Chapter 5 fmt 패키지를 이용한 텍스트 입출력

날짜	@October 13, 2021
발표자	김윤서

5.1 표준 입출력

Go 언어에서는 fmt패키지를 사용해서 간편하게 표준 입출력 처리를 할 수 있다.

(표준 입출력스트림은 os 패키지의 Stdin, Stdout, Stderr을 제공, 입출력 스트림 처리는 io.Reader, io.Writer 인터페이스로 처리)

5.1.1 fmt 패키지

```
import "fmt"
```

3가지 표준 출력용 함수

- **Print()**
- **Println()**
- **Printf()**

```
//ch5/ex5.1/ex5.1.go
package main

import "fmt"

func main() {
  var a int = 10
  var b int = 20
  var f float64 = 32799438743.8297

  fmt.Print("a:", a, "b:", b)              // ❶
```

```
  fmt.Println("a:", a, "b:", b, "f:", f)    // ❷
  fmt.Printf("a: %d b: %d f:%f\n", a, b, f) // ❸
}
```

출력값

```
a:10b:20a: 10 b: 20 f: 3.27994387438297e+10
a: 10 b: 20 f:32799438743.829700
```

2번의 f값이 지수형태로 출력된 이유: 실숫값의 기본 서식이 %f가 아닌 %g이기 때문.

5.1.2 서식문자

자주 사용 : **%d, %f, %s**

마법의 서식 문자 : **%v** (기본 서식에 맞춰 출력)

그외

- %T : 타입 출력
- %b : 2진수로 출력
- %c : 유니코드 문자 출력 (정수만 가능)
- %e : 지수 형태로 출력
- %f : 실숫값 그대로 출력
- %g : 값이 큰 실숫값은 지수 형태로, 작은 실숫값은 실숫값 그대로
- %q : 특수문자 기능 동작 X 그대로 출력
- %p : 메모리 주솟값 출력

5.1.3 최소 출력 너비 지정

- 최소 출력 너비 지정 : 예) %5d = 최소 5칸을 사용해 출력
- 공란 채우기 : 너비 앞에 0을 붙이면 빈자리를 0으로 채움
- 왼쪽 정렬하기 : 마이너스 -를 붙이면 왼쪽을 기준선 삼아 출력

```
//ch5/ex5.2/ex5.2.go
package main

import "fmt"
```

```
func main() {
  var a = 123
  var b = 456
  var c = 123456789

  fmt.Printf("%5d, %5d\n", a, b)    // ❶ 최소 너비보다 짧은 값 너비 지정
  fmt.Printf("%05d, %05d\n", a, b)  // ❷ 최소 너비보다 짧은 값 0 채우기
  fmt.Printf("%-5d, %-05d\n", a, b) // ❸ 최소 너비보다 짧은 값 왼쪽 정렬

  fmt.Printf("%5d, %5d\n", c, c)    // ❹ 최소 너비보다 긴 값 너비 지정
  fmt.Printf("%05d, %05d\n", c, c)  // ❺ 최소 너비보다 긴 값 0 채우기
  fmt.Printf("%-5d, %-05d\n", c, c) // ❻ 최소 너비보다 긴 값 왼쪽 정렬
}
```

출력값

```
  123,   456
00123, 00456
123  , 456
123456789, 123456789
123456789, 123456789
123456789, 123456789
```

5.1.4 실수 소수점 이하 자릿수

- %f : 예) %5.2f = 최소 너비 5칸 소수점 이하 값 2개 출력
- %g : 정수부와 소수점 이하 숫자 포함해 출력 숫자 제한, 정해진 길이로 정수부 표현 못하면 지수 표현으로 전환, 기본 길이 6개, 예) %5.3g = 최소 너비 5칸 소수점 이하 포함 총 숫자 3개

```
//ch5/ex5.3/ex5.3.go
package main

import "fmt"

func main() {
  var a = 324.13455
  var b = 324.13455
  var c = 3.14

  fmt.Printf("%08.2f\n", a) // ❶ 최소너비:8 소수점이하:2자리 0을 채움.
  fmt.Printf("%08.2g\n", b) // ❷ 최소너비:8 총숫자: 2자리 0을 채움
  fmt.Printf("%8.5g\n", b)  // ❸ 최소너비:8 총숫자: 5자리
  fmt.Printf("%f\n", c)     // ❹ 소수점이하 6자리까지 출력
}
```

출력값

```
00324.13
03.2e+02
  324.13
3.140000
```

1. 최소 너비 8카 소수점 숫자 2개, 공란 0으로 ⇒ 00324.13
2. 총 출력 숫자 2개, 324는 2개로 표현 불가, 지수표현으로 전환, 공란 0으로 ⇒ 03.2e+02
3. 총 출력 숫자 5개, 324는 5개로 표현 가능 ⇒ 324.13
4. 디폴트 소수점 이하 6개 (%f == %.6f) ⇒ 3.140000

5.1.5 특수문자

- \n
- \t
- \\ : \ 출력
- \" : " 출력

```
//ch5/ex5.4/ex5.4.go
package main

import "fmt"

func main() {
  str := "Hello\tGo\t\tWorld\n\"Go\"is Awesome!\n" // ❶ 문자열

  fmt.Print(str)        // ❷ 문자열을 기본 서식으로 출력
  fmt.Printf("%s", str) // ❸ 문자열을 %s 서식으로 출력
  fmt.Printf("%q", str) // ❹ 문자열을 %q 서식으로 출력
}
```

출력값

```
Hello Go    World
"Go"is Awesome!
Hello Go    World
```

```
"Go"is Awesome!
"Hello\tGo\t\tWorld\n\"Go\"is Awesome!\n"
```

2. 문자열의 기본 서식은 %s
3. 특수문자 제대로 동작
4. %q : 특수문자 기능을 잃고 문자 자체로 동작

5.2 표준 입력

5.2.2 Scan()

Scan()의 인수 : 값을 채워넣을 변수들의 메모리 주소

```
//ch5/ex5.5/ex5.5.go
package main

import "fmt"

func main() {
  var a int // ❶ 값을 저장할 변수
  var b int

  n, err := fmt.Scan(&a, &b) // ❷ 입력 두 개 받기
  if err != nil {            // ❸ 에러가 발생하면 에러 코드 출력
    fmt.Println(n, err)
  } else { // ❹ 정상 입력되면 입력값 출력
    fmt.Println(n, a, b)
  }
}
```

출력값

```
3 4
2 3 4

Hello 4
0 expected integer

4 Hello
1 expected integer
```

2. n : 성공적으로 입력한 값 개수, err : 입력 시 발생한 에러

3. err ≠ nil 에러 발생 → 에러값 출력
4. 정상 입력 → 받은 값 출력

"0 expected integer"인 이유 : 첫 번째 입력을 잘못 입력하면 두 번째 입력을 받지 않고 함수가 에러를 반환한다.

5.2.3 Scanf()

Scanf() : 서식에 맞춘 입력 → 힘들기 때문에 Scan()이나 Scanln() 추천

```
//ch5/ex5.6/ex5.6.go
package main

import "fmt"

func main() {
  var a int
  var b int

  n, err := fmt.Scanf("%d %d\n", &a, &b) // ❶ 입력 두 개 받기
  if err != nil {
    fmt.Println(n, err)
  } else {
    fmt.Println(n, a, b)
  }
}
```

5.2.4 Scanln()

Scanln() : 한 줄을 입력 받아서 인수로 들어온 변수 메모리 주소에 값 채움

Scan()과 다른 점 : 마지막 입력값 이후 반드시 enter키로 입력 종료해야함

```
//ch5/ex5.7/ex5.7.go
package main

import "fmt"

func main() {
  var a int
  var b int

  n, err := fmt.Scanln(&a, &b) // ❶ 값을 입력받습니다.
  if err != nil {              // 에러 발생 시
    fmt.Println(err) // 에러를 출력합니다.
  } else {
    fmt.Println(n, a, b)
```

```
  }
}
```

5.3 키보드 입력과 Scan() 함수의 동작 원리

사용자가 표준 입력 장치로 입력하면 입력 데이터는 컴퓨터 내부에 표준 입력스트림이라는 메모리 공간에 임시 저장된다.

표준 입력 스트림 작동 원리

FIFO 구조

```
var a, b int
fmt.Scanln(&a, &b)
```

키보드 ⇒ \n4 olleH ⇒ 읽기 H ⇒ 숫자가 아니면 Error!

여러 번 Scan() 함수 호출 시 위 문제에서 벗어나려면 입력에 실패한 경우 표준 입력 스트림을 지워야 한다.

```
//ch5/ex5.8/ex5.8.go
package main

import (
  "bufio" // ❶ io를 담당하는 패키지
  "fmt"
  "os" // 표준 입출력 등을 가지고 있는 패키지
)

func main() {
  stdin := bufio.NewReader(os.Stdin) // ❷ 표준 입력을 읽는 객체

  var a int
  var b int

  n, err := fmt.Scanln(&a, &b)
  if err != nil { // 에러 발생 시
    fmt.Println(err)       // 에러 출력
    stdin.ReadString('\n') // ❸ 표준 입력 스트림 지우기
  } else {
    fmt.Println(n, a, b)
  }
  n, err = fmt.Scanln(&a, &b) // ❹ 다시 입력받기
  if err != nil {
    fmt.Println(err)
  } else {
    fmt.Println(n, a, b)
```

```
  }
}
```

1. bufio 패키지는 Reader 객체 제공 (func NewReader(rd io.Reader) *Reader
2. os.Stdin : 표준 입력 스트림
3. stdin.ReadString('\n) : 줄바꿈 문자가 나올 때까지 읽어서 표준 입력 스트림을 비운다.
4. 다시 입력받는다. 기존에 남아있는 값이 비워져서 입력의 실패를 피했다.

연습문제

1.

```
00345
 3.14
```

2. Scanln() 함수로 입력을 받기 위해서는 값을 채울 변수의 메모리 주소를 인수로 받아야한다. a, b 대신 &a, &b로 수정해야한다.
3.

```
package main

import "fmt"

func main(){
  var a = 123
  var b int = 4567
  f := 3.14159269

  fmt.Printf("%6d\n", a)
  fmt.Printf("%06d\n", b)
  fmt.Printf("%6.2f\n", f)
}
```