MODUL II
TIPE DATA & VARIABEL

Pertemuan : 1
Waktu : 8 x 60 menit (Online)

1.1 Tujuan Modul II

Setelah mahasiswa mempelajari materi ini, diharapkan dapat :

1. Memahami tipe data dasar di golang.
2. Memahami penggunaan variable di golang.
3. Memahami input output data dasar di golang.

1.2 Landasan Teori

1.2.1 Variable

Go mengadopsi dua jenis penulisan variabel, yaitu yang dituliskan tipe data-nya, dan juga yang tidak. Kedua cara tersebut valid dan tujuannya sama, pembedanya hanya cara penulisannya saja.

1.2.1.1 Deklarasi Variabel Beserta Tipe Data

Go memiliki aturan cukup ketat dalam hal penulisan variabel. Ketika deklarasi, tipe data yg digunakan harus dituliskan juga. Istilah lain dari konsep ini adalah **manifest typing**.Berikut adalah contoh cara pembuatan variabel yang tipe datanya harus ditulis. Silahkan tulis pada projek baru atau pada proyek yang sudah ada, bebas.

```
package main

import "fmt"

func main() {

    var firstName string = "john"

    var lastName string

    lastName = "wick"

    fmt.Printf("halo %s %s!\n", firstName, lastName)

}
```

Keyword `var` di atas digunakan untuk deklarasi variabel, contohnya bisa dilihat pada `firstName` dan `lastName`.Nilai variabel `firstName` diisi langsung ketika deklarasi, berbeda dibanding `lastName` yang nilainya diisi setelah baris kode deklarasi, hal seperti ini diperbolehkan di Go.

1.2.1.2 Deklarasi Variabel Menggunakan Keyword var

Pada kode di atas bisa dilihat bagaimana sebuah variabel dideklarasikan dan diisi nilainya. Keyword `var` digunakan untuk membuat variabel baru.Skema penggunaan keyword var:

```
var <nama-variabel> <tipe-data>
var <nama-variabel> <tipe-data> = <nilai>
```

Contoh:

```
var lastName string
var firstName string = "john”
```

Nilai variabel bisa di-isi langsung pada saat deklarasi variabel.

1.2.1.3 Deklarasi Variabel Tanpa Tipe Data

Selain *manifest typing*, Go juga mengadopsi konsep **type inference**, yaitu metode deklarasi variabel yang tipe data-nya ditentukan oleh tipe data nilainya, cara kontradiktif jika dibandingkan dengan cara pertama. Dengan metode jenis ini, keyword `var` dan tipe data tidak perlu ditulis.

```
var firstName string = "john"
lastName := "wick"

fmt.Printf("halo %s %s!\n", firstName, lastName)
```

Variabel `lastName` dideklarasikan dengan menggunakan metode type inference. Penandanya tipe data tidak dituliskan pada saat deklarasi. Pada penggunaan metode ini, operand `=` harus diganti dengan `:=` dan keyword `var` dihilangkan.

Tipe data `lastName` secara otomatis akan ditentukan menyesuaikan value atau nilai-nya. Jika nilainya adalah berupa `string` maka tipe data variabel adalah `string`. Pada contoh diatas, nilainya adalah string `"wick"`.

Diperbolehkan untuk tetap menggunakan keyword `var` pada saat deklarasi meskipun tanpa menuliskan tipe data, dengan ketentuan tidak menggunakan tanda `:=`, melainkan tetap menggunakan `=`.

```
// menggunakan var, tanpa tipe data, menggunakan perantara
"="
var firstName = "john"

// tanpa var, tanpa tipe data, menggunakan perantara ":="
lastName := "wick”
```

Kedua deklarasi di atas maksudnya sama. Silakan pilih yang nyaman di hati.Tanda `:=` hanya digunakan sekali di awal pada saat deklarasi. Untuk assignment nilai selanjutnya harus menggunakan tanda `=`, contoh:

```
lastName := "wick"

lastName = "ethan"

lastName = "bourne"
```

1.2.1.4 Deklarasi Multi Variabel

Go mendukung metode deklarasi banyak variabel secara bersamaan, caranya dengan menuliskan variabel-variabel-nya dengan pembatas tanda koma (`,`). Untuk pengisian nilainya-pun diperbolehkan secara bersamaan.

```
var first, second, third string
first, second, third = "satu", "dua", "tiga"
```

Pengisian nilai juga bisa dilakukan bersamaan pada saat deklarasi. Caranya dengan menuliskan nilai masing-masing variabel berurutan sesuai variabelnya dengan pembatas koma (`,`).

```
var fourth, fifth, sixth string = "empat", "lima", "enam"
```

Kalau ingin lebih ringkas:

```
seventh, eighth, ninth := "tujuh", "delapan", "sembilan"
```

1.2.1.5 Variabel Underscore

Go memiliki aturan unik yang jarang dimiliki bahasa lain, yaitu tidak boleh ada satupun variabel yang menganggur. Artinya, semua variabel yang dideklarasikan harus digunakan. Jika ada variabel yang tidak digunakan tapi dideklarasikan, error akan muncul pada saat kompilasi dan program tidak akan bisa di-run.

Underscore (`_`) adalah *reserved variable* yang bisa dimanfaatkan untuk menampung nilai yang tidak dipakai. Bisa dibilang variabel ini merupakan keranjang sampah.

```
_ = "belajar Golang"

_ = "Golang itu mudah"

name, _ := "john", "wick"
```

Pada contoh diatas, variabel `name` akan berisikan text `john`, sedang nilai `wick` ditampung oleh variabel underscore, menandakan bahwa nilai tersebut tidak akan digunakan.

Variabel underscore adalah *predefined*, jadi tidak perlu menggunakan `:=` untuk pengisian nilai, cukup dengan `=` saja. Namun khusus untuk pengisian nilai multivariabel yang dilakukan dengan metode type inference, boleh didalamnya terdapat variabel underscore.

Biasanya variabel underscore sering dimanfaatkan untuk menampung nilai balik fungsi yang tidak digunakan.Perlu diketahui, bahwa isi variabel underscore tidak dapat ditampilkan. Data yang sudah masuk variabel tersebut akan hilang.

1.2.1.6 Deklarasi Variabel Menggunakan Keyword new

Keyword `new` digunakan untuk membuat variabel **pointer** dengan tipe data tertentu. Nilai data default-nya akan menyesuaikan tipe datanya.

```
name := new(string)

fmt.Println(name)   // 0x20818a220
fmt.Println(*name)  // ""
```

Variabel `name` menampung data bertipe **pointer string**. Jika ditampilkan yang muncul bukanlah nilainya melainkan alamat memori nilai tersebut (dalam bentuk notasi heksadesimal). Untuk menampilkan nilai aslinya, variabel tersebut perlu di-**dereference** terlebih dahulu, menggunakan tanda asterisk (`*`).Mungkin untuk sekarang banyak yang akan bingung tentang apa itu pointer, namun tak apa, karena nantinya kita bahas topik tersebut.

1.2.2 Tipe Data

Go mengenal beberapa jenis tipe data, diantaranya adalah tipe data numerik (desimal & non-desimal), string, dan boolean.

1.2.2.1 Tipe Data Numerik Non-Desimal

Tipe data numerik non-desimal atau **non floating point** di Go ada beberapa jenis. Secara umum ada 2 tipe data kategori ini yang perlu diketahui.

- **uint**, tipe data untuk bilangan cacah (bilangan positif).
- **int**, tipe data untuk bilangan bulat (bilangan negatif dan positif).

Kedua tipe data di atas kemudian dibagi lagi menjadi beberapa jenis, dengan pembagian berdasarkan lebar cakupan nilainya, detailnya bisa dilihat di tabel berikut.

Tipe Data	Range Bilangan
uint8	0 ↔ 255
uint16	0 ↔ 65535
uint32	0 ↔ 4294967295
uint64	0 ↔ 18446744073709551615
uint	sama dengan uint32 atau uint64 (tergantung nilai)
byte	sama dengan uint8
int8	-128 ↔ 127
int16	-32768 ↔ 32767

int32	-2147483648 ↔ 2147483647
int64	-9223372036854775808 ↔ 9223372036854775807
int	sama dengan int32 atau int64 (tergantung nilai)
rune	sama dengan int32

Dianjurkan untuk tidak sembarangan dalam menentukan tipe data variabel, sebisa mungkin tipe yang dipilih harus disesuaikan dengan nilainya, karena efeknya adalah ke alokasi memori variabel. Pemilihan tipe data yang tepat akan membuat pemakaian memori lebih optimal, tidak berlebihan.

```
var positiveNumber uint8 = 89
var negativeNumber = -1243423644

fmt.Printf("bilangan positif: %d\n", positiveNumber)
fmt.Printf("bilangan negatif: %d\n", negativeNumber)
```

Variabel `positiveNumber` bertipe `uint8` dengan nilai awal `89`. Sedangkan variabel `negativeNumber` dideklarasikan dengan nilai awal `-1243423644`. Compiler secara cerdas akan menentukan tipe data variabel tersebut sebagai `int32` (karena angka tersebut masuk ke cakupan tipe data `int32`).Template `%d` pada `fmt.Printf()` digunakan untuk memformat data numerik non-desimal.

1.2.2.2 Tipe Data Numerik Desimal

Tipe data numerik desimal yang perlu diketahui ada 2, float32 dan float64. Perbedaan kedua tipe data tersebut berada di lebar cakupan nilai desimal yang bisa ditampung. Untuk lebih jelasnya bisa merujuk ke spesifikasi IEEE-754 32-bit floating-point numbers.Untuk kodenya seperti dibawah ini.

```
var decimalNumber = 2.62

fmt.Printf("bilangan desimal: %f\n", decimalNumber)
fmt.Printf("bilangan desimal: %.3f\n", decimalNumber)
```

Template `%f` digunakan untuk memformat data numerik desimal menjadi string. Digit desimal yang akan dihasilkan adalah **6 digit**. Pada contoh diatas, hasil format variable `decimalNumber` adalah `2.620000`. Jumlah digit yang muncul bisa dikontrol menggunakan `%.nf`, tinggal ganti `n` dengan angka yang diinginkan. Contoh: `%.3f` maka akan menghasilkan 3 digit desimal, `%.10f` maka akan menghasilkan 10 digit desimal.

1.2.2.3 Tipe Data Bool (Boolean)

Tipe data `bool` berisikan hanya 2 variansi nilai, `true` dan `false`. Tipe data ini biasa dimanfaatkan dalam seleksi kondisi dan perulangan.untuk contoh kodenya seperti dibawah ini.

```
var exist bool = true
fmt.Printf("exist? %t \n", exist)
```

Gunakan `%t` untuk memformat data `bool` menggunakan fungsi `fmt.Printf()`.

1.2.2.4 Tipe Data String

Ciri khas dari tipe data string adalah nilainya diapit oleh tanda *quote* atau petik dua (`"`). Contoh penerapannya:

```
var message string = "Halo"
fmt.Printf("message: %s \n", message)
```

Selain menggunakan tanda quote, deklarasi string juga bisa dengan tanda *grave accent/backticks* (`` ` ``), tanda ini terletak di sebelah kiri tombol 1. Keistimewaan string yang dideklarasikan menggunakan backticks adalah membuat semua karakter di dalamnya **tidak di escape**, termasuk `\n`, tanda petik dua dan tanda petik satu, baris baru, dan lainnya. Semua akan terdeteksi sebagai string.Contoh penerapannya:

```
var message = `Nama saya "John Wick".
Salam kenal.
Mari belajar "Golang".`
```

```
fmt.Println(message)
```

1.2.2.5 Nilai Nil & Zero Value

`nil` bukan merupakan tipe data, melainkan sebuah nilai. Variabel yang isi nilainya `nil` berarti memiliki nilai kosong.Semua tipe data yang sudah dibahas di atas memiliki zero value (nilai default tipe data). Artinya meskipun variabel dideklarasikan dengan tanpa nilai awal, tetap akan ada nilai default-nya.

- Zero value dari `string` adalah `""` (string kosong).
- Zero value dari `bool` adalah `false`.
- Zero value dari tipe numerik non-desimal adalah `0`.
- Zero value dari tipe numerik desimal adalah `0.0`.

Zero value berbeda dengan `nil`. `Nil` adalah nilai kosong, benar-benar kosong. `nil` tidak bisa digunakan pada tipe data yang sudah dibahas di atas. Ada beberapa tipe data yang bisa di-set nilainya dengan `nil`, diantaranya:

- pointer
- tipe data fungsi
- slice
- map
- channel
- interface kosong atau `interface{}`

1.3 Praktikum

1.3.1 Latihan Praktikum

```
1 package main
2
3 import (
4         "fmt"
5 )
6
7 func main() {
8         var nama, mataKuliah string
9         var nilai, nim int
10        fmt.Println("Masukkan NIM Anda : ")
11        fmt.Scan(&nim)
12        fmt.Println("Masukkan Nama Anda: ")
13        fmt.Scanf("%s", &nama)
14        fmt.Println("Masukkan Mata Kuliah Anda: ")
15        fmt.Scanf("%s", &mataKuliah)
16        fmt.Println("Masukkan Nilai Anda: ")
17        fmt.Scanln(&nilai)
18
19        fmt.Printf("Nama(NIM) => %d %d \n", nama, nim)
20        fmt.Printf("Matkul(Nilai) => %s %s \n", mataKuliah, nilai)
21
22 }
```

Untuk Latihan Praktikum Hari ini coba pahami source code yang di atas,jika ada kesalahan dalam penulisan kode perbaiki agar program bisa berjalan lancar

1.3.2 Tugas Praktikum

Coming Soon