3 Perulangan, Percabangan (Seleksi Kondisi) pada Bahasa Pemrograman Go

Objektif :

- **Mahasiswa Diharapkan Mampu Mengetahui Dan Memahami Macam-Macam Struktur Perulangan Bahasa Pemrograman Go.**
- **Mahasiswa Diharapkan Mampu Mengetahui Dan Memahami Macam-Macam Struktur Percabangan (Struktur Kondisi) Bahasa Pemrograman Go.**

3.1 Perulangan pada Bahasa Pemrograman Go

Perulangan secara umum adalah proses mengulang-ulang eksekusi blok kode tanpa henti selama kondisi yang dijadikan acuan terpenuhi. Biasanya disiapkan sebuah variabel untuk melakukan iterasi (perulangan) atau sebuah variabel penanda kapan perulangan akan diberhentikan. Pada bahasa pemrograman Go, kata kunci perulangan hanya terdapat ***for*** saja, walaupun demikian kemampuannya merupakan gabungan *for* , *foreach* , dan *while* pada bahasa pemrograman lain.

Terdapat beberapa cara menggunakan statemen *for* standar, yaitu :

1. Memasukkan variabel *counter* perulangan beserta kondisinya setelah kata kunci *for*.
2. Menuliskan kondisi setelah kata kunci *for* (hanya kondisi). Konsepnya mirip seperti *while* seperti bahasa pemrograman lain.
3. Menuliskan kata kunci *for* tanpa argumen atau kondisi.
4. Perulangan dengan menggunakan kombinasi kata kunci *for* dan *range*.
5. Perulangan dengan menggunakan kombinasi kata kunci *break* dan *continue*.
6. Perulangan bersarang.
7. Perulangan dengan memanfaatkan teknik pemberian label.

3.1.1 Perulangan dengan memasukkan variabel *counter* perulangan.

Cara perulangan ini yaitu memasukkan variabel *counter* perulangan beserta kondisinya setelah kata kunci *for*. Bentuk umum ini hampir sama seperti bahasa pemrograman C/C++.

Bentuk umum :

```
For inisialisasi;kondisi;iterasi{
        statement
}
```

Contoh Program:

```
for i := 0; i < 5; i++ {
    fmt.Println("Angka", i)
}
```

Perulangan di atas hanya akan berjalan ketika variabel i bernilai dibawah 5, dengan ketentuan setiap kali perulangan, nilai variabel i akan dilakukan iterasi atau ditambahkan 1 (i++ artinya ditambah satu, sama seperti i = i + 1). Karena i pada awalnya bernilai 0, maka perulangan akan berlangsung sebanyak 5 kali, yaitu ketika i bernilai 0, 1, 2, 3, dan 4.

3.1.2 Perulangan *for* mirip seperti konsep *while.*

Konsep perulangan ini yaitu menuliskan kondisi setelah kata kunci *for* (hanya kondisi). Deklarasi dan iterasi variabel *counter* tidak dituliskan setelah kata kunci, tetapi hanya kondisi perulangan saja. Konsepnya mirip seperti *while* milik bahasa pemrograman lain.

Bentuk umum :

```
For kondisi {
        statement
        iterasi
}
```

Contoh Program:

```
var i = 0

for i < 5 {
    fmt.Println("Angka", i)
    i++
}
```

Pada program diatas, kata kunci *for* diberikan sebuah kondisi yaitu jika variabel i kurang dari 5, maka perulangan akan terus dilakukan yaitu diwakili oleh iterasi variabel i++. Karena i pada awalnya bernilai kurang dari 5 yaitu nol, maka perulangan akan berlangsung sebanyak 5 kali, yaitu ketika i bernilai 0, 1, 2, 3, dan 4.

3.1.3 Menuliskan kata kunci *for* tanpa argumen atau kondisi.

Cara berikut ini adalah perulangan dengan kata kunci *for* ditulis tanpa kondisi. Dengan ini akan dihasilkan perulangan tanpa henti (sama dengan *for* bernilai *true*). Pemberhentian perulangan dapat dilakukan dengan menggunakan bantuan kata kunci *break*.

Bentuk umum :

For {
    statement
    iterasi
    kondisi
    break
}

Contoh Program:

```
var i = 0

for {
    fmt.Println("Angka", i)

    i++
    if i == 5 {
        break
    }
}
```

Pada perulangan tanpa henti di atas, variabel i yang nilai awalnya nol akan dilakukan proses inkrementasi. Ketika nilai i sudah mencapai 5, kata kunci *break* digunakan untuk menghentikan perulangan.

3.1.4 Perulangan dengan menggunakan kombinasi kata kunci *for* dan *range*

Cara ini digunakan untuk melakukan perulangan data bertipe *array* yang jumlah itemnya sudah ditentukan.

Bentuk umum :

```
Var array = [ ]type{..,..,..}

For var, var := range var_array {
        statement
}
```

Contoh Program:

```
var fruits = [4]string{"apple", "grape", "banana", "melon"}

for i, fruit := range fruits {
    fmt.Printf("elemen %d : %s\n", i, fruit)
}
```

Pada program diatas, *array fruits* diambil elemennya secara berurutan. Nilai setiap elemen ditampung variabel yaitu variabel *fruit* (tanpa huruf s), sedangkan *indeks* nya ditampung pada variabel i sesuai jumlah dari item yang ditentukan sebelumnya yaitu sebanyak 4.

3.1.5 Perulangan dengan menggunakan kombinasi kata kunci *break* dan *continue.*

Kata kunci *break* digunakan untuk menghentikan secara paksa sebuah perulangan yang berjalan, sedangkan *continue* dipakai untuk memaksa program untuk lanjut ke perulangan berikutnya.

Bentuk umum :

```
For inisialisasi;kondisi;iterasi{
        kondisi1 {continue}
        kondisi2 {break}
        statement
}
```

Contoh Program:

```
for i := 1; i <= 10; i++ {
    if i % 2 == 1 {
        continue
    }

    if i > 8 {
        break
    }

    fmt.Println("Angka", i)
}
```

Berikut adalah penjelasan dari program diatas:

1. Dilakukan perulangan dimulai dari angka 1 hingga 10 dengan i sebagai variabel iterasinya.
2. Ketika i adalah bilangan ganjil (dapat diketahui dari i % 2 , jika hasilnya 1 , berarti ganjil), maka akan dipaksa lanjut ke perulangan berikutnya.
3. Ketika i lebih besar dari 8, maka perulangan akan berhenti.
4. Nilai i akan ditampilkan.

3.1.6 Perulangan bersarang.

Cara pengaplikasian perulangan bersarang pada pemrograman Go kurang lebih sama dengan perulangan bersarang pada bahasa pemrograman lainnya, yaitu dengan menulikan blok statement perulangan didalam perulangan.

Bentuk umum :

```
For inisialisasi;kondisi;iterasi{
        For inisialisasi;kondisi;iterasi
                { statement }
        statement
}
```

Contoh Program :

```
for i := 0; i < 5; i++ {
    for j := i; j < 5; j++ {
        fmt.Print(j, " ")
    }

    fmt.Println()
}
```

Pada kode program di atas, yang akan dieksekusi pertama kali yaitu statemen *for* terdalam dan fungsi *fmt.Println()* dipanggil tanpa disisipkan parameter. Cara seperti ini bisa digunakan untuk menampilkan baris baru. Kegunaannya sama seperti output dari statement *fmt.Print("\n")*.

3.1.7 Perulangan dengan memanfaatkan teknik pemberian label.

Pada salah satu kasus perulangan bersarang, *break* dan *continue* akan berlaku pada blok perulangan dimana *break* dan *continue* jika digunakan. Ada cara agar kedua kata kunci ini bisa tertuju pada perulangan terluar atau perulangan tertentu, yaitu dengan memanfaatkan teknik pemberian label.

Bentuk umum :

nama_label:

For *inisialisasi;kondisi;iterasi*{
    For *inisialisasi;kondisi;iterasi*{
        kondisi {
        break ***nama_label***
        }
    statement
    }
}

Contoh Program :

```
outerLoop:
for i := 0; i < 5; i++ {
    for j := 0; j < 5; j++ {
        if i == 3 {
            break outerLoop
        }
        fmt.Print("matriks [", i, "][", j, "]", "\n")
    }
}
```

Pada program diatas, sebelum kata kunci *for* terluar, terdapat baris kode nama label *outerLoop:*. Maksud dari kode tersebut adalah sengaja disiapkan untuk sebuah label yang bernama *outerLoop* untuk *for* dibawahnya. Penamaan label bisa saja diganti dengan nama lain, hanya harus diakhiri dengan tanda titik dua atau colon (:). Kemudian pada *for* bagian dalam, terdapat seleksi kondisi untuk pengecekan nilai i . Ketika nilai tersebut sama dengan 3, maka kata kunci *break* dipanggil dengan target perulangan yang diberikan label *outerLoop* , dan perulangan tersebut akan dihentikan.

3.2 Percabangan (Seleksi Kondisi) pada Bahasa Pemrograman Go

Seleksi kondisi digunakan untuk mengontrol alur program. Analoginya mirip seperti fungsi keran air pada sebuah rumah. Kapan air diperbolehkan keluar dan kapan harus berhenti yang diatur oleh keran tersebut. Sama seperti pada seleksi kondisi, kapan sebuah blok kode akan dieksekusi dan juga dikontrol. Nilai dijadikan acuan oleh seleksi kondisi adalah nilai bertipe *bool*, bisa saja berasal dari variabel, ataupun hasil operasi perbandingan. Nilai tersebut menentukan blok kode mana yang akan dieksekusi.

Pada bahasa pemrograman Go, terdapat 2 macam kata kunci untuk seleksi kondisi, yaitu *if else* dan *switch*. Pada bab ini kita akan mempelajarinya satu-persatu. Sebagai catatan, pada bahasa pemrograman Go tidak mendukung seleksi kondisi menggunakan *ternary*, seperti: *var data = (isExist ? "ada" : "tidak ada")* adalah *invalid* dan menghasilkan *error*.

3.2.1 Seleksi Kondisi Menggunakan Kata Kunci *if, else if,* dan *else*

Penerapan *if-else* pada bahasa pemrograman Go, sama seperti pada bahasa pemrograman lainnya, yang membedakan hanya pada tanda kurungnya (*parentheses*), di bahasa pemrograman Go tidak perlu ditulis tanda kurung.

Bentuk umum :

```
IF kondisi{
        statement
}Else If kondisi{
        statement
}Else If{
        statement
}Else{
        statement
}
```

Contoh Program :

```
var point = 8

if point == 10 {
    fmt.Println("lulus dengan nilai sempurna")
} else if point > 5 {
    fmt.Println("lulus")
} else if point == 4 {
    fmt.Println("hampir lulus")
} else {
    fmt.Printf("tidak lulus. nilai anda %d\n", point)
}
```

Pada program diatas, terdapat empat kondisi, yang terpenuhi adalah *if point > 5* , karena nilai variabel *point* memang lebih besar dari 5. Maka blok kode tepat dibawah kondisi tersebut akan dieksekusi (blok kode ditandai kurung kurawal buka dan tutup), hasilnya text *"lulus"* muncul sebagai output. Skema *if else* pada bahasa pemrograman Go sama seperti pada pemrograman umumnya, yaitu di awal seleksi kondisi menggunakan *if*, dan ketika kondisinya tidak terpenuhi maka akan menuju pada statement *else* (jika ada). Ketika terdapat banyak kondisi, maka gunakankanlah statement *else if* .

Catatan, pada bahasa pemrograman lain, ketika terdapat seleksi kondisi yang isi blok-nya hanya 1 baris, maka kurung kurawal boleh tidak dituliskan. Namun, berbeda dengan aturan di bahasa pemrograman Go, kurung kurawal harus tetap dituliskan meski isinya hanya 1 blok satement.

3.2.2 Variabel *Temporary* pada Statemen *if - else*

Pengertian dari variabel *temporary* adalah variabel yang hanya bisa digunakan pada blok seleksi kondisi dimana ia ditempatkan saja. Penggunaan variabel ini membawa beberapa manfaat, antara lain:

1. Scope atau cakupan variabel jelas, hanya bisa digunakan pada blok seleksi kondisi itu saja.
2. Kode menjadi lebih rapih.
3. Ketika nilai variabel tersebut didapat dari sebuah komputasi, perhitungan tidak perlu dilakukan di dalam blok masing-masing kondisi.

Bentuk umumnya sama seperti statement *if-else* biasa, yang berbeda adalah penggunaan dari nilai variabelnya saja.

Contoh Program :

```
var point = 8840.0

if percent := point / 100; percent >= 100 {
    fmt.Printf("%.1f%s perfect!\n", percent, "%")
} else if percent >= 70 {
    fmt.Printf("%.1f%s good\n", percent, "%")
} else {
    fmt.Printf("%.1f%s not bad\n", percent, "%")
}
```

Pada variabel *percent* nilainya didapat dari hasil perhitungan, dan hanya bisa digunakan pada deretan blok seleksi kondisi itu saja. Deklarasi variabel *temporary* hanya bisa dilakukan melalui metode *type inference*, yaitu menggunakan tanda :=. Penggunaan kata kunci var tidak diperbolehkan karena akan menyebabkan *error*.

3.2.3 Seleksi Kondisi Menggunakan Kata Kunci *switch - case*

Switch merupakan seleksi kondisi yang sifatnya fokus pada satu variabel, lalu kemudian dicek nilainya. Perlu diketahui, *switch* pada pemrograman Go memiliki perbedaan dibanding

bahasa lain. Pada pemrograman Go, ketika sebuah *case* terpenuhi, tidak akan dilanjutkan ke pengecekkan *case* selanjutnya, meskipun tidak ada kata kunci *break*. Konsep ini berkebalikan dengan *switch* pada umumnya, yaitu ketika sebuah *case* terpenuhi, maka akan tetap dilanjut untuk mengecek *case* selanjutnya kecuali jika ada kata kunci *break*.

Bentuk umum :

```
Switch var{
        case nilai1 :
                statement
        case nilai n :
                statement
        default :
                statement
}
```

Contoh program :

```
var point = 6

switch point {
case 8:
    fmt.Println("perfect")
case 7:
    fmt.Println("awesome")
default:
    fmt.Println("not bad")
}
```

Pada kode di atas, tidak terdapat kondisi atau *case* yang terpenuhi karena nilai variabel point tetap/konstanta sama dengan 6. Ketika hal seperti ini terjadi, blok kondisi *default* akan dipanggil. Bisa dikatakan bahwa *default* merupakan *else* dalam sebuah statement *switch*.

3.2.4 Pemanfaatan *case* untuk Banyak Kondisi

Sebuah *case* dapat menampung banyak kondisi. Cara penerapannya yaitu dengan menuliskan nilai pembanding-pembanding variabel yang dilakukan *switch* setelah kate kunci *case* dipisah oleh tanda koma (,).

Bentuk umum :

```
Switch var{
        case nilai1 :
                statement
        case nilai 2,3,4,5,dst :
                statement
        default :
                statement
}
```

Contoh program :

```
var point = 6

switch point {
case 8:
    fmt.Println("perfect")
case 7, 6, 5, 4:
    fmt.Println("awesome")
default:
    fmt.Println("not bad")
}
```

Pada program diatas, kondisi *case* 7, 6, 5, 4: akan terpenuhi ketika nilai variabel *point* adalah 7 atau 6 atau 5 atau 4, selain itu maka akan diteruskan pada statemen *default*.

3.2.5 Kurung Kurawal Pada Kata Kunci *case* dan *default*

Tanda kurung kurawal ({ }) dapat diterapkan pada kata kunci *case* dan *default* . Tanda kurung kurawal ini bersifat opsional, dapat digunakan juga dapat tidak digunakan. Baiknya jika dipakai pada blok kondisi yang didalamnya terdapat banyak statement, maka kode akan terlihat lebih rapih dan mudah dilakukan perawatan.

Bentuk umumnya sama seperti statement *switch case* biasa, yang berbeda adalah penggunaan dari *statement default* -nya saja.

Contoh program :

```
var point = 6

switch point {
case 8:
    fmt.Println("perfect")
case 7, 6, 5, 4:
    fmt.Println("awesome")
default:
    {
        fmt.Println("not bad")
        fmt.Println("you can be better!")
    }
}
```

Perhatikan kode program diatas, dapat dilihat pada kata kunci *default* terdapat kurung kurawal yang mengapit 2 statement didalamnya.

3.2.6 Switch Dengan Gaya if - else

Salah satu hal unik pada pemrograman Go yaitu, *statement switch* dapat digunakan dengan gaya seperti *statement if-else*. Nilai yang akan dibandingkan tidak dituliskan setelah kata kunci *switch*, melainkan akan ditulis langsung dalam bentuk perbandingan dalam kata kunci *case*.

Bentuk umum :

```
var nama_var
Switch{
        case nama_var:
                statement
        case (kondisi)
                statement
        default :
                statement
}
```

Contoh program :

```
var point = 6

switch {
case point == 8:
    fmt.Println("perfect")
case (point < 8) && (point > 3):
    fmt.Println("awesome")
default:
    {
        fmt.Println("not bad")
        fmt.Println("you need to learn more")
    }
}
```

Pada kode program di atas, kode program *switch* diubah ke dalam gaya *if-else.* Variabel *point* dihilangkan dari kata kunci *switch* , lalu kondisi-kondisinya dituliskan pata setiap *case*.

3.2.7 Penggunaan Kata Kunci *fallthrough* dalam *statement switch*

Seperti yang sudah dijelaskan sebelumnya, bahwa konsep *switch* pada pemrograman Go memiliki perbedaan dengan bahasa lain. Ketika sebuah *case* terpenuhi, pengecekkan kondisi tidak akan diteruskan pada *case-case* setelahnya. Untuk mengantisipasi agar pengecekan diteruskan pada *case* selanjutnya, maka dibutuhkan kata kunci *fallthrough* untuk memaksa proses pengecekkan diteruskan pada *case* selanjutnya.

Bentuk umum :

```
var nama_var
Switch{
        case nama_var:
                statement
        case (kondisi)
                statement
                fallthrough
        default :
                statement
}
```

Contoh program :

```
var point = 6

switch {
case point == 8:
    fmt.Println("perfect")
case (point < 8) && (point > 3):
    fmt.Println("awesome")
    fallthrough
case point < 5:
    fmt.Println("you need to learn more")
default:
    {
        fmt.Println("not bad")
        fmt.Println("you need to learn more")
    }
}
```

Pada program diatas, setelah melakukan pengecekkan *case (point < 8) && (point > 3)* selesai, maka akan dipaksa dilanjut ke pengecekkan *case point < 5* , karena terdapat *statement fallthrough*.

3.2.8 Seleksi Kondisi Bersarang

Seleksi kondisi bersarang merupakan seleksi kondisi yang berada didalam seleksi kondisi lain, yang mungkin juga berada dalam seleksi kondisi lain, dan seterusnya. Seleksi kondisi bersarang bisa dilakukan pada *statement if - else, switch,* ataupun kombinasi keduanya.

Bentuk umum :

```
var nama_var

If kondisi{
        Switch nama_var{
                case nama_var:
                        statement
                default :
                        statement
                }

} Else {

        If kondisi{
                statement
        } Else If  kondisi{
                statement
        } Else {
                statement
                If kondisi{
                        statement
                }
        }
}
```

Contoh program :

```
var point = 10

if point > 7 {
    switch point {
    case 10:
        fmt.Println("perfect!")
    default:
        fmt.Println("nice!")
    }
} else {
    if point == 5 {
        fmt.Println("not bad")
    } else if point == 3 {
        fmt.Println("keep trying")
    } else {
        fmt.Println("you can do it")
        if point == 0 {
            fmt.Println("try harder!")
        }
    }
}
```

Pada program diatas, terdapat dua statement kondisi yang dikombinasikan untuk menyeleksi nilai dari variabel *point*.