Struct

Struct

Bahasa Go tidak menyediakan tipe data class untuk membuat sebuah struktur data seperti layaknya bahasa pemrograman yang menganut paradigma OOP. Tetapi Go menyediakan sebuah tipe data yang bernama **Struct** untuk membuat sebuah struktur.

Struct adalah sebuah tipe data berupa kumpulan/koleksi dari berbagai macam property/field dan juga method. Cara membuat *struct* pada Go cukup mudah. Contohnya seperti pada gambar di sebelah kanan.

```
package main
...
type Employee struct {
  name     string
  age      int
  division string
}

func main() {

}
```

Untuk membuat sebuah tipe data *struct* , kita perlu membuat terlebih dahulu strukturnya dengan urutan format penulisan:

- Penulisan keyword *type*
- Nama dari *struct*
- Penulisan keyword *struct*
- Kemudian diikuti dengan tanda kurung kurawal {}
- Mendefinisikan *field* yang diinginkan

Struct (Giving value to struct)

Agar kita dapat memberikan nilai kepada field yang terdapat pada sebuah *struct*, kita perlu terlebih dahulu menyimpan tipe data dari *struct* kepada sebuah variable. Contohnya seperti pada gambar pertama di sebelah kanan.

Kita membuat sebuah struct dengan nama *Employee* yang memiliki 3 field. Lalu kita membuat sebuah variable bernama *employee* yang memiliki tipe data berupa *struct Employee*. Kemudian kita memberikan nilai- nilai kepada field-field pada struct *Employee* dengan cara mengakses field-field melewati variable yang menyimpan tipe data *struct Employee*.

Jika kita jalankan pada terminal kita, maka hasilnya akan seperti pada gambar kedua.

```
7  type Employee struct {
8      name      string
9      age       int
10     division string
11 }
12
13 func main() {
14     var employee Employee
15
16     employee.name = "Airell"
17
18     employee.age = 23
19
20     employee.division = "Curriculum Developer"
21
22     fmt.Println(employee.name)
23     fmt.Println(employee.age)
24     fmt.Println(employee.division)
25 }
```

```
Airell
23
Curriculum Developer
```

Struct (Initializing struct)

Kita juga dapat menginisialisasi sebuah *struct* sekaligus memberikan nilai-nilai nya. Contohnya seperti pada gambar pertama di sebelah kanan.

Pada line 14 kita membuat sebuah variable bernama *employee1* yang dimana variable tersebut telah diinisialisasi namun tidak langsung diberikan nilai, melainkan kita memberikan nilainya dengan cara mengakses satu per satu fieldnya seperti pada halaman sebelumnya.

Pada line 19 kita membuat variable *employee2* yang dimana variable ini telah diinisialisasi sekaligus diberikan nilainya. Perhatikan cara penulisan cara memberikan nilainya secara langsung.

Kemudian kita menggunakan verb %+v untuk memformat sebuah struct menjadi *string*. Jika kita jalankan pada terminal maka hasilnya akan seperti pada gambar kedua.

```
7  type Employee struct {
8    name     string
9    age      int
10   division string
11 }
12
13 func main() {
14   var employee1 = Employee{}
15   employee1.name = "Airell"
16   employee1.age = 23
17   employee1.division = "Curriculum Developer"
18
19   var employee2 = Employee{name: "Ananda", age: 23, division: "Finance"}
20
21   fmt.Printf("Employee1: %+v\n", employee1)
22   fmt.Printf("Employee2: %+v\n", employee2)
23 }
```

```
Employee1: {name:Airell age:23 division:Curriculum Developer}
Employee2: {name:Ananda age:23 division:Finance}
```

HACKTIV8

Struct (Pointer to a struct)

Kita juga dapat menggunakan *pointer* pada sebuah *struct*. Contoh nya seperti pada gambar pertama di sebelah kanan.

Pada line 43 kita membuat sebuah variable bernama *employee2* dengan tipe data *pointer to a struct* yang menyimpan alamat memori dari variable *employee1*. Kemudian pada line 50, *employee2* merubah nilai dari field *name* nya menjadi "Ananda" yang semula nilai nya adalah "Airell".

Ketika variable *employee2* merubah nilai dari fieldnya, maka variable employee1 juga akan ikut terubah nilainya. Hal tersebut dapat dilihat dari hasil eksekusi syntax gambar pertama pada gambar kedua.

```
34  type Employee struct {
35    name      string
36    age       int
37    division string
38  }
39
40  func main() {
41    var employee1 = Employee{name: "Airell", age: 23, division: "Curriculum Developer"}
42
43    var employee2 *Employee = &employee1
44
45    fmt.Println("Employee1 name:", employee1.name)
46    fmt.Println("Employee2 name", employee2.name)
47
48    fmt.Println(strings.Repeat("#", 20))
49
50    employee2.name = "Ananda"
51
52    fmt.Println("Employee1 name:", employee1.name)
53    fmt.Println("Employee2 name", employee2.name)
54
55  }
```

```
Employee1 name: Airell
Employee2 name Airell
####################
Employee1 name: Ananda
Employee2 name Ananda
```

Struct (Embedded struct)

Struct juga dapat mengandung tipe data *struct* lainnya dengan menjadikannya sebuah field. Contohnya seperti pada gambar pertama di sebelah kanan.

Terdapat 2 tipe data struct dengan nama *Person* dan *Employee*. Lalu tipe data employee mengandung field *person* yang memiliki tipe data struct *Person*.

Kemudian perhatikan pada line 45 dan 46 yang merupakan cara variable *employee1* memberikan nilai kepada field *name* dan *age* yang berasal dari field *person*. Kita perlu mengakses field *person* terlebih dahulu kemudian mengakses field yang ingin diberikan nilai.

Jika kita jalankan pada terminal kita, maka hasilnya akan seperti pada gambar kedua dibawah.

```
27  package main
28
29  import "fmt"
30
31  type Person struct {
32    name string
33    age  int
34  }
35
36  type Employee struct {
37    division string
38    person   Person
39  }
40
41  func main() {
42    var employee1 = Employee{}
43
44    employee1.person.name = "Airell"
45    employee1.person.age = 23
46    employee1.division = "Curriculum Developer"
47
48    fmt.Printf("%+v", employee1)
49  }
```

```
{division:Curriculum Developer person:{name:Airell age:23}}
```

Struct (Anonymous struct)

Anonymous struct adalah sebuah struct yang tidak dideklerasikan di awal sebagai sebuah tipe data *struct* baru, melainkan langsung dideklerasikan bersamaan dengan pembuatan variable. Contohnya seperti pada gambar pertama di sebelah kanan.

Terdapat sebuah *struct* bernama *Person* yang mempunyai 2 field. Kemudian pada line 64, terdapat sebuah variable bernama *employee1* yang mengandung tipe data struct yang merupakan sebuah *Anonymous struct* tanpa pengisian nilai field. Salah satu aturan yang perlu diingat dalam pembuatan anonymous struct adalah, deklarasi harus diikuti dengan inisialisasi. Bisa dilihat pada variable *employee1* setelah deklarasi struktur struct, terdapat kurung kurawal untuk inisialisasi. Meskipun nilai tidak diisikan di awal, kurung kurawal tetap harus ditulis.

Kemudian pada line 72 terdapat sebuah variable bernama *employee2* yang juga mengandung tipe data struct berupa *Anonymous struct* dengan pengisian nilai field.

Ketika dijalankan pada terminal kita, maka hasilnya akan seperti pada gambar kedua.

```
57  type Person struct {
58    name string
59    age  int
60  }
61
62  func main() {
63    //Anonymous struct tanpa pengisian field
64    var employee1 = struct {
65      person   Person
66      division string
67    }{}
68    employee1.person = Person{name: "Airell", age: 23}
69    employee1.division = "Curriculum developer"
70
71    //Anonymous struct dengan pengisian field
72    var employee2 = struct {
73      person   Person
74      division string
75    }{
76      person:   Person{name: "Ananda", age: 23},
77      division: "Finance",
78    }
79
80    fmt.Printf("Employee1: %+v\n", employee1)
81    fmt.Printf("Employee1: %+v\n", employee2)
82
83  }
```

```
Employee1: {person:{name:Airell age:23} division:Curriculum developer}
Employee1: {person:{name:Ananda age:23} division:Finance}
```

Struct (Slice of struct)

Slice juga dapat dikombinasikan dengan tipe data struct, cara penulisannya mirip seperti *slice of map* yang telah kita bahas pada materi sebelumnya. Contohnya seperti gambar pertama di sebelah kanan.

Terdapat sebuah variable bernama *people* yang memiliki tipe data *slice of struct* dan tipe data *struct* yang dimasukkan kedalam *slice* nya adalah *struct* bernama *Person*.

Kemudian kita juga dapat me-looping *slice of struct* tersebut dengan *range loop*.

Jika kita jalankan pada terminal, maka hasilnya akan seperti pada gambar kedua.

```go
package main

import "fmt"

type Person struct {
  name string
  age  int
}

func main() {

  var people = []Person{
    {name: "Airell", age: 23},
    {name: "Ananda", age: 23},
    {name: "Mailo", age: 23},
  }

  for _, v := range people {
    fmt.Printf("%+v\n", v)
  }

}
```

```
{name:Airell age:23}
{name:Ananda age:23}
{name:Mailo age:23}
```

Struct (Slice of anonymous struct)

Anonymous struct juga dapat digabungkan dengan tipe data *slice* dan pengisian nilainya pun dapat dilakukan secara langsung. Contohnya seperti pada gambar pertama di sebelah kanan.

Terdapat sebuah variable bernama *employee* dengan tipe data *slice* dan isi dari *slice* nya adalah sebuah *anonymous struct* yang langsung diberikan nilai untuk setiap fieldnya.

Kita juga dapat me-loopingnya menggunakan *range loop*.

Jika kita jalankan pada terminal, maka hasilnya akan seperti pada gambar kedua.

```go
package main

import "fmt"

type Person struct {
  name string
  age  int
}

func main() {

  var employee = []struct {
    person   Person
    division string
  }{
    {person: Person{name: "Airell", age: 23}, division: "Curriculum Developer"},
    {person: Person{name: "Ananda", age: 23}, division: "Finance"},
    {person: Person{name: "Mailo", age: 21}, division: "Marketing"},
  }

  for _, v := range employee {
    fmt.Printf("%+v\n", v)
  }

}
```

```
{person:{name:Airell age:23} division:Curriculum Developer}
{person:{name:Ananda age:23} division:Finance}
{person:{name:Mailo age:21} division:Marketing}
```