Structs

Struct pada go adalah type yang terdiri dari sekumpulan field. Sangat berguna untuk mengelompokkan data bersama untuk menjadi records. Atau bisa dikatakan ini adalah metode OOP dalam Go

Struct type orang terdiri dari field nama dan umur

Syntax ini untuk membuat struct baru.

Anda dapat memberi nama struct ketika proses inisialisasi struct.

Field yang tidak terdefenisi akan dianggap bernilai nol.

Tanda & adalah pointer ke struct

Akses field pada struct dengan menggunakan titik (.)

Anda juga dapat menggunakan titik (.) dengan pointer.

Structs memungkinkan untuk berubah.

```
package main

import "fmt"

type orang struct {
    nama string
    umur  int
}
func main() {

    fmt.Println(orang{"cinta", 20})

    fmt.Println(orang{nama: "anti", umur:
30})


    fmt.Println(orang{nama: "rangga"})

    fmt.Println(&orang{nama: "beti", umur:
40})
    s := person{nama: "budi", umur: 50}
    fmt.Println(s.nama)

    sp := &s
    fmt.Println(sp.umur)

    sp.umur = 51
    fmt.Println(sp.umur)
}
```

Contoh lain :

```
package main
import "fmt"

type Passport struct {
	Photo       []byte
	Name        string
	Surname     string
	DateOfBirth string
}

func main() {
	var p1 Passport
	p2 := Passport{}
	p3 := Passport{
		Photo:       make([]byte, 0, 0),
		Name:        "Nugroho",
		Surname:     "Agung",
		DateOfBirth: "UdahLAma",
	}

	fmt.Println(p1, p2, p3)

	p3.DateOfBirth = "Lama Sekali"
	fmt.Println(p3.DateOfBirth)

	pp3 := &p3
	fmt.Println(pp3)

	pp4 := new(Passport)
	fmt.Println(pp4)
}
```

Methods

Go mendukung methods yang dapat didefenisikan pada type struct.

Area method memiliki penerima (receiver) type *rect.

Method dapat didefinisi untuk pointer atau nilai type receiver. Dalam baris ini adalah contoh nilai dari receiver.

Disini kita memanggil 2 method yg sebelumnya terdefenisi pada struct.

Go secara otomatis dapat menghandel konversi antara nilai dan pointer untuk method call. Anda juga dapat menggunakan type pointer receiver untuk menghindari duplikasi pada method call.

```
package main

import "fmt"

type rect struct {
    width, height int
}
func (r *rect) area() int {
    return r.width * r.height
}

func (r rect) perim() int {
    return 2*r.width + 2*r.height
}

func main() {
    r := rect{width: 10, height: 5}

    fmt.Println("area: ", r.area())
    fmt.Println("perim:", r.perim())

    rp := &r
    fmt.Println("area: ", rp.area())
    fmt.Println("perim:", rp.perim())
}
```

```
package main
import "fmt"

type Passport struct {
	Photo		[]byte
	Name		string
	Surname		string
	DateOfBirth	string
}

func (p Passport) NamaLengkap() string {
	return fmt.Sprintf("%s %s", p.Name, p.Surname)
}

func main() {
	p1 := new(Passport)
	p1.Name = "nugroho"
	p1.Surname = "agung"
	fmt.Println(p1.NamaLengkap())
}
```

Jika anda eksekusi, akan menghasilkan output “nugroho agung”. Pada contoh dibawah ini, kita membuat 1 fungsi lagi untuk GantiNama.

```
package main
import "fmt"

type Passport struct {
	Photo []byte
	Name string
	Surname string
	DateOfBirth string
}

func (p Passport) NamaLengkap() string {
```

```
	return fmt.Sprintf("%s %s", p.Name, p.Surname)
}

func (p *Passport) GantiNamaLengkap(name string, surname string) {
	p.Name = name
	p.Surname = surname
}

func main() {
	p1 := new(Passport)
	p1.GantiNamaLengkap("Tonny", "Oscar")
	fmt.Println(p1.NamaLengkap())
}
```

Interfaces

Interfaces adalah nama koleksi dari method.

Misalkan dalam contoh ini membuat interface yang bernama geometry.

Dalam contoh ini interface akan diimplementasikan dalam type rect dan circle.

Untuk implementasi interface pada go, kita harus implementasikan semua method kedalam interface. Dibaris ini, kita akan implementasi interface geometry pada rect.

Implementasi interface geometry pada circle

Jika variable memiliki type interface, maka anda dapat memanggil method yang telah terdefenisi didalam interface. Fungsi measure memanfaatkan interface geometry

Type struct Circle dan rect, dimana keduanya implementasi geometry interface, sehingga anda dapat menggunakan struct sebagai argumen untuk measure.

```
package main
import "fmt"
import "math"
type geometry interface {
    area() float64
    perim() float64
}
type rect struct {
    width, height float64
}
type circle struct {
    radius float64
}
func (r rect) area() float64 {
    return r.width * r.height
}
func (r rect) perim() float64 {
    return 2*r.width + 2*r.height
}
func (c circle) area() float64 {
    return math.Pi * c.radius *
c.radius
}
func (c circle) perim() float64 {
    return 2 * math.Pi * c.radius
}
func measure(g geometry) {
    fmt.Println(g)
    fmt.Println(g.area())
    fmt.Println(g.perim())
}
func main() {
    r := rect{width: 3, height: 4}
    c := circle{radius: 5}

    measure(r)
    measure(c)
}
```

Question

Apa perbedaan method dan fungsi ?
Berikan kesimpulan anda tentang struct, method dan interfaces ?