LAPORAN PRAKTIKUM
ALGORITMA PEMROGRAMAN 2
MODUL II
REVIEW STRUKTUR KONTROL

Disusun Oleh:
Fadilah Salehah / 2311102164

S1 IF 11 05

DOSEN PENGAMPU
Arif Amrulloh,S.Kom.,M.Kom.

PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2024

MODUL II

I. Dasar Teori

1. Struktur Kontrol Perulansgan

   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 perulangan di go dapat dilihat gambar dibawah ini

```
for inisialisasi; kondisi; update {
    // .. for-loop ala C
    // .. ke-3 bagian opsional, tetapi ";" tetap harus ada
}
```

```
for kondisi {
    // .. ulangi kode di sini selama kondisi terpenuhi
    // .. sama seperti "for ; kondisi; {"
}
```

```
for {
    // .. tanpa kondisi, berarti loop tanpa henti (perlu if-break)
}
```

```
for ndx, var := range slice/array {
    // .. iterator mengunjungi seluruh isi slice/array
    // .. pada setiap iterasi ndx diset indeks dan var diisi nilainya
}
```

   Dalam konsep pemrograman terstruktur, setiap rancangan algoritma harus memenuhi syarat satu pintu masuk dan satu pintu keluar. Karena itu tidaklah diperkenankan untuk membuat program sumber yang mempunyai struktur loop yang mempunyai pintu keluar lebih dari satu, seperti:

   - Satu pintu keluar dari kondisi for dan satu lagi dari instruksi If-break
   - Atau mempunyai instruksi If-break yang lebih dari satu

2. Struktur Kontrol Percabangan

   Untuk analisa kasus, bahasa Go mendukung dua bentuk percabangan, yaitu if-else dan switch-case. 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. Dalam bahasa Go 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 (atau alias dari) susunan suatu if-else if-..-else-endif.

II. Guided

- Guided 1

Source Code:

```go
package main

import (
    "bufio"
    "fmt"
    "os"
    "strings"
)

func main() {
    correctOrder := []string{"merah", "kuning", "hijau", "ungu"}
    reader := bufio.NewReader(os.Stdin)
    success := true
    for i := 1; i <= 5; i++ {
        fmt.Printf("Percobaan %d: ", i)
        input, _ := reader.ReadString('\n')
        input = strings.TrimSpace(input)
        colors := strings.Split(input, " ")
        for j := 0; j < 4; j++ {
            if colors[j] != correctOrder[j] {
                success = false
                break
            }
        }
        if !success {
            break
        }
    }
    if success {
        fmt.Println("BERHASIL : true")
    } else {
        fmt.Println("BERHASIL : false")
    }
}
```

Penjelasan:

Kode di atas meminta pengguna untuk memasukkan urutan warna sebanyak lima kali, lalu memeriksa apakah urutan yang dimasukkan sama dengan urutan yang benar, yaitu `["merah", "kuning", "hijau", "ungu"]`. Pada setiap percobaan, input pengguna dibaca, dipisahkan berdasarkan spasi, dan dibandingkan dengan urutan yang benar. Jika pengguna memasukkan urutan yang salah pada percobaan mana pun, program akan

menghentikan pengecekan dan mencetak "BERHASIL : false". Jika semua urutan benar selama lima percobaan, program akan mencetak "BERHASIL : true".

Output:

```
PS C:\Users\USER\Documents\PRAKALPRO2> go run "c:\Users\USER
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:\Users\USER\Documents\PRAKALPRO2>
```

```
PS C:\Users\USER\Documents\PRAKALPRO2> go run "c:\Users\USER\Documents\PRAKALPRO2\guided\guided1.go"
Percobaan 1: merah kuning hijau ungu
Percobaan 2: merah kuning hjau ungu
BERHASIL : false
```

- Guided 2

Source Code:

```
package main

import "fmt"

func main() {
    var a, b, c, d, e int
    var hasil int
    fmt.Scanln(&a, &b, &c, &d, &e)
    hasil = a + b + c + d + e
    fmt.Println("Hasil Penjumlahan ", a, b, c, d, e, "adalah = ",
hasil)
}
```

Penjelasan:

Kode di atas meminta input dari pengguna berupa lima angka bertipe integer (disimpan dalam variabel `a`, `b`, `c`, `d`, dan `e`). Setelah semua input diterima, program menghitung jumlah kelima angka tersebut dan menyimpannya dalam variabel `hasil`. Kemudian, program mencetak hasil penjumlahan dengan format "Hasil Penjumlahan [a, b, c, d, e] adalah = [hasil]".

Output:

```
PS C:\Users\USER\Documents\PRAKALPRO2> go run "c:\Use
10 39 20 39 29
Hasil Penjumlahan  10 39 20 39 29 adalah =  137
PS C:\Users\USER\Documents\PRAKALPRO2>
```

- Guided 3

Source Code:

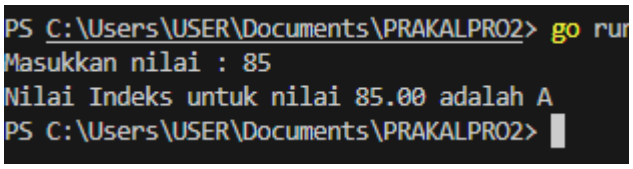
```go
package main

import "fmt"

func main() {
    var nam float32
    var nmk string
    fmt.Print("Masukkan nilai : ")
    fmt.Scan(&nam)
    if nam > 80 {
        nmk = "A"
    } else if nam > 72.5 {
        nmk = "B"
    } else if nam > 65 {
        nmk = "C"
    } else if nam > 50 {
        nmk = "D"
    } else if nam > 40 {
        nmk = "E"
    } else {
        nmk = "F"
    }
    fmt.Printf("Nilai Indeks untuk nilai %.2f adalah %s\n", nam, nmk)
}
```

Penjelasan:

Kode di atas meminta pengguna untuk memasukkan nilai numerik bertipe `float32`, yang disimpan dalam variabel `nam`. Berdasarkan nilai yang dimasukkan, program menentukan nilai huruf (indeks) sesuai dengan kondisi if-else. Jika nilai lebih besar dari 80, maka indeksnya adalah "A", dan seterusnya, hingga nilai yang kurang dari atau sama dengan 40 diberikan indeks "F". Setelah menentukan indeks yang sesuai,

program mencetak hasilnya dengan format "Nilai Indeks untuk nilai [nilai] adalah [indeks]", menampilkan nilai yang dimasukkan dan indeksnya.

Output:



III. Unguided

1. Suatu pita (string) berisi kumpulan nama-nama bunga yang dipisahkan oleh spasi dan '-', contoh pita diilustrasikan seperti berikut ini. Pita: mawar - melati - tulip-teratal - kamboja-anggrek. Buatlah sebuah program yang menerima input sebuah bilangan bulat positif (dan tidak nol) N, kemudian program akan meminta input berupa nama bunga secara berulang sebanyak N kali dan nama tersebut disimpan ke dalam pita. (Petunjuk: gunakan operasi penggabungan string dengan operator "+"). Tampilkan isi pita setelah proses input selesai. Perhatikan contoh sesi interaksi program seperti di bawah ini (teks bergaris bawah adalah Input/read):

   Source Code:

```go
package main

import (
    "bufio"
    "fmt"
    "os"
    "strings"
)

func main() {
    var jumlahBunga int
    var pita []string
    scanner := bufio.NewScanner(os.Stdin)

    fmt.Print("N: ")
    fmt.Scanln(&jumlahBunga)

    for i := 1; i <= jumlahBunga; i++ {
        fmt.Printf("Bunga %d: ", i)
        scanner.Scan()
        bunga := scanner.Text()
        pita = append(pita, bunga)
    }

    result := strings.Join(pita, " - ")
    fmt.Printf("Pita: %s\n", result)
}
```

   Penjelasan:

Kode di atas meminta pengguna untuk memasukkan jumlah bunga (`jumlahBunga`), kemudian pengguna diminta memasukkan nama bunga satu per satu sebanyak jumlah yang dimasukkan. Nama bunga yang diinputkan disimpan dalam slice `pita`. Setelah semua bunga dimasukkan, program menggabungkan nama-nama bunga tersebut menjadi satu string dengan pemisah " - " menggunakan fungsi `strings.Join()`, dan kemudian mencetak hasilnya dengan format "Pita: [nama bunga yang digabungkan]".

Output:

```
PS C:\Users\USER\Documents\PRAKALPRO2>
N: 3
Bunga 1: Pita
Bunga 2: Mawar
Bunga 3: Kembang
Pita: Pita - Mawar - Kembang
```

2. Setiap hari Pak Andi membawa banyak barang belanjaan dari pasar dengan mengendarai sepeda motor. Barang belanjaan tersebut dibawa dalam kantong terpal di kiri-kanan motor. Sepeda motor tidak akan oleng jika selisih berat barang di kedua kantong sisi tidak lebih dari 9 kg. Buatlah program Pak Andi yang menerima input dua buah bilangan real 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. Perhatikan contoh sesi interaksi program seperti di bawah ini (teks bergaris bawah adalah input/read):

```
Masukan berat belanjaan di kedua kantong: 5.5 1.0
Masukan berat belanjaan di kedua kantong: 7.1 8.5
Masukan berat belanjaan di kedua kantong: 2 6
Masukan berat belanjaan di kedua kantong: 9 5.8
Proses selesai.
```

Source Code:

```go
package main

import (
    "fmt"
    "math"
)

func main() {
    var beratKantong1, beratKantong2 float64
```

```
    const batasTotalBerat = 150.0

    for {
        fmt.Print("Masukkan berat belanjaan di kedua kantong: ")
        fmt.Scan(&beratKantong1, &beratKantong2)

        if beratKantong1 < 0 || beratKantong2 < 0 {
            fmt.Println("Proses selesai.")
            break
        }

        totalBerat := beratKantong1 + beratKantong2
        if totalBerat > batasTotalBerat {
            fmt.Println("Proses selesai.")
            break
        }

        selisih := math.Abs(beratKantong1 - beratKantong2)

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

Penjelasan:

Kode di atas meminta pengguna memasukkan berat dua kantong belanjaan, lalu mengevaluasi beberapa kondisi dalam loop. Pertama, jika berat salah satu kantong bernilai negatif, program berhenti dengan pesan "Proses selesai." Kedua, jika total berat kedua kantong melebihi batas 150.0, program juga berhenti. Jika kedua kondisi tersebut tidak terpenuhi, program menghitung selisih berat antara kedua kantong. Jika selisihnya lebih dari atau sama dengan 9, program menampilkan pesan bahwa sepeda motor akan oleng (tidak seimbang), jika tidak, motor tidak akan oleng. Program terus meminta input hingga salah satu kondisi penghentian terpenuhi.

Output:

```
PS C:\Users\USER\Documents\PRAKALPRO2> go run "c:\Users\
Masukkan berat belanjaan di kedua kantong: 5.5 10.5
Sepeda motor Pak Andi akan oleng: false
Masukkan berat belanjaan di kedua kantong: 55.5 70.2
Sepeda motor Pak Andi akan oleng: true
Masukkan berat belanjaan di kedua kantong: 72.3 66.9
Sepeda motor Pak Andi akan oleng: false
Masukkan berat belanjaan di kedua kantong: 44.5 55.4
Sepeda motor Pak Andi akan oleng: true
Masukkan berat belanjaan di kedua kantong: 59.8 97.7
Proses selesai.
```

3. Buatlah sebuah program yang menerima input sebuah bilangan sebagai K, kemudian menghitung dan menampilkan nilai f(K) sesuai persamaan di atas. Perhatikan contoh sesi interaksi program seperti di bawah ini (teks bergaris bawah adalah input/read):

$$f(k) = \frac{(4k+2)^2}{(4k+1)(4k+3)}$$

Source Code:

```go
package main

import (
    "fmt"
)

func main() {
    var K int

    fmt.Print("Masukkan nilai K: ")
    fmt.Scanln(&K)

    prod := 1.0
    for k := 0; k <= K; k++ {
        atas := (4*float64(k) + 2) * (4*float64(k) + 2)
        bawah := (4*float64(k) + 1) * (4*float64(k) + 3)
        prod *= atas / bawah
    }

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

Penjelasan:

Kode di atas menghitung nilai perkiraan dari akar 2 menggunakan rumus produk yang melibatkan parameter `K`, yang diminta pengguna untuk diinput. Program mulai dengan mendeklarasikan variabel `prod` yang diinisialisasi dengan 1.0. Dalam loop dari 0 hingga `K`, program menghitung nilai atas sebagai \((4k + 2)^2\) dan nilai bawah sebagai \((4k + 1)(4k + 3)\). Nilai produk diperbarui dengan hasil pembagian atas dan bawah untuk setiap iterasi. Setelah loop selesai, program mencetak nilai akhir dari `prod` sebagai perkiraan akar 2 dengan format desimal hingga sepuluh angka di belakang koma.

Output:

```
PS C:\Users\USER\Documents\PRAKALPRO2> g
Masukkan nilai K: 100
Nilai akar 2 = 1.4133387072
PS C:\Users\USER\Documents\PRAKALPRO2>
```

4. PT POS membutuhkan aplikasi perhitungan biaya kirim berdasarkan berat parsel. Maka, buatlah program BiayaPos untuk menghitung biaya pengiriman tersebut dengan ketentuan sebagai berikut! Dari berat parsel (dalam gram), harus dihitung total berat dalam kg dan sisanya (dalam gram). Biaya jasa pengiriman adalah Rp. 10.000,- per kg. Jika sisa berat tidak kurang dari 500 gram, maka tambahan biaya kirim hanya Rp. 5,- per gram saja. Tetapi jika kurang dari 500 gram, maka tambahan biaya akan dibebankan sebesar Rp. 15,- per gram. Sisa berat (yang kurang dari 1kg) digratiskan biayanya apabila total berat ternyata lebih dari 10kg. Perhatikan contoh sesi interaksi program seperti di bawah ini (teks bergaris bawah adalah input/read):

1	Contoh #1 Berat parsel (gram): **8500** Detail berat: 8 kg + 500 gr Detail biaya: Rp. 80000 + Rp. 2500 Total biaya: Rp. 82500
2	Contoh #2 Berat parsel (gram): **9250** Detail berat: 9 kg + 250 gr Detail biaya: Rp. 90000 + Rp. 3750 Total biaya: Rp. 93750
3	Contoh #3 Berat parsel (gram): **11750** Detail berat: 11 kg + 750 gr Detail biaya: Rp. 110000 + Rp. 3750 Total biaya: Rp. 110000

Source Code:

```go
package main

import (
    "fmt"
)

func main() {
    var beratParselInGram, totalBiaya float64
```

```
    var beratKg, beratGram int

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

    beratKg = int(beratParselInGram) / 1000
    beratGram = int(beratParselInGram) % 1000

    totalBiaya = float64(beratKg) * 10000

    if beratGram >= 500 {
        totalBiaya += 2500
    } else if beratGram > 0 {
        totalBiaya += float64(beratGram) * 15
    }

    if beratKg >= 10 {
        totalBiaya = 100000
    }

    fmt.Printf("\nBerat parsel (gram): %.0f\n", beratParselInGram)
    fmt.Printf("Detail berat: %d kg + %d gram\n", beratKg, beratGram)
    fmt.Printf("Total biaya: Rp. %.0f\n", totalBiaya)
}
```

Penjelasan:
Pertama, program meminta input berat parsel, kemudian mengonversinya menjadi kilogram dan gram. Biaya dihitung dengan mengalikan jumlah kilogram dengan tarif per kilogram (Rp. 10.000). Jika berat gram 500 atau lebih, ditambahkan biaya tambahan Rp. 2.500, atau jika lebih dari 0 gram, ditambahkan biaya Rp. 15 per gram. Jika berat mencapai 10 kg atau lebih, total biaya diatur menjadi Rp. 100.000. Terakhir, program menampilkan berat parsel, rincian berat, dan total biaya kepada pengguna.

Output:

```
Masukkan berat parsel (dalam gram): 8500

Berat parsel (gram): 8500
Detail berat: 8 kg + 500 gram
Total biaya: Rp. 82500
```

5. Buatlah program yang menerima input sebuah bilangan bulat b dan b > 1. Program harus dapat mencari dan menampilkan semua faktor dari bilangan tersebut! Sebuah bilangan bulat b memiliki faktor bilangan f > 0 jika f habis membagi b. Contoh: 2 merupakan faktor dari bilangan 6 karena 6 habis dibagi 2. Buatlah program yang menerima input sebuah bilangan bulat b dan b > 1. Program harus dapat mencari dan menampilkan semua faktor dari bilangan tersebut! Perhatikan contoh sesi interaksi program seperti di bawah ini (teks bergaris bawah adalah input/read)

Bilangan: 12 Faktor: 1 2 3 4 6 12	Bilangan: 7 Faktor: 1 7

Source Code:

```go
package main

import "fmt"

func hitungFaktorDanPrima(bil int) (bool, []int) {
    var faktor []int
    prima := true

    for i := 1; i <= bil; i++ {
        if bil%i == 0 {
            faktor = append(faktor, i)
            if i != 1 && i != bil {
                prima = false
            }
        }
    }
    return prima, faktor
}

func main() {
    var bil1, bil2 int

    fmt.Print("Bilangan 1: ")
    fmt.Scanln(&bil1)

    prima1, faktor1 := hitungFaktorDanPrima(bil1)
    fmt.Print("Faktor: ")
    for _, f := range faktor1 {
        fmt.Print(f, " ")
    }
    fmt.Println()
    fmt.Println("Prima 1:", prima1)

    fmt.Print("Bilangan 2: ")
    fmt.Scanln(&bil2)

    prima2, faktor2 := hitungFaktorDanPrima(bil2)
    fmt.Print("Faktor: ")
    for _, f := range faktor2 {
        fmt.Print(f, " ")
    }
    fmt.Println()

    if bil1 == 1 {
        prima1 = false
    }
    if bil2 == 1 {
        prima2 = false
```

```
    }

    fmt.Println("Prima 2:", prima2)
}
```

Penjelasan:
Fungsi hitungFaktorDanPrima digunakan untuk menghitung faktor dan status prima; ia mengembalikan slice berisi faktor dan boolean yang menunjukkan apakah bilangan itu prima. Program ini mencetak faktor dari kedua bilangan setelah input, dan juga menampilkan status prima, dengan pengecualian untuk bilangan 1 yang selalu dianggap bukan prima. Kode ini terstruktur dengan baik, memisahkan logika penghitungan dari bagian utama program untuk meningkatkan keterbacaan dan pemeliharaan.

Output:

```
Bilangan 1: 12
Faktor: 1 2 3 4 6 12
Prima 1: false
Bilangan 2: 7
Faktor: 1 7
Prima 2: true
```