

**LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK**

**MODUL 2
REVIEW STRUKTUR KONTROL**



Oleh:

M. AZKA HERMAWAN

2311102230

IF – 11- 02

**S1 TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO**

2024

I. DASAR TEORI

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)
}
```

2. Koding, Kompilasi, dan Eksekusi Go

a. 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.
- Setiap satu program lengkap Co disimpan dalam satu folder tersendiri. Nama folder merupakan nama program tersebut, Karena itu secara prinsip, satu program Co dapat dipecah dalam beberapa file dengan ekstensi ^go selama disimpan dalam folder yang sama.

b. 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.

c. 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 garis bawah.

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	

d. 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

e. 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 ifelse 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	}

II. GUIDED

1. Guided 1

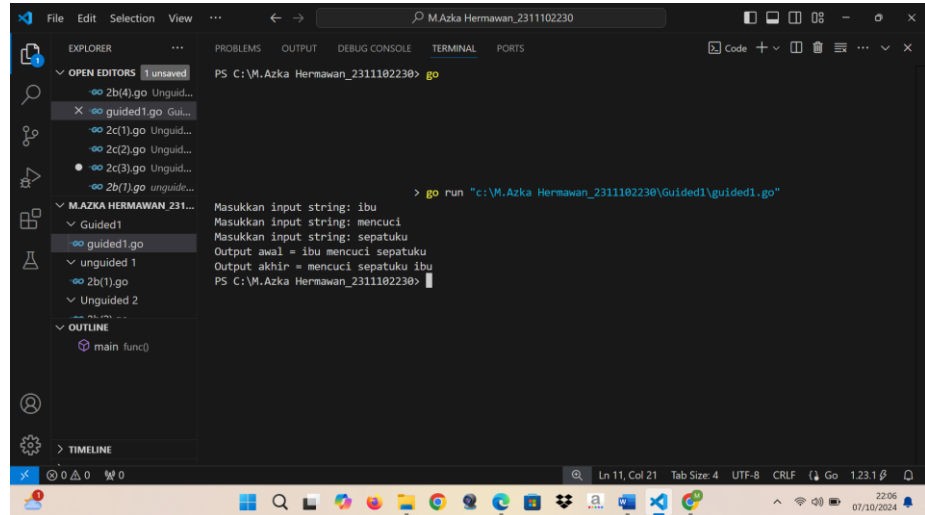
Source Code

```
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



```
File Edit Selection View ... PS C:\M.Azka Hermawan_2311102230> go run "c:\M.Azka Hermawan_2311102230\Guided1\guided1.go"
Masukkan input string: ibu
Masukkan input string: mencuci
Masukkan input string: sepatuku
Output awal = ibu mencuci sepatuku
Output akhir = mencuci sepatuku ibu
PS C:\M.Azka Hermawan_2311102230>
```

Deskripsi

Program ini berfungsi untuk menukar urutan tiga string input yang diberikan oleh pengguna.

2. Guided 2

Source Code

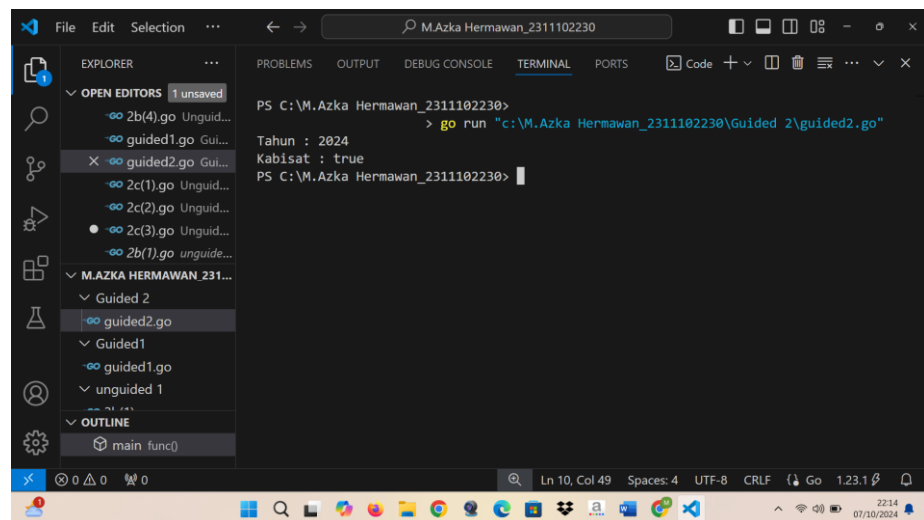
```
package main

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 meminta pengguna untuk memasukkan suatu tahun, kemudian menentukan apakah tahun tersebut adalah tahun kabisat atau bukan. Tahun kabisat adalah tahun yang habis dibagi 4, kecuali jika tahun tersebut habis dibagi 100 tetapi tidak habis dibagi 400. Hasil akhirnya ditampilkan sebagai nilai boolean (true atau false) yang menandakan apakah tahun tersebut kabisat (true) atau tidak (false).

3. Guided 3

Source Code

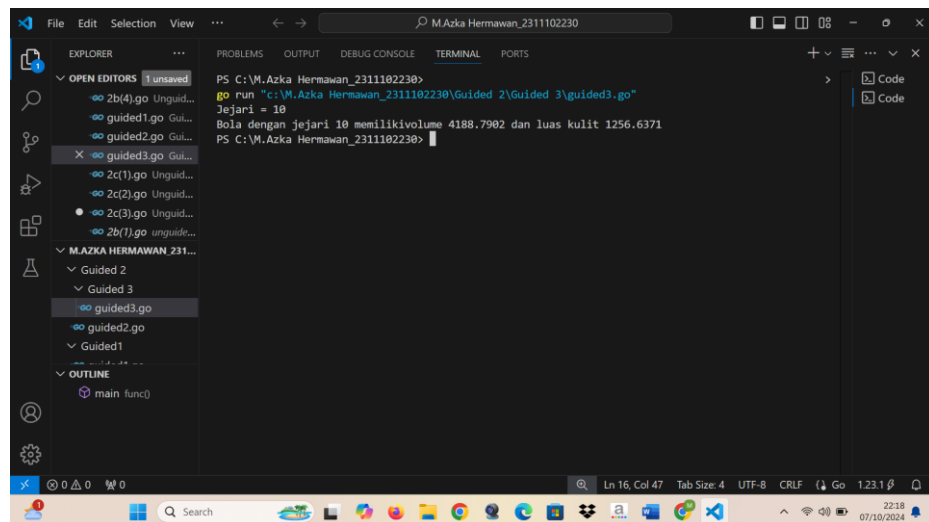
```
package main

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 memilikivolume %.4f
dan luas kulit %.4f \n", jari, volume, luasKulit)
}
```

Screenshot



Deskripsi

Program ini menghitung volume dan luas permukaan sebuah bola berdasarkan input jejari (radius) yang diberikan oleh pengguna.

Program meminta pengguna memasukkan nilai jejari, kemudian menampilkan hasil perhitungan volume dan luas permukaan dengan format empat angka di belakang koma.

III. UNGUIDED

1. Unguided 1 (2B nomor 1)

Source Code

```
package main

import "fmt"

func main() {

    warna := [4]string{"merah", "kuning", "hijau", "ungu"}

    berhasil := true

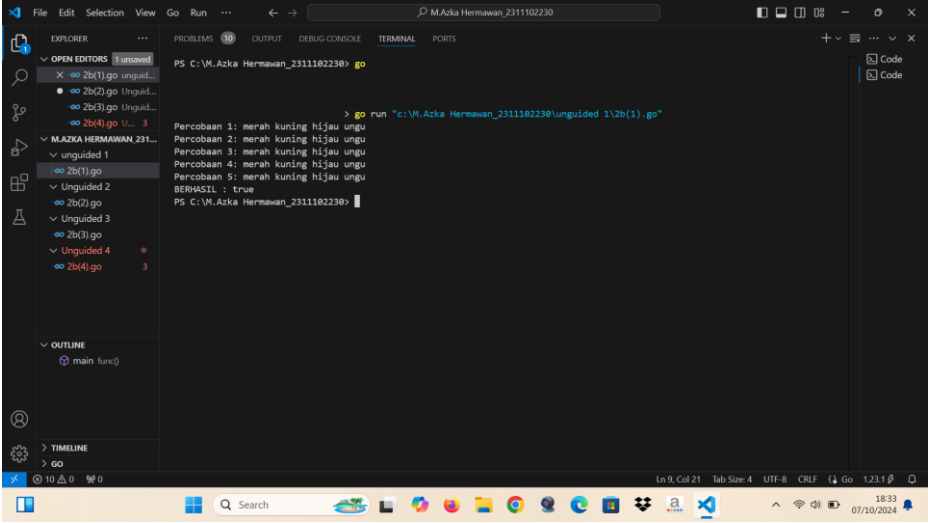
    for i := 1; i <= 5; i++ {
        var c1, c2, c3, c4 string
        fmt.Printf("Percobaan %d: ", i)

        _, err := fmt.Scan(&c1, &c2, &c3, &c4)
        if err != nil {
            berhasil = false
            break
        }

        if c1 != warna[0] || c2 != warna[1] || c3 !=
warna[2] || c4 != warna[3] {
            berhasil = false
            break
        }
    }

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

Screenshot



```
PS C:\M.Azka Hermawan_2311102230> go run "c:\M.Azka Hermawan_2311102230\unguided 1\2b(1).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:\M.Azka Hermawan_2311102230>
```

Deskripsi

Program ini dibuat untuk memverifikasi apakah user dapat memasukkan empat warna yang benar secara berurutan sebanyak lima kali. Program melakukan inisialisasi variable *warna* dengan isi array (*merah, kuning, hijau, ungu*). Jika user berhasil memasukkan semua warna dengan benar dalam semua percobaan, program akan menampilkan bahwa proses berhasil. Sebaliknya, jika terdapat kesalahan dalam input pada salah satu percobaan, program akan menghentikan proses dan menampilkan bahwa proses tidak berhasil.

2. Unguided 2 (2B nomor 2)

Source Code

```
package main

import "fmt"

func main() {
    var N int
    fmt.Print("N: ")
    _, err := fmt.Scan(&N)
    if err != nil {
        fmt.Println("Terjadi kesalahan saat membaca input.")
        return
    }

    if N <= 0 {
        fmt.Println("Bilangan N harus positif dan lebih dari nol.")
        return
    }

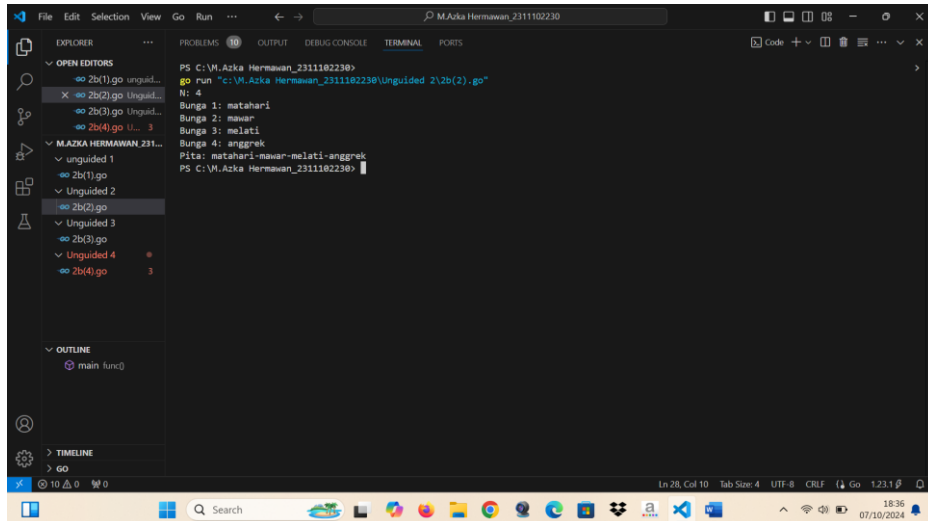
    pita := ""

    for i := 1; i <= N; i++ {
        var bunga string
        fmt.Printf("Bunga %d: ", i)
        _, err := fmt.Scan(&bunga)
        if err != nil {
            fmt.Println("Terjadi kesalahan saat membaca input bunga.")
            return
        }

        if pita == "" {
            pita = bunga
        } else {
            pita += "-" + bunga
        }
    }

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

Screenshot



```
PS C:\M.Azka Hermawan_2311102230>
go run "c:\M.Azka Hermawan_2311102230\Unguided 2\2b(2).go"
N: 4
Bunga 1: matahari
Bunga 2: mawar
Bunga 3: melati
Bunga 4: anggrek
Pita: matahari-mawar-melati-anggrek
PS C:\M.Azka Hermawan_2311102230>
```

Deskripsi

Program ini merupakan program yang meminta pengguna untuk memasukkan jumlah (n) bilangan positif. Kemudian program meminta pengguna untuk mengisi nama-nama bunga sejumlah (n) yang telah pengguna inputkan sebelumnya. Setelah itu program akan menampilkan output berupa nama-nama bunga yang telah dimasukkan dan akan digabungkan menjadi satu string dengan pemisah -, dan hasil akhirnya ditampilkan kepada pengguna.

3. Unguided 3 (2B nomor 3)

Source Code

```
package main

import "fmt"

func main() {
    var kantong1, kantong2 float64
    var totalBerat float64

    for {

        fmt.Print("Masukan berat belanjaan di kedua
kantong: ")
        _, err := fmt.Scan(&kantong1, &kantong2)

        if err != nil {
            fmt.Println("Input tidak valid, silakan coba
lagi.")

            var discard string
            fmt.Scanln(&discard)
            continue
        }

        if kantong1 < 0 || kantong2 < 0 {
            fmt.Println("Proses selesai.")
            break
        }

        var selisih float64
        if kantong1 > kantong2 {
            selisih = kantong1 - kantong2
        } else {
            selisih = kantong2 - kantong1
        }

        if selisih >= 9 {
            fmt.Println("Sepeda motor pak Andi akan oleng:
true")
        } else {
            fmt.Println("Sepeda motor pak Andi akan oleng:
false")
        }
    }
}
```

```

    }

    totalBerat += kantong1 + kantong2

    if totalBerat > 150 {
        fmt.Println("Proses selesai.")
        break
    }
}
}

```

Screenshot

```

PS C:\M.Azka Hermawan_2311102230>
go run "c:\M.Azka Hermawan_2311102230\Unguided 3\2b(3).go"
Masukan berat belanjaan di kedua kantong: 55 57
Sepeda motor pak Andi akan oleng: false
Masukan berat belanjaan di kedua kantong: 10 20
Sepeda motor pak Andi akan oleng: true
Masukan berat belanjaan di kedua kantong: 

```

Deskripsi

Program ini merupakan program untuk menghitung selisih pada dua kantong yang berada pada motor. Program meminta pengguna untuk memasukkan berat belanjaan di dua kantong secara berulang. Jika input tidak valid, program akan memberikan peringatan dan meminta input ulang. Jika selisihnya 9 kg atau lebih, program akan menampilkan pesan bahwa sepeda motor Pak Andi akan oleng. Jika selisih kurang dari 9 kg, program akan menampilkan pesan bahwa sepeda motor tidak akan oleng.

4. Unguided 4 (2B nomor 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 {
    result := 1.0
    for k := 0; k < K; k++ {
        result *= f(k)
    }
    return result
}

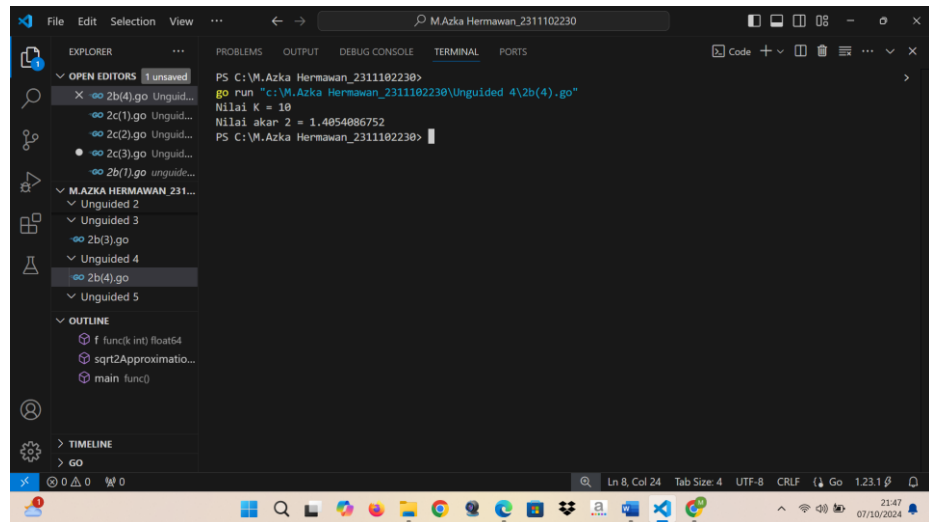
func main() {
    var K int

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

    approx := sqrt2Approximation(K)

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


Screenshot



The screenshot shows a Go IDE with the following content:

```
PS C:\M.Azka Hermawan_2311102230>
go run "C:\M.Azka Hermawan_2311102230\Unguided 4\2b(4).go"
Nilai K = 10
Nilai akar 2 = 1.4054086752
PS C:\M.Azka Hermawan_2311102230>
```

The Explorer panel on the left shows the project structure:

- OPEN EDITORS (1 unsaved)
- 2b(4).go Unguid...
- 2c(1).go Unguid...
- 2c(2).go Unguid...
- 2c(3).go Unguid...
- 2b(1).go Unguid...
- M.AZKA HERMAWAN_231...
- Unguided 2
- Unguided 3
- 2b(3).go
- Unguided 4
- 2b(4).go
- Unguided 5
- OUTLINE
- f funcik int) float64
- sqrt2Approximatio...
- main func()
- TIMELINE
- go

The status bar at the bottom indicates: Ln 8, Col 24, Tab Size: 4, UTF-8, CRLF, 1.23.1, 21:47, 07/10/2024.

Deskripsi

Program di atas digunakan untuk menghitung perkiraan nilai akar 2 menggunakan metode aproksimasi berbasis produk dari fungsi tertentu. Program ini bertujuan untuk menghitung nilai perkiraan akar 2 dengan pendekatan numerik yang semakin akurat seiring dengan bertambahnya nilai K.

5. Unguided 5 (2C nomor 1)

Source Code

```
package main

import "fmt"

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

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

    beratKg = beratParsel / 1000
    sisaGram = beratParsel % 1000

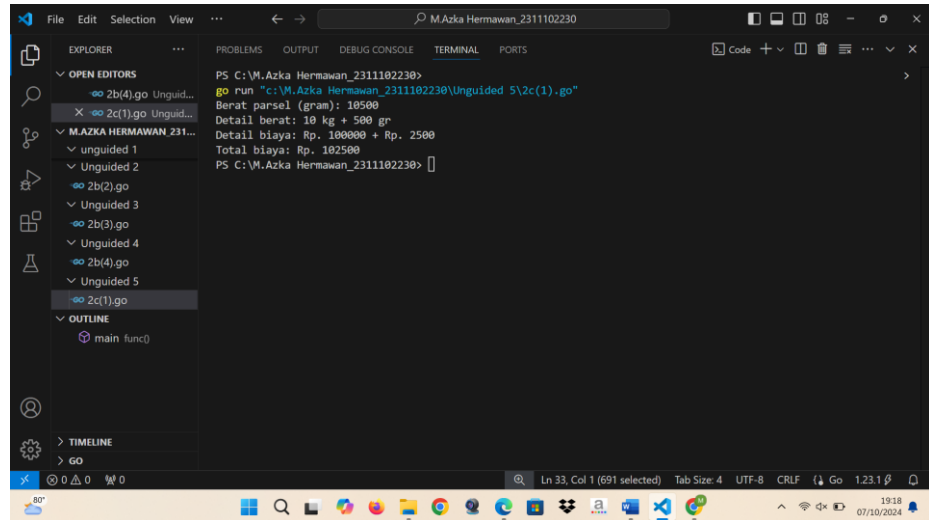
    biayaKg = beratKg * 10000

    if beratKg > 10 {
        biayaTambahan = 0
    } else {
        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:\M.Azka Hermawan_2311102230>
go run "c:\M.Azka Hermawan_2311102230\Unguided 5\2c(1).go"
Berat parcel (gram): 10500
Detail berat: 10 kg + 500 gr
Detail biaya: Rp. 100000 + Rp. 2500
Total biaya: Rp. 102500
PS C:\M.Azka Hermawan_2311102230>
```

Deskripsi

Program diatas merupakan program yang berfungsi untuk menghitung biaya pengiriman oleh PT POS.

Pertama-tama program meminta user untuk menginputkan berat parcel dalam gram. Setelah itu berat parcel dikonversi ke kilogram dan gram sisa dengan cara membagi berat total dengan 1000, biaya dasar dihitung berdasarkan berat kilogram, di mana setiap kilogram dikenakan biaya Rp. 10.000.

Jika beratnya kurang dari atau sama dengan 10 kg, biaya tambahan dihitung dari sisa gram:

- Jika sisa gram ≥ 500 , biaya tambahan dihitung dengan tarif Rp. 5 per gram.
- Jika sisa gram < 500 , biaya tambahan dihitung dengan tarif Rp. 15 per gram.

Program kemudian menampilkan rincian berat dan biaya yang terdiri dari biaya dasar dan biaya tambahan, serta total biaya pengiriman.

6. Unguided 6 (2C nomor 2)

Source Code Program Sebelum diperbaiki

```
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)
}
```

Soal & Jawaban

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

Jawaban: Program eror, dikarenakan program tersebut tidak konsisten, terutama pada variabel `nam` dideklarasikan sebagai

float64, namun di dalam blok if, program mencoba mengassign string (seperti "A", "AB", dll.) ke nam.

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

Jawaban:

Kesalahan program:

- Program tersebut tidak konsisten, terutama pada variabel nam dideklarasikan sebagai float64, namun di dalam blok if, program mencoba mengassign string (seperti "A", "AB", dll.) ke nam.
- Semua kondisi if adalah independen, sehingga jika nam lebih besar dari beberapa batas, beberapa blok if akan dieksekusi secara berurutan, dan nilai akhir nmk akan ditimpa beberapa kali.
- Sebagai contoh, jika nam = 80.1, maka semua kondisi berikut akan terpenuhi:
nam > 80 → nmk = "A"
nam > 72.5 → nmk = "AB"
nam > 65 → nmk = "B"
nam > 57.5 → nmk = "BC"
nam > 50 → nmk = "C"
nam > 40 → nmk = "D"

Akhirnya, nilai nmk akan menjadi "D", yang jelas tidak sesuai dengan spesifikasi yang diharapkan.

Perbaiki Program:

- Gantilah nam = "A" dengan nmk = "A", dan seterusnya untuk semua kondisi.
- Gunakan else if untuk memastikan hanya satu kondisi yang terpenuhi dan mencegah nilai nmk ditimpa oleh kondisi berikutnya.

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

Jawaban:

Program setelah diperbaiki:

```
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)
}

```

Perubahan yang dilakukan:

Batas nilai untuk "A" diubah menjadi ≥ 85 .

Batas nilai untuk "AB" diubah menjadi ≥ 75 .

Batas nilai untuk "BC" diubah menjadi ≥ 55 .

Dengan perbaikan ini, kita bisa menguji dengan masukan 93.5, 70.6, dan 49.5 maka output yang diperoleh akan sesuai.

Input: 93.5 -> Output: A.

Input: 70.6 -> Output: B.

Input: 49.5 -> Output: D.

7. Unguided 7 (2C nomor 3)

Source Code

```
package main

import "fmt"

func main() {
    var azka int

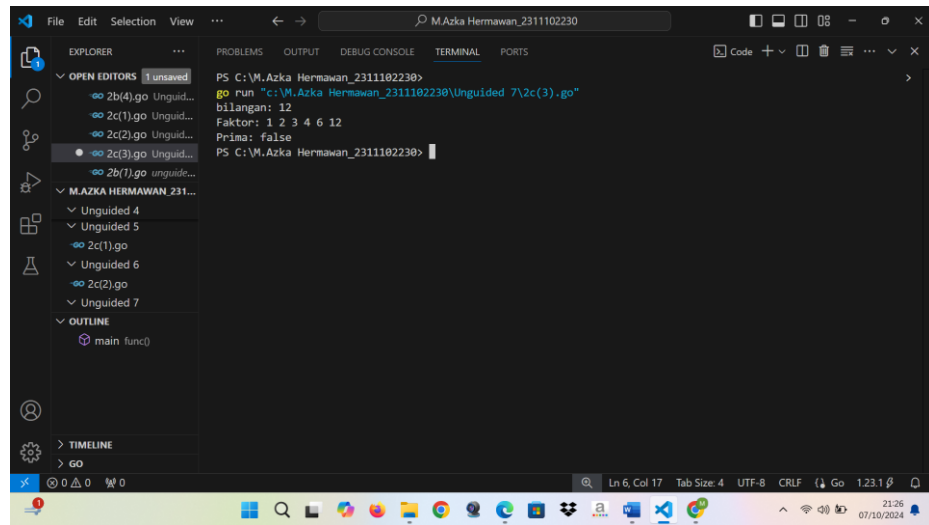
    fmt.Print("bilangan: ")
    fmt.Scanln(&azka)

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

    var jumlahFaktor int = 0
    for i := 1; i <= azka; i++ {
        if azka%i == 0 {
            jumlahFaktor++
        }
    }

    if jumlahFaktor == 2 {
        fmt.Println("Prima: true")
    } else {
        fmt.Println("Prima: false")
    }
}
```

Screenshot



The screenshot shows a Go IDE interface with a dark theme. The Explorer panel on the left shows a project named 'MAJKA HERMAWAN_231...' with several unguided files. The main editor displays a Go program that calculates the prime factors of the number 12. The output in the terminal shows the factors 1, 2, 3, 4, 6, and 12, and indicates that 12 is not a prime number.

```
PS C:\M.Azka Hermawan_2311102230>  
go run "c:\M.Azka Hermawan_2311102230\Unguided 7\2c(3).go"  
bilangan: 12  
Faktor: 1 2 3 4 6 12  
Prima: false  
PS C:\M.Azka Hermawan_2311102230>
```

Deskripsi

Program di atas bertujuan untuk mengecek faktor-faktor dari sebuah bilangan yang diinputkan pengguna dan menentukan apakah bilangan tersebut adalah bilangan prima atau bukan.