MODUL XII
POINTER

Pertemuan : 12
Waktu : 8 x 60 menit (Online)

1.1 Tujuan Modul III

Setelah mahasiswa mempelajari materi ini, diharapkan dapat :

1. Memahami penggunaan pointer di golang baik parameter maupun method.

1.2 Landasan Teori

Pointer adalah *reference* atau alamat memori. Variabel pointer berarti variabel yang berisi alamat memori suatu nilai. Sebagai contoh sebuah variabel bertipe integer memiliki nilai **4**, maka yang dimaksud pointer adalah **alamat memori dimana nilai 4 disimpan**, bukan nilai 4 itu sendiri.

Variabel-variabel yang memiliki *reference* atau alamat pointer yang sama, saling berhubungan satu sama lain dan nilainya pasti sama. Ketika ada perubahan nilai, maka akan memberikan efek kepada variabel lain (yang referensinya sama) yaitu nilainya ikut berubah.

1.2.1 Penerapan Pointer

Variabel bertipe pointer ditandai dengan adanya tanda **asterisk** (*) tepat sebelum penulisan tipe data ketika deklarasi.

```
var number *int
var name *string
```

Nilai default variabel pointer adalah `nil` (kosong). Variabel pointer tidak bisa menampung nilai yang bukan pointer, dan sebaliknya variabel biasa tidak bisa menampung nilai pointer.

Ada dua hal penting yang perlu diketahui mengenai pointer:

- Variabel biasa bisa diambil nilai pointernya, caranya dengan menambahkan tanda **ampersand** (&) tepat sebelum nama variabel. Metode ini disebut dengan **referencing**.
- Dan sebaliknya, nilai asli variabel pointer juga bisa diambil, dengan cara menambahkan tanda **asterisk** (*) tepat sebelum nama variabel. Metode ini disebut dengan **dereferencing**.

OK, langsung saja kita praktekan.

```
var numberA int = 4
var numberB *int = &numberA

fmt.Println("numberA (value)   :", numberA)  // 4
fmt.Println("numberA (address) :", &numberA) //
0xc20800a220

fmt.Println("numberB (value)   :", *numberB) // 4
fmt.Println("numberB (address) :", numberB)  //
0xc20800a220
```

Variabel `numberB` dideklarasikan bertipe pointer `int` dengan nilai awal adalah referensi variabel `numberA` (bisa dilihat pada kode `&numberA`). Dengan ini, variabel `numberA` dan `numberB` menampung data dengan referensi alamat memori yang sama.

```
[novalagung:belajar-golang $ go run bab22.go
numberA (value)   : 4
numberA (address) : 0xc20800a220
numberB (value)   : 4
numberB (address) : 0xc20800a220
novalagung:belajar-golang $
```

Variabel pointer jika di-print akan menghasilkan string alamat memori (dalam notasi heksadesimal), contohnya seperti `numberB` yang diprint menghasilkan `0xc20800a220`.

Nilai asli sebuah variabel pointer bisa didapatkan dengan cara di-dereference terlebih dahulu (bisa dilihat pada kode `*numberB`).

1.2.2 Efek Perubahan Nilai Pointer

Ketika salah satu variabel pointer di ubah nilainya, sedang ada variabel lain yang memiliki referensi memori yang sama, maka nilai variabel lain tersebut juga akan berubah.

```
var numberA int = 4
var numberB *int = &numberA

fmt.Println("numberA (value)   :", numberA)
fmt.Println("numberA (address) :", &numberA)
fmt.Println("numberB (value)   :", *numberB)
fmt.Println("numberB (address) :", numberB)

fmt.Println("")

numberA = 5

fmt.Println("numberA (value)   :", numberA)
fmt.Println("numberA (address) :", &numberA)
fmt.Println("numberB (value)   :", *numberB)
fmt.Println("numberB (address) :", numberB)
```

Variabel `numberA` dan `numberB` memiliki referensi memori yang sama. Perubahan pada salah satu nilai variabel tersebut akan memberikan efek pada variabel lainnya. Pada contoh di atas, `numberA` nilainya diubah menjadi `5`. membuat nilai asli variabel `numberB` ikut berubah menjadi `5`.

1.2.3 Parameter Pointer

Parameter bisa juga didesain sebagai pointer. Cara penerapannya kurang lebih sama, dengan cara mendeklarasikan parameter sebagai pointer.

```
package main

import "fmt"

func main() {
    var number = 4
    fmt.Println("before :", number) // 4
```

```
    change(&number, 10)
    fmt.Println("after  :", number) // 10
}

func change(original *int, value int) {
    *original = value
}
```

Fungsi `change()` memiliki 2 parameter, yaitu `original` yang tipenya adalah pointer `int`, dan `value` yang bertipe `int`. Di dalam fungsi tersebut nilai asli parameter pointer `original` diubah.

Fungsi `change()` kemudian diimplementasikan di `main`. Variabel `number` yang nilai awalnya adalah `4` diambil referensi-nya lalu digunakan sebagai parameter pada pemanggilan fungsi `change()`.

Nilai variabel `number` berubah menjadi `10` karena perubahan yang terjadi di dalam fungsi `change` adalah pada variabel pointer.

1.2.4 Method Pointer

Method pointer adalah method yang variabel objek pemilik method tersebut berupa pointer.

Kelebihan method jenis ini adalah, ketika kita melakukan manipulasi nilai pada property lain yang masih satu struct, nilai pada property tersebut akan diubah pada reference nya. Lebih jelasnya perhatikan kode berikut.

```
package main

import "fmt"

type student struct {
    name  string
    grade int
}

func (s student) changeName1(name string) {
    fmt.Println("---> on changeName1, name changed to",
name)
    s.name = name
}
```

```
func (s *student) changeName2(name string) {
    fmt.Println("---> on changeName2, name changed to",
name)
    s.name = name
}

func main() {
    var s1 = student{"john wick", 21}
    fmt.Println("s1 before", s1.name)
    // john wick

    s1.changeName1("jason bourne")
    fmt.Println("s1 after changeName1", s1.name)
    // john wick

    s1.changeName2("ethan hunt")
    fmt.Println("s1 after changeName2", s1.name)
    // ethan hunt
}
```

Output:

```
[novalagung:chapter-24 $ go run 2-method-pointer.go
s1 before john wick
---> on changeName1, name changed to jason bourne
s1 after changeName1 john wick
---> on changeName2, name changed to ethan hunt
s1 after changeName2 ethan hunt
```

Setelah eksekusi statement `s1.changeName1("jason bourne")`, nilai `s1.name` tidak berubah. Sebenarnya nilainya berubah tapi hanya dalam method `changeName1()` saja, nilai pada reference di objek-nya tidak berubah. Karena itulah ketika objek di print value dari `s1.name` tidak berubah.

Keistimewaan lain method pointer adalah, method itu sendiri bisa dipanggil dari objek pointer maupun objek biasa.

```
// pengaksesan method dari variabel objek biasa
var s1 = student{"john wick", 21}
s1.sayHello()

// pengaksesan method dari variabel objek pointer
var s2 = &student{"ethan hunt", 22}
```

```
s2.sayHello()
```

1.3 Praktikum

1.3.1 Latihan Praktikum

Coming Soon