

LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2
MODUL 2
“REVIEW STRUKTUR KONTROL”



DISUSUN OLEH:
SHEILA STEPHANIE ANINDYA
103112400086
S1 IF-12-01
S1 TEKNIK INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

I. DASAR TEORI

A. Pengertian Struktur Kontrol

Struktur control merupakan perintah dengan bentuk (struktur) tertentu yang digunakan untuk mengatur (mengontrol) jalannya program. Terdapat 3 bentuk struktur kontrol, antara lain :

1) Struktur Kontrol Sekuensial

Struktur kontrol sekuensial merupakan proses algoritma secara beruntunan atau proses secara bersambung-sambungan (sequence) terhadap satu atau lebih instruksi, yang berarti bahwa :

- a) tiap instruksi dikerjakan satu persatu
- b) tiap instruksi dilaksanakan tepat sekali, tidak instruksi yang diulang
- c) urutan instruksi yang dilaksanakan pemroses sama dengan urutan aksi sebagaimana yang tertulis di dalam teks algoritmanya
- d) akhir dari instruksi terakhir merupakan akhir dari algoritma program juga.

2) Struktur Kontrol Percabangan

Struktur control percabangan memungkinkan alur algoritma program dijalankan berdasarkan sebuah kondisi tertentu dimana kondisi itu menghasilkan pilihan True /False. Dalam hal ini, perintah dalam suatu kondisi dapat dijalankan saat kondisi tersebut memiliki nilai benar. Jika bernilai sebaliknya, maka perintah pemrograman tidak dapat dijalankan.

Secara umum, struktur percabangan pemrograman yang sering digunakan antara lain :

- a) If-else: Struktur kontrol percabangan yang paling sederhana.
- b) Switch-case: Struktur kontrol percabangan yang sering digunakan untuk mengevaluasi jika banyak kondisi yang berbeda.
- c) Conditional operator: Struktur kontrol percabangan yang lebih singkat dan lebih fleksibel dibandingkan dengan if-then-else.

Cara kerja struktur kontrol percabangan:

- 1. Kondisi dievaluasi.
- 2. Jika kondisi bernilai benar, maka perintah dalam kondisi tersebut akan dijalankan.

3. Jika kondisi bernilai salah, maka perintah dalam kondisi tersebut tidak akan dijalankan.

3) Struktur Kontrol Perulangan

Struktur kontrol perulangan adalah proses mengulang-ulang eksekusi blok kode tanpa henti, selama kondisi yang dijadikan acuan terpenuhi. Biasanya disiapkan variabel untuk iterasi atau variabel penanda kapan perulangan akan diberhentikan.

Beberapa jenis perulangan yang umum digunakan adalah:

a) For Loop: Digunakan jika jumlah perulangan diketahui sebelumnya.

b) While Loop: Digunakan jika perulangan dilakukan berdasarkan kondisi yang harus tetap bernilai benar.

c) Do-While Loop: Mirip dengan while, tetapi akan dijalankan setidaknya sekali sebelum kondisi dicek.

B. Tujuan Struktur Kontrol

Struktur kontrol memungkinkan programmer untuk menentukan urutan eksekusi pernyataan program . Struktur kontrol ini memungkinkan pengguna melakukan dua hal: 1) melewati beberapa pernyataan saat mengeksekusi yang lain, dan 2) mengulang satu atau lebih pernyataan saat beberapa kondisi benar.

II. GUIDED

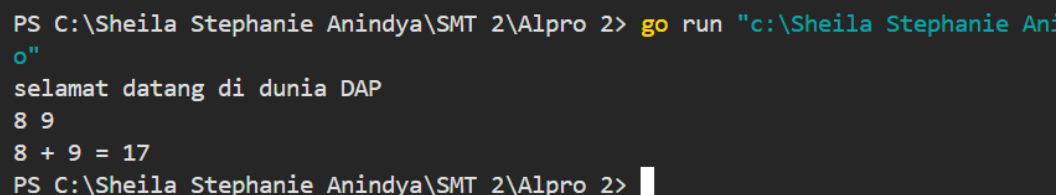
1. Source Code:

```
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)
}
```

Output :



```
PS C:\Sheila Stephanie Anindya\SMT 2\Alpro 2> go run "c:\Sheila Stephanie Anindya\SMT 2\Alpro 2\main.go"
selamat datang di dunia DAP
8 9
8 + 9 = 17
PS C:\Sheila Stephanie Anindya\SMT 2\Alpro 2>
```

Penjelasan :

Program menyimpan tulisan "selamat datang di dunia DAP" ke dalam variabel greetings, lalu menampilkannya di terminal. Lalu program meminta menginput 2 angka, dan program menjumlahkannya.

2. Source Code:

```
package main

import "fmt"

func main() {
    for i := 1; i <= 5; i++ {
        fmt.Println("Iterasi ke-", i)
    }
}
```

Output:

```
PS C:\Sheila Stephanie Anindya\SMT 2\Alpro 2> go run "c:\Sheila Stephanie Anindya\
Iterasi ke- 1
Iterasi ke- 2
Iterasi ke- 3
Iterasi ke- 4
Iterasi ke- 5
PS C:\Sheila Stephanie Anindya\SMT 2\Alpro 2>
```

Penjelasan :

Program memakai for loop dengan variabel i yang dimulai dari 1 dan bertambah 1 di setiap iterasi, sampai 5. Di setiap loop, program mencetak "Iterasi ke-" diikuti dengan nilai i. Jika sudah lebih dari 5, loop berhenti dan program selesai.

3. Source Code:

```
package main

import "fmt"

func main() {
    nilai := 80
    pctHadir := 0.75
```

```

adaTubes := true

var indeks string

if nilai > 75 && adaTubes {
    indeks = "A"
} else if nilai > 65 {
    indeks = "B"
} else if nilai > 50 && pctHadir > 0.7 {
    indeks = "C"
} else {
    indeks = "F"
}

fmt.Printf("Nilai %d dengan kehadiran %.2f%% dan buat tubes = %t, mendapat indeks %s\n", nilai, pctHadir*100, adaTubes, indeks)
}

```

Output:

```

PS C:\Sheila Stephanie Anindya\SMT 2\Alpro 2> go run "c:\Sheila Stephanie Anindya\
Nilai 80 dengan kehadiran 75.00% dan buat tubes = true, mendapat indeks A
PS C:\Sheila Stephanie Anindya\SMT 2\Alpro 2>

```

Penjelasan :

Program ini berfungsi untuk menentukan indeks nilai berdasarkan tiga faktor: nilai ujian, persentase kehadiran, dan keberadaan tugas besar (tubes). Pertama, program menyatakan variable nilai ujiannya langsung 80, persentase kehadirannya 0.75, dan menyatakan apakah ada tugas besar atau tidak, kalau true berarti ada. Program menggunakan if-else untuk mengevaluasi kondisi. Jika nilai lebih dari 75 dan tugas besar tersedia, maka indeks yang diberikan adalah "A". Jika nilai lebih dari 65, maka indeksnya "B". Jika nilai lebih dari 50 dan persentase kehadiran lebih dari 70%, maka indeksnya "C". Jika tidak memenuhi kriteria tersebut, maka indeks yang diberikan adalah "F".

III. UNGUIDED

1. Source Code:

```
package main

import "fmt"

func main() {
    var tahun int
    fmt.Scan(&tahun)

    if tahun%4 == 0 && tahun%100 != 0 {
        fmt.Println(true)
    } else if tahun%400 == 0 {
        fmt.Println(true)
    } else {
        fmt.Println(false)
    }
}
```

Output:

```
PS C:\Sheila Stephanie Anindya\SMT 2\Alpro 2\code> go run "c:\Sheila Stephanie Anindya\
2016
true
PS C:\Sheila Stephanie Anindya\SMT 2\Alpro 2\code> go run "c:\Sheila Stephanie Anindya\
2019
false
PS C:\Sheila Stephanie Anindya\SMT 2\Alpro 2\code> █
```

Penjelasan :

Program ini mengecek apakah tahun yang diinput kabisat atau bukan. Jika tahun bisa dibagi 4 tapi tidak habis dibagi 100, atau bisa dibagi 400, itu kabisat(true). Kalau bukan kabisat hasilnya false.

2. Source Code:

```
package main

import "fmt"

func main() {
    var i, k int
    var fk float64
    fmt.Print("Nilai K : ")
    fmt.Scan(&k)

    for i = k; i <= k; i++ {
        pembilang := (4*i + 2) * (4*i + 2)
        penyebut := (4*i + 1) * (4*i + 3)
        fk = float64(pembilang) / float64(penyebut)
    }

    fmt.Printf("Nilai f(K) = %.10f\n", fk)
}
```

Output:

```
PS C:\Sheila Stephanie Anindya\SMT 2\Alpro 2\code> go run "c:\Sheila
Nilai K : 100
Nilai f(K) = 1.0000061880
PS C:\Sheila Stephanie Anindya\SMT 2\Alpro 2\code> █
```


Penjelasan :

Program ini menghitung $f(k) = \frac{(4k+2)^2}{(4k+1)(4k+3)}$ namun menggunakan struktur kontrol for Di dalam perulangan, program menghitung pembilang dan penyebut, lalu membaginya untuk mendapatkan hasil. Agar hasilnya terdapat 10 angka di belakang koma, gunakan float64.

2. Source Code:

```
package main

import "fmt"

func main() {
    var k int
    var hasil float64
    fmt.Print("Nilai K : ")
    fmt.Scan(&k)

    for i := 0; i < k; i++ {
        pembilang := (4*i + 2) * (4*i + 2)
        penyebut := (4*i + 1) * (4*i + 3)

        if i == 0 {
            hasil = float64(pembilang) / float64(penyebut)
        } else {
            hasil = hasil * (float64(pembilang) / float64(penyebut))
        }
    }
    fmt.Printf("Nilai akar 2 = %.10f\n", hasil)
}
```

Output:

```
PS C:\Sheila Stephanie Anindya\SMT 2\Alpro 2\code> go run "c:\Sheila  
Nilai K : 1000  
Nilai akar 2 = 1.4141251768  
PS C:\Sheila Stephanie Anindya\SMT 2\Alpro 2\code> █
```

Penjelasan :

Program ini menghitung perkiraan nilai $\sqrt{2}$ namun dengan memodifikasi code sebelumnya. Karena menggunakan struktur kontrol for, untuk iterasi pertama, hasil dihitung sebagai $\frac{(4k+2)^2}{(4k+1)(4k+3)}$, lalu iterasi selanjutnya, hasil dikalikan dengan pecahan yang sama dari iterasi sebelumnya. Agar hasilnya terdapat 10 angka di belakang koma, gunakan float64.

3. Source Code:

```
package main

import "fmt"

func main() {

    var berat, totalBiaya, kirimKg, kirimGr int

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

    beratKg := berat / 1000
    sisaGr := berat % 1000
    kirimKg = beratKg * 10000

    if beratKg <= 10 {
        if sisaGr >= 500 {
            kirimGr = sisaGr * 5
        } else {
```

```

        kirimGr = sisaGr * 15
    }
}

totalBiaya = kirimKg + kirimGr

fmt.Printf("Detail berat: %d kg + %d gr\n", beratKg, sisaGr)
fmt.Printf("Detail biaya: Rp. %d + Rp. %d\n", kirimKg, kirimGr)
fmt.Printf("Total biaya: Rp. %d\n", totalBiaya)
}

```

Output:

```

PS C:\Sheila Stephanie Anindya\SMT 2\Alpro 2\code> go run "c:\Sheila Stephanie Anindya\
Berat parsel (gram) : 8500
Detail berat: 8 kg + 500 gr
Detail biaya: Rp. 80000 + Rp. 2500
Total biaya: Rp. 82500
PS C:\Sheila Stephanie Anindya\SMT 2\Alpro 2\code>

```

Penjelasan :

Program ini menghitung ongkir parsel berdasarkan berat dalam gram. Berat dikonversi ke kilogram dan sisa gram, lalu biaya dihitung dengan aturan: setiap kilogram dikenai Rp. 10.000, sedangkan sisa gram dikenai Rp. 5 per gram jika ≥ 500 gram atau Rp. 15 per gram jika < 500 gram, asalkan berat total tidak melebihi 10 kg. Output berupa detail berat, detail biaya, dan total biayanya.

IV. KESIMPULAN

Struktur kontrol adalah perintah dengan bentuk tertentu yang digunakan untuk mengatur jalannya program. Terdapat tiga jenis struktur kontrol utama:

1. **Struktur Kontrol Sekuensial**, di mana instruksi dijalankan secara berurutan tanpa ada pengulangan atau percabangan.
2. **Struktur Kontrol Percabangan**, yang memungkinkan program mengambil keputusan berdasarkan kondisi tertentu (contoh: if-else, switch-case).
3. **Struktur Kontrol Perulangan**, yang memungkinkan eksekusi kode berulang selama kondisi tertentu terpenuhi (contoh: for loop, while loop, do-while loop).

Tujuan utama struktur kontrol adalah untuk mengatur urutan eksekusi pernyataan dalam program, baik dengan melewati beberapa pernyataan maupun mengulang pernyataan tertentu sesuai kondisi yang ditentukan. Hal ini membuat program menjadi lebih fleksibel, efisien, dan mudah dikendalikan.

V. REFERENSI

1. Nguyen, P. (n.d.). *Raptor control structures*.
<https://home.csulb.edu/~pnguyen/cecs100/lecturenotes/raptorcontrolstructures.pdf>
2. Noval Agung. (n.d.). *Dasar pemrograman Golang*.
<https://dasarpemrogramangolang.novalagung.com/>
3. Penulis tidak dikenal. (n.d.). *MODUL 4: Sekuensial dan percabangan*.
<https://www.scribd.com/document/716851239/MODUL-4-Sekuensial-Dan-Percabangan>
4. Penulis tidak dikenal. (n.d.). *Struktur flow control algoritma*.
<https://www.studocu.id/id/document/universitas-negeri-padang/coding/4-struktur-flow-control-algoritma/32177131>
5. Penulis tidak dikenal. (n.d.). *Definisi struktur kontrol percabangan dalam pemrograman*.
Kumparan.
<https://kumparan.com/how-to-teknologi/definisi-struktur-kontrol-percabangan-dalam-pemrograman-1xlqcdCPug2/full>
6. Penulis tidak dikenal. (n.d.). *Pemrograman dasar*. Universitas Esa Unggul.
https://lms-paralel.esaunggul.ac.id/pluginfile.php?file=%2F79693%2Fmod_resource%2Fcontent%2F1%2FOL%205%20Pemrograman-Dasar.pdf