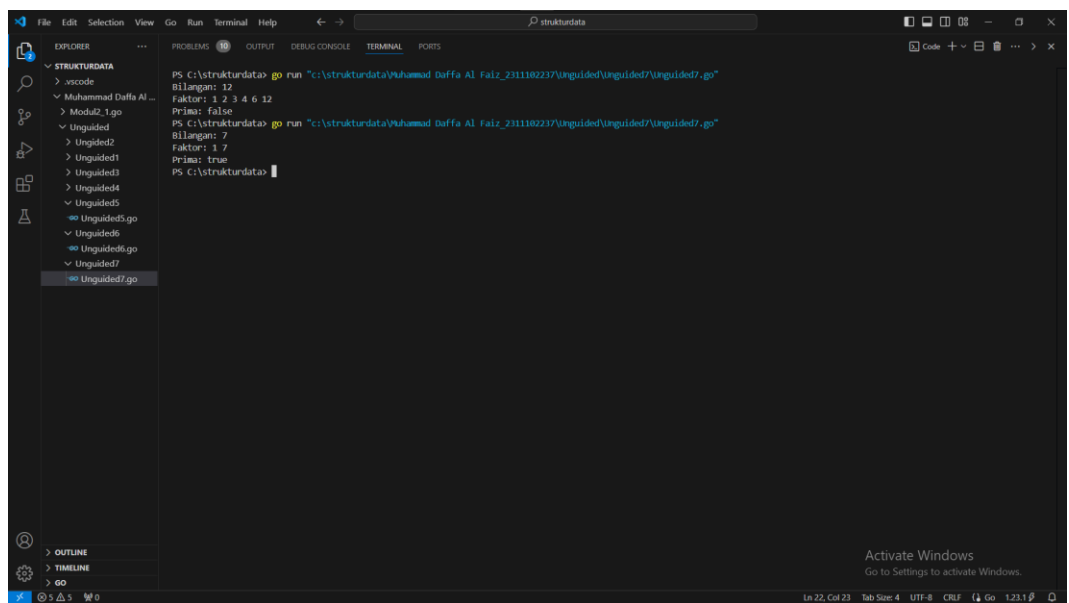
    }

    var isPrime bool
    if bil <= 1 {
        isPrime = false
    } else {
        isPrime = true
        for i := 2; i*i <= bil; i++ {
            if bil%i == 0 {
                isPrime = false
                break
            }
        }
    }

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

```

## Screenshot



## Deskripsi

Program ini adalah untuk menentukan factor dan keutamaan suatu bilangan bulat tertentu