

목차

• 포인터

• 구조체

포인터

- 다른 변수, 혹은 그 변수의 메모리 주소공간을 가리키는 변수
- 보통 c, c++등 하위레벨 언어에서 많이 사용됨
- 직접 메모리 주소를 조작할 수 있음 (golang은 안됨)
- * 키워드를 사용함

포인터

```
package main
1
2
       import "fmt"
3
4
5
      func main(){
          var numPtr *int
6
          fmt.Println(numPtr)
7
          numPtr = new(int)
8
          fmt.Println(numPtr)
9
10
          i := 100
          numPtr = &i
11
          fmt.Println(*numPtr)
12
13
 \Psi
     <nil>
:÷
     0xc00000a0a8
     100
```

```
package main
       import "fmt"
      func main(){
           var p *[]int
           fmt.Println(p)
           a := make([]int, 5)
10
           fmt.Println(a)
11
12
           p = &a
           fmt.Println(*p)
14
           a[0] = 10
           a[1] = 20
15
           fmt.Println(*p)
16
17
          NY YO SCLOP CALLS?
          <nil>
          [0 \ 0 \ 0 \ 0 \ 0]
          [0 0 0 0 0]
          [10 20 0 0 0]
```

포인터

```
package main
       import "fmt"
 5
       func main(){
           var a =[5]int\{1,2,3,4,5\}
           fmt.Println(a)
           changeVal(a)
           fmt.Println(a)
10
           changeValPoint(&a)
           fmt.Println(a)
11
12
       func changeVal(val [5]int){
13
           val[0] = 10
14
15
16
17
       func changeValPoint(val *[5]int){
           val[0] = 10
18
19
20
               [1 2 3 4 5]
               [1 2 3 4 5]
               [10 2 3 4 5]
```

```
package main
                  2
                        import "fmt"
                  3
                  5
                       func main(){
                            var p *int = new(int)
                  6
                  7
                            p++
                  8
                            p = 0x01
                  9
                 10
                            fmt.Println(p)
                 11
                 12
    # command-line-arguments
    .\2day.go:8:3: invalid operation: p++ (non-numeric type *int)
:;
    .\2day.go:9:4: cannot use 1 (type int) as type *int in assignment
```

- 구조화된 데이터를 관리하는 자료구조
- 다양한 타입을 하나의 타입으로 관리 가능
- Class, Object와 비슷한 개념
- Struct 키워드를 사용

- Golang 구조체
 - Custom Data Type을 표현
 - 필드(맴버) 데이터만 가질 수 있음
 - OOP 구현을 위해 메서드를 분리해서 사용
 - 임베딩을 통한 상속 효과
- Golang 메서드
 - OOP 구현을 Golang의 고유 방식으로 지원
 - Value receiver, Pointer Receiver 방식

```
package main
         import "fmt"
   3
   4
         type Person struct {
             gender bool
             age int
   8
             name string
   9
         func main(){
  10
  11
             var p Person
  12
             p.gender = true
  13
             p.age = 33
  14
             p.name = "taehoon"
  15
             fmt.Println(p)
  16
  17
    {true 33 taehoon}
=
```

```
package main
        import "fmt"
  3
        type Person struct {
  5
            gender bool
            age int
            name string
        } •
        func main(){
 10
 11
            var pp *Person
 12
            pp = new(Person)
 13
            fmt.Println(*pp)
 14
 15
            pp.gender = false
            pp.age = 23
 16
            pp.name = "roh"
 17
            fmt.Println(*pp)
 18
 19
 20
    <4 go setup calls>
    {false 0 }
    {false 23 roh}
=
```

```
package main
       import "fmt"
3
4
      type Person struct {
5
           gender bool
           age int
           name string
8
9
      func main(){
10
11
           p1 := Person{}
          fmt.Println(p1)
12
           p2 := new(Person)
13
           fmt.Println(*p2)
14
           p3 := make([]Person, 3)
15
          fmt.Println(p3)
16
17
18
     {false 0 }
     {false 0 }
     [{false 0 } {false 0 } {false 0 }]
≟
```

```
package main
2
       import "fmt"
 3
 4
                                                          {true 0 a}
       type Person struct {
                                                          {true 0 a}
           gender bool
                                                          {false 1 b} {false 1 b}
           age int
                                                          [{false 0 } {false 0 } {false 0 }]
           name string
8
                                                          [{false 1 b} {false 1 b} {false 0 }]
       func main(){
10
           p1 := Person{}
11
12
           p1.name = "a";p1.age = 0; p1.gender = true;
           fmt.Println(p1)
13
           p2 := new(Person)
14
15
           p2 = &p1
16
           fmt.Println(*p2)
17
           p2.name = "b";p2.age = 1; p2.gender = false;
           fmt.Println(p1, *p2)
18
           p3 := make([]Person, 3)
19
           fmt.Println(p3)
20
           p3[0] = p1
21
           p3[1] = *p2
22
           fmt.Println(p3)
23
24
25
```

₽

```
package main
               import "fmt"
         3
               type Person struct {
                    gender bool
                    age int
                   name string
         9
        10
               func main(){
                   p1 := Person{ gender: true, age: 10, name: "name"}
        11
                   fmt.Println(p1)
        12
                   p2 := Person{ gender: "test", age: 1, name: false}
        13
                   fmt.Println(p2)
        14
        15
        16
<3 go setup calls>
# command-line-arguments
.\2day.go:13:15: cannot use "test" (type untyped string) as type bool in field value
```

.\2day.go:13:25: cannot use false (type untyped bool) as type string in field value

```
package main
 2
 3
       import "fmt"
       type Person struct {
           gender bool
                  int
           age
           name
                  string
 9
10
      func main() {
12
           p1 := Person{ gender: true, age: 10, name: "name"}
           fmt.Println(p1)
13
           p2 := Person{gender: false,
14
15
               age: 20,
               name: "ttt",
16
17
           fmt.Println(p2)
18
19
20
                      {true 10 name}
                      {false 20 ttt}
                 =
```

```
package main
2
3
       import "fmt"
4
       type Person struct {
5
           gender bool
6
7
           age
                  int
8
           name
                 string
9
10
      func initPerson(gender bool, age int, name string) *Person{
11
           return &Person{ gender: gender, age: age, name: name}
12
13
14
       func main() {
15
           p1 := Person{ gender: true, age: 10, name: "name"}
16
           fmt.Println(p1)
17
18
           p2 := initPerson( gender: true, age: 10, name: "test")
19
           fmt.Println(p2)
20
           p3 := &Person{ gender: false, age: 20, name: "test2"}
22
           fmt.Println(p3)
23
24
25
```

생성자(Constructor) 처럼 사용 가능

구조체 메서드

```
package main
1
2
3
       import "fmt"
4
5
       type Rectangle struct {
           width int
6
7
           height int
8
9
       func (rect Rectangle) area() int{
10
           return rect.width * rect.width
11
12
13
14
15
      func main() {
           rect := Rectangle{ width: 10, height: 20}
16
17
          fmt.Println(rect.area())
18
19
           100
```

```
package main
2
       import "fmt"
 3
       type Rectangle struct {
           width int
           height int
 8
       func (rect Rectangle) area() int{...}
13
       func (rect *Rectangle) fix(w, h int) {
           rect.width = w
14
15
           rect.height = h
16
17
18
       func main() {
19
           rect := Rectangle{ width: 10, height: 20}
20
           fmt.Println(rect)
21
           rect.fix( w: 20, h: 30)
           fmt.Println(rect)
23
24
25
                   {10 20}
                   {30 20}
              ₽
```

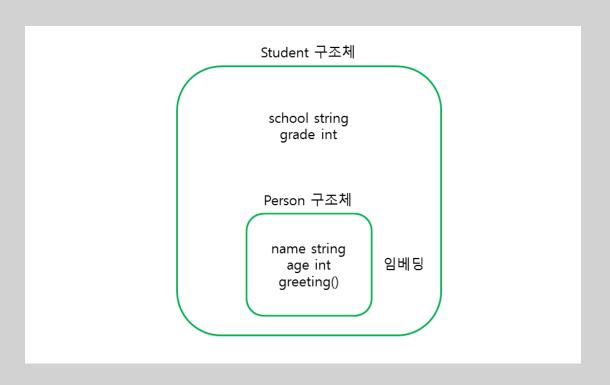
구조체 메서드

```
package main
2
      import "fmt"
      type Rectangle struct {
5
          width int
          height int
8
      func (rect *Rectangle) fix(w, h int) {
          rect.width = w
10
          rect.height = h
                                                  {10 20}
                                                  {20 30}
      func (rect Rectangle) fixVal(){
14
                                                  {20 30}
          rect.width = 100
          rect.height = 200
19
     func main() {
          rect := Rectangle{ width: 10, height: 20}
          fmt.Println(rect)
          rect.fix( w: 20, h: 30)
          fmt.Println(rect)
          rect.fixVal()
          fmt.Println(rect)
```

```
package main
 2
 3
       import "fmt"
 4
       type Rectangle struct {
 5
           width int
 6
           height int
 8
 9
       func (_ Rectangle) dummy(){
10
           fmt.Println( a...: "dummy")
11
12
13
14
       func main() {
15
           rect := Rectangle{ width: 10, height: 20}
16
           fmt.Println(rect)
17
           rect.dummy()
18
19
20
                        {10 20}
                        dummy
                   =
```

구조체 임베딩

```
package main
2
       import "fmt"
3
4
       type Person struct {
5
           name string
6
           age int
8
9
       func (p *Person) greeting() {
10
11
           fmt.Println( a...: "Hello~")
12
13
       type Student struct {
14
15
                  Person
           school string
16
           grade int
17
18
19
       func main() {
20
           var s Student
21
22
           s.p.greeting()
23
          THE GO SCLOP COLLS?
           Hello~
```



출처: http://pyrasis.com/book/GoForTheReallyImpatient/Unit31

구조체 임베딩

```
package main
3
       import "fmt"
       type Person struct {
5
           name string
           age int
8
9
       func (p *Person) greeting() {
10
11
           fmt.Println( a...: "Hello~")
12
13
14
       type Student struct {
           Person
15
           school string
16
17
           grade int
18
19
      func main() {
20
21
           var s Student
22
23
           s.Person.greeting()
           s.greeting()
24
25
         Hello~
         Hello~
```

```
package main
       import "fmt"
      type Person struct {
          name string
           age int
8
10 ■ | func (p *Person) greeting() {
          fmt.Println( a...: "Hello~")
     type Student struct {
           Person
          school string
16
           grade int
18
19
20 of func (s * Student) greeting(){
          fmt.Println( a...: "Hello Student~")
23
24 ▶ ⊟func main() {
          var s Student
          s.Person.greeting()
          s.greeting()
28
29
30
```

Hello~ Hello Student~