LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK

MODUL II

REVIEW STRUKTUR KONTROL

Oleh:

NAMA : KARTIKA PRINGGO HUTOMO

NIM : 2311102196

KELAS ; IF-11-02

S1 TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2024

I. DASAR TEORI

Struktur Program Go

Struktur program bahasa pemrograman Go (Golang) terdiri dari berbagai elemen yang mengontrol urutan eksekusi kode.

Memahami struktur kendali adalah kunci untuk menulis program yang efisien dan mudah dibaca.

Ada tiga struktur kontrol utama di Go: cabang, loop, dan pernyataan switch.

Tipe Data Dan Instruksi Dasar

Tipe data dalam pemrograman adalah klasifikasi yang menentukan tipe nilai dan operasi yang dapat dilakukan pada data.

Tipe data dasar meliputi Integer, Float, Char, String, dan Boolean.

Selain itu, ada tipe data komposit seperti array, struktur, dan kelas yang digunakan untuk menyimpan kumpulan dan struktur data yang lebih kompleks.

Pernyataan dasar dalam pemrograman meliputi pernyataan if untuk mengeksekusi kode berdasarkan suatu kondisi, pernyataan for untuk mengulang kode sebanyak N kali, dan pernyataan switch untuk memilih salah satu dari beberapa blok kode berdasarkan nilai ekspresi yang disertakan.

Memahami dasar-dasar teori tipe data dan kursus dasar ini sangat penting ketika mempelajari pemrograman, karena akan membantu Anda menulis kode yang lebih efisien dan efektif.

Struktur Kontrol Perulangan

Struktur data loop adalah konsep dasar dalam pemrograman yang memungkinkan Anda menjalankan serangkaian instruksi berulang kali.

Loop memungkinkan pemrogram mengulangi tugas tertentu tanpa menulis kode yang sama berulang kali, sehingga meningkatkan efisiensi dan mengurangi redundansi kode.

Ada berbagai jenis loop yang umum digunakan, antara lain: B.

perulangan for, perulangan while, dan perulangan do-sementara.

Setiap jenis memiliki properti yang berbeda dan kegunaan yang berbeda tergantung pada kebutuhan spesifik program Anda.

Perulangan biasanya terdiri dari kondisi yang menentukan kapan harus keluar dari perulangan, dan blok kode yang berjalan selama kondisi tersebut terpenuhi.

Penggunaan struktur data loop yang tepat dapat meningkatkan kinerja program secara signifikan, terutama untuk kumpulan data besar atau operasi yang harus sering diulang.

Struktur Data Percabangan

Struktur data loop adalah konsep dasar dalam pemrograman yang memungkinkan Anda menjalankan serangkaian instruksi berulang kali.

Loop memungkinkan pemrogram mengulangi tugas tertentu tanpa menulis kode yang sama berulang kali, sehingga meningkatkan efisiensi dan mengurangi redundansi kode.

Ada berbagai jenis loop yang umum digunakan, antara lain: B.

perulangan for, perulangan while, dan perulangan do-sementara.

Setiap jenis memiliki properti yang berbeda dan kegunaan yang berbeda tergantung pada kebutuhan spesifik program Anda.

Perulangan biasanya terdiri dari kondisi yang menentukan kapan harus keluar dari perulangan, dan blok kode yang berjalan selama kondisi tersebut terpenuhi.

Penggunaan struktur data loop yang tepat dapat meningkatkan kinerja program secara signifikan, terutama untuk kumpulan data besar atau operasi yang harus sering diulang.

II. GUIDED

Guided 1

Source code

```go
package main


import "fmt"
```

```go
func main() {

    var (

        satu, dua, tiga string

        temp            string

    )

    fmt.Print("Masukan input string: ")

    fmt.Scanln(&satu)
```

```
    fmt.Print("Masukan input string: ")

    fmt.Scanln(&dua)

    fmt.Print("Masukan input string: ")

    fmt.Scanln(&tiga)

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

    temp = satu

    satu = dua

    dua = tiga

    tiga = temp

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

}
```

Output :

```
PS D:\Alpro 2\Laprak modul2> go run "d:\Alpro 2\Laprak modul2\Guided1.go"
Masukan input string: 123
Masukan input string: 112
Masukan input string: 124
Output awal = 123 112 124
Output akhir = 112 124123
PS D:\Alpro 2\Laprak modul2>
```

Penjelasan

Program di atas meminta pengguna untuk tiga input string, menukar nilai string satu per satu, dan menampilkan hasilnya sebelum dan sesudah pertukaran.

Variabel "temp" digunakan untuk menyimpan nilai sementara selama proses pertukaran.

Ada kesalahan kecil di bagian keluaran akhir dimana seharusnya ada spasi di antara variabel "dua" dan "tiga".

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)))

}
```

Output

```
PS D:\Alpro 2\Laprak modul2> go run "d:\Alpro 2\Laprak modul2\Guided2.go"
Tahun : 2008
Kabisat : true
PS D:\Alpro 2\Laprak modul2>
```

Penjelasan

Program di atas merupakan program sederhana dalam bahasa Go yang digunakan untuk menentukan apakah suatu tahun merupakan tahun kabisat atau tidak. Pengguna diminta untuk memasukkan sebuah angka tahun, kemudian program akan mengevaluasi apakah tahun tersebut memenuhi aturan kabisat.

Guided 3

Source code

```
package main



import (

    "fmt"

    "math"

)
```

```
func main() {
```

```
    var jari, volume, luasKulit float64

    fmt.Print(" Masukan 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)

}
```

Output :

```
PS D:\Alpro 2\Laprak modul2> go run "d:\Alpro 2\Laprak modul2\Guided3.go"
 Masukan Jejari = 12
Bola dengan jejari 12 memiliki volume 7238.2295 dan luas kulit 1809.5574
PS D:\Alpro 2\Laprak modul2>
```

III. UNGUIDED

2B.1

Source code

```
package main



import (

    "bufio"

    "fmt"

    "os"

    "strings"

)
```

```
func main() {
```

```
    correctOrder := []string{"merah", "kuning", "hijau", "ungu"}
```

```
    reader := bufio.NewReader(os.Stdin)

    success := false
```

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

        input, _ := reader.ReadString('\n')
        input = strings.TrimSpace(input)

        colors := strings.Split(input, " ")

        attemptSuccess := true // flag for the current attempt

        if len(colors) == 4 { // make sure input has 4 colors
            for j := 0; j < 4; j++ {
                if colors[j] != correctOrder[j] {
                    attemptSuccess = false
                    break
                }
            }
        } else {
            attemptSuccess = false
        }

        if attemptSuccess {
            success = true
        } else {
            success = false
        }
    }

    if success {
```

```
        fmt.Println("BERHASIL : true")

    } else {

        fmt.Println("BERHASIL : false")

    }

}
```

Output :

```
PS D:\Alpro 2\Laprak modul2> go run "d:\Alpro 2\Laprak modul2\Unguided2b_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 D:\Alpro 2\Laprak modul2>
```

Penjelasan

Program di atas meminta pengguna menebak urutan warna "merah", "kuning", "hijau", dan "ungu" sebanyak lima kali.

Pengguna diminta memasukkan empat warna dan pesanan diperiksa untuk melihat apakah cocok.

Jika pengguna dapat menebak urutan yang benar pada upaya apa pun, program akan mengembalikan "SUCCESS: true".

Jika tidak, program akan mengembalikan "SUCCESS: false".

Namun, variabel "sukses" direset setelah setiap upaya, sehingga hasil akhirnya salah.

2B.2

Source code

```
package main



import (

    "fmt"

    "strings"

)
```

```
func main() {

    var N int
    fmt.Print("Masukkan jumlah bunga (N): ")
    fmt.Scan(&N)

    var pita strings.Builder
    var bunga string
    count := 0

    for i := 1; i <= N; i++ {
        fmt.Printf("Bunga %d: ", i)
        fmt.Scan(&bunga)

        if strings.ToUpper(bunga) == "SELESAI" {
            break
        }

        if pita.Len() > 0 {
            pita.WriteString(" - ")
        }

        pita.WriteString(bunga)
        count++
    }


    fmt.Printf("Pita: %s\n", pita.String())
    fmt.Printf("Bunga: %d\n", count)
```

```
    }
```

Output :

```
PS D:\Alpro 2\Laprak modul2> go run "d:\Alpro 2\Laprak modul2\Unguided2b_2.go"
Masukkan jumlah bunga (N): 4
Bunga 1: t
Bunga 2: k
Bunga 3: u
Bunga 4: SELESAI
Pita: t - k - u
Bunga: 3
PS D:\Alpro 2\Laprak modul2>
```

Penjelasan

Program di atas adalah aplikasi sederhana yang menanyakan nama bunga kepada pengguna hingga angka tertentu ('N') tercapai atau pengguna memasukkan 'DONE'.

Program ini menggunakan ``strings.Builder" untuk menyimpan nama bunga dalam format yang rapi dengan pembatas ``-" di antara setiap bunga.

Proses dimulai dengan meminta pengguna memasukkan jumlah bunga yang ingin dimasukkan.

Program kemudian meminta pengguna untuk memasukkan nama bunga satu demi satu dalam satu lingkaran.

Perulangan berhenti ketika pengguna mengetik "FINISH" (semua huruf besar).

Nama setiap bunga yang sah ditambahkan ke "pita" dan jumlah bunga yang sah dihitung.

Terakhir, program mencetak daftar bunga yang dimasukkan beserta nomornya.

2B.3

Source code

```
package main


import "fmt"
```

```
func main() {

    const (
```

```
        maxSelisihBerat = 9
        maxTotalBerat   = 150
    )

    var beratKiri, beratKanan float64
    var oleng bool

    for {
        fmt.Print("Masukkan berat belanjaan di kedua kantong: ")
        fmt.Scan(&beratKiri, &beratKanan)

        if beratKiri < 0 || beratKanan < 0 {
            fmt.Println("Berat tidak boleh negatif.")
            break
        }

        if beratKiri+beratKanan > maxTotalBerat {
            fmt.Println("Total berat melebihi batas.")
            break
        }

        oleng = beratKiri-beratKanan >= maxSelisihBerat || beratKanan-beratKiri >=
maxSelisihBerat
        fmt.Printf("Sepeda motor pak Andi akan oleng: %t\n", oleng)
    }

    fmt.Println("Proses selesai.")
}
```

Output :

```
PS D:\Alpro 2\Laprak modul2> go run "d:\Alpro 2\Laprak modul2\Unguided2b_3.go"
Masukkan berat belanjaan di kedua kantong: 40 1.4
Sepeda motor pak Andi akan oleng: true
Masukkan berat belanjaan di kedua kantong:
```

Penjelasan

Program di atas meminta pengguna memasukkan berat dua kantong makanan dan melakukan pengecekan saldo.

Jika salah satu bobotnya negatif, program akan memberi tahu Anda bahwa bobotnya tidak boleh negatif dan menghentikan prosesnya.

Program juga membatalkan pemrosesan jika berat total melebihi 150 kg.

Selanjutnya, program memeriksa apakah perbedaan berat kedua tas lebih dari 9 kg.

Jika iya, maka program akan menunjukkan bahwa sepeda motor akan “berbelok”.

Proses ini berlanjut hingga salah satu kondisi terpenuhi.

Kemudian program akan menampilkan "Pemrosesan selesai".

2B.4

Source Code

```go
package main



import (

    "fmt"

    "math"

)
```

```go
// Fungsi untuk menghitung nilai f(k)

func hitungF(k int) float64 {

    // Menghitung pembilang dan penyebut dari rumus f(k)

    pembilang := math.Pow(float64(4*k+2), 2)
```

```
    penyebut := float64((4*k+1) * (4*k+3))

    return pembilang / penyebut

}
```

```
// Fungsi untuk menghitung perkalian f(i) hingga k untuk mendekati akar 2

func hitungAkarDua(k int) float64 {

    hasil := 1.0

    for i := 0; i <= k; i++ {

        hasil *= hitungF(i)

    }

    return hasil

}
```

```
func main() {

    // Input nilai k

    var k int

    fmt.Print("Masukkan nilai K: ")

    fmt.Scan(&k)
```

```
    // Menghitung nilai f(k)

    nilaiF := hitungF(k)

    fmt.Printf("Nilai f(%d) = %.10f\n", k, nilaiF)
```

```
    // Menghitung nilai aproksimasi akar 2 untuk k

    akarDua := hitungAkarDua(k)

    fmt.Printf("Nilai akar 2 untuk K = %d: %.10f\n", k, akarDua)

}
```

Output :

```
PS D:\Alpro 2\Laprak modul2> go run "d:\Alpro 2\Laprak modul2\Unguided2b_4.go"
Masukkan nilai K: 1
Nilai f(1) = 1.0285714286
Nilai akar 2 untuk K = 1: 1.3714285714
PS D:\Alpro 2\Laprak modul2>
```

Penjelasan

Program di atas menghitung nilai fungsi matematika tertentu dan menggunakan fungsi ini untuk memperkirakan nilai akar 2.

Program ini terdiri dari dua fungsi utama.

'calculateF(k)' yang menghitung nilai \(f(k) \ ) berdasarkan ekspresi pembilang dan penyebut, dan 'calculateRootTwo(k)' yang menghitung fungsi berikut: Perkirakan akar 2 dengan mengalikan nilai nol \(f(i) \) dengan \(k \).

Pada fungsi "utama", pengguna diminta memasukkan nilai \(k \).

Program kemudian menghitung nilai \(f(k) \) dan perkiraan nilai akar 2 dan menampilkannya menggunakan hasil fungsi \(f(i) \).

Hasilnya ditampilkan dalam format desimal hingga 10 digit setelah koma desimal.

2C.1

Source code

```
package main



import "fmt"
```

```
func hitungBiaya(beratGram int) (int, int, int) {



    beratKg := beratGram / 1000

    sisaGram := beratGram % 1000
```

```
    biayaKg := beratKg * 10000
```

```
    var biayaSisa int

    if sisaGram >= 500 {
```

```
        biayaSisa = sisaGram * 5
    } else if sisaGram > 0 && beratKg <= 10 {
        biayaSisa = sisaGram * 15
    } else {
        biayaSisa = 0
    }
```

```
    return biayaKg, biayaSisa, beratKg
}
```

```
func main() {
    var beratGram int
    fmt.Print("Masukkan berat parsel (dalam gram): ")
    fmt.Scanln(&beratGram)
```

```
    biayaKg, biayaSisa, beratKg := hitungBiaya(beratGram)
```

```
    fmt.Printf("Total berat: %d kg %d gram\n", beratKg, beratGram%1000)
    fmt.Printf("Biaya pengiriman: Rp. %d\n", biayaKg+biayaSisa)
    fmt.Printf("  - Biaya per kg: Rp. %d\n", biayaKg)
    fmt.Printf("  - Biaya sisa gram: Rp. %d\n", biayaSisa)
}
```

Output :

```
PS D:\Alpro 2\Laprak modul2> go run "d:\Alpro 2\Laprak modul2\Unguided2c_1.go"
Masukkan berat parsel (dalam gram): 400
Total berat: 0 kg 400 gram
Biaya pengiriman: Rp. 6000
  - Biaya per kg: Rp. 0
  - Biaya sisa gram: Rp. 6000
PS D:\Alpro 2\Laprak modul2>
```

Penjelasan

Program ini menghitung biaya pengiriman paket berdasarkan beratnya dalam gram.

Fungsi "calculateCost(weightGrams)" mengubah berat menjadi kilogram, menghitung biaya total berat (Rp.

10.

000 per kg) dan menentukan biayanya.

Sisa gram dalam dua kondisi: Rp per gram jika sisa 500 gram atau lebih, atau Rp 15 per gram jika sisa 0 > dan berat 10 kg atau kurang.

Pada fungsi "utama", pengguna diminta memasukkan berat paket, dan program menampilkan berat total dan biaya pengiriman, yang terdiri dari biaya per kilogram dan sisa biaya gram.

2C.2

Source code

```go
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)
}
```

Output

```
PS D:\Alpro 2\Laprak modul2> go run "d:\Alpro 2\Laprak modul2\Unguided2c_2.go"
Nilai akhir mata kuliah: 89
Nilai mata kuliah: A
PS D:\Alpro 2\Laprak modul2>
```

Penjelasan

Program ini menghitung biaya pengiriman paket berdasarkan beratnya dalam gram.

Fungsi "calculateCost(weightGrams)" mengubah berat menjadi kilogram, menghitung biaya total berat (Rp.

10.

000 per kg) dan menentukan biayanya.

Sisa gram dalam dua kondisi: Rp per gram jika sisa 500 gram atau lebih, atau Rp 15 per gram jika sisa 0 > dan berat 10 kg atau kurang.

Pada fungsi "utama", pengguna diminta memasukkan berat paket, dan program menampilkan berat total dan biaya pengiriman, yang terdiri dari biaya per kilogram dan sisa biaya gram.

2C.3

Source code

```go
package main

import "fmt"

func main() {

    var bil int

    var prima bool = true

    fmt.Print("Bilangan: ")

    fmt.Scan(&bil)

    fmt.Print("Faktor: ")

    for i := 1; i <= bil; i++ {

        if bil%i == 0 {

            fmt.Print(i, " ")

            if !(i == 1 || i == bil) {

                prima = false

            }

        }

    }

    fmt.Println()

    fmt.Print("Prima: ", prima)

}
```

Output :

```
PS D:\Alpro 2\Laprak modul2> go run "d:\Alpro 2\Laprak modul2\Unguided2c_3.go"
Bilangan: 35
Faktor: 1 5 7 35
Prima: false
PS D:\Alpro 2\Laprak modul2>
```

Penjelasan

Program ini meminta pengguna untuk memasukkan bilangan bulat dan mencetak semua faktor dari bilangan tersebut.

Program ini juga memeriksa apakah suatu bilangan prima dengan memeriksa apakah hanya ada dua elemen: 1 dan bilangan itu sendiri.

Jika unsur lain ditemukan, program akan menandai bilangan tersebut sebagai bukan bilangan prima.

Terakhir, program mencetak semua faktor dan bilangan prima dari bilangan masukan.