Chapter 7 함수

날짜	@November 3, 2021
발표자	김하연

7.1 함수 정의

함수는 함수 키워드, 함수명, 매개변수, 반화 타입, 함스 코드 블록으로 구성된다.

func 키워드를 사용해서 함수 정의를 알린다.

그 뒤 함수명이 온다. 함수명의 명명 규칙은 변수명과 같다.

 소괄호 안에 매개변수를 넣는다. 매개변수는 함수 코드 수행 시 필요한 입력값이다. 필요하지 않으면 비워둔다.

반환 타입이 온다. 반환하는 값이 있으면 적고, 아니면 비워둔다.

중괄호로 함수 코드 블록을 표시한다. Go 언어에는 함수 코드 블록의 시작을 알리는 중괄호 {가 함수를 정의하는 라인과 항상 같은 줄에 있어야 합니다.

```
package main

import "fmt"

func Add(a int, b int) int {
  return a + b
}

func main() {
  c := Add(3, 6)
  fmt.Println(c)
}
```

출력값

```
9
```

7.2 함수를 호출하면 생기는 일

함수를 호출할 때 입력하는 값을 argument 또는 인수라고 한다. 반면 함수가 외부로부터 입력받는 변수를 parameter 또는 매개변수라고 한다.

함수를 호출하며 입력한 값은 실제 함수에 전달될 때 보낸 값을 그대로 사용하는 것이 아니라 값을 복사해 사용하게 된다.

```
c := Add(3, 6)  //함수 호출
```

```
func Add(a int, b int) int { //매개변수 생성 및 초기화
  //a와 b를 더한 결과를 반환
  return a+b  //값 반환(복사)
} //로컬 변수 삭제
```

1. Add() 함수 호출
2. 매개변수 선언 → 입력한 인숫값을 복사 (a, b, 두 변수에 초깃값으로 3과 6을 대입)
3. return 키워드로 함수 결과 반환 → 값 전달
4. 반환된 값은 함수가 호출된 곳을 대체하는 것과 같다
5. 호출한 함수 종료 시 함수에서 사용한 지역 변수에 접근 불가

```
c := Add(3, 6)
c := 9
```

6. c에 반환값이 대입(복사)된다.

핵심 포인트 : 인수는 매개변수로 복사된다. 배개변수와 함수 내에서 선언된 변수는 함수가 종료되면 변수 범위를 벗어나서 접근하지 못한다.

7.3 함수는 왜 쓰나?

함수를 사용해서 반복 사용되는 코드를 묶을 수 있다. 중복 코드를 제거하여 코드를 간결하게 만들 수 있다.

```
package main

import "fmt"

func main() {
  math := 80
  eng := 74
  history := 95
  fmt.Println("김일등 님 평균 점수는", (math+eng+history)/3, "입니다.")

  math = 88
  eng = 92
  history = 53

  fmt.Println("송이등 님 평균 점수는", (math+eng+history)/3, "입니다.")

  math = 78
  eng = 73
  history = 78

  fmt.Println("박삼등 님 평균 점수는", (math+eng+history)/3, "입니다.")
}
```

출력값

```
김일등 님 평균 점수는 83 입니다.
송이등 님 평균 점수는 77 입니다.
박삼등 님 평균 점수는 76 입니다.
```

- 각 변수에 점수를 입력하는 코드와 점수를 출력하는 코드가 3번 반복되었다.
- 만약 학생이 1000명? 국어 점수가 추가된다면? → 비효율적이다

```
package main

import "fmt"

func printAvgScore(name string, math int, eng int, history int) {
  total := math + eng + history
  avg := total / 3
  fmt.Println(name, "님 평균 점수는", avg, "입니다.")
}

func main() {
  printAvgScore("김일등", 80, 74, 95)
  printAvgScore("송이등", 88, 92, 53)
```

```
  printAvgScore("박삼등", 78, 73, 78)
}
```

출력값

```
김일등 님 평균 점수는 83 입니다.
송이등 님 평균 점수는 77 입니다.
박삼등 님 평균 점수는 76 입니다.
```

- PrintAvgScore() 정의, 이름과 각 성적을 입력받아 이름과 평균 점수를 출력한다
- 반복 호출하여 처리한다.

자주 사용되거나 변경 가능성이 있는 코드 블록을 묶어서 함수를 만들면 효율적으로 코딩할 수 있고 추후 프로그램 변경 요구에도 간단히 대처할 수 있다. 또 관련된 코드를 묶어서 이름을 부여하기 때문에 코드를 읽기에도 훨씬 편해진다.

7.3.1 멀티 반환 함수

- 함수는 값을 여러개 반환할 수 있다. → 반환 타입들을 소괄호로 묶어서 표현한다.

```
package main

import "fmt"

func Divide(a, b int) (int, bool) {
  if b == 0 {
    return 0, false //제수가 0일 때 반환
  }
  return a / b, true //제수가 0이 아닐 때 반환
}

func main() {
  c, success := Divide(9, 3) //제수가 0이 아닌 경우
  fmt.Println(c, success)
  d, success := Divide(9, 0) //제수가 0인 경우
  fmt.Println(d, success)
}
```

출력값

```
3 true
0 false
```

- (a int, b int) 같이 매개변수 타입이 같으면 (a, b int)처럼 표현 가능
- 나눗셈 제수가 0이면 0과 false 반환, 0이 아니면 나눗셈 결과와 true 반환

```
c, success := Divide(9, 3)
c, success := 3, true
```

7.3.2 변수명을 지정해 반환하기

함수 선언부에 반환 타입을 적을 때 변수명까지 지정해주면, return문으로 해당 변수를 명시적으로 반환하지 않아도 값을 반환할 수 있다.

```
package main

import "fmt"

func Divide(a, b int) (result int, success bool) {  //반환할 변수명 명시
  if b == 0 {
    result = 0
    success = false
    return //명시적으로 반환할 값을 지정하지 않은 return문
  }
  result = a / b
  success = true
  return
}

func main() {
  c, success := Divide(9, 3)
  fmt.Println(c, success)
  d, success := Divide(9, 0)
  fmt.Println(d, success)
}
```

출력값

```
3 true
0 false
```

- 함수 선언 시 지정한 반환할 변수 result, success는 함수 내부에서 변수로 동작한다.
- 함수 결과를 반환할 때 명시적으로 result, success를 지정하지 않았지만 두 값이 반환된다.

주의 : 반환할 변수의 이름을 지정할 경우 모든 반환 변수의 이름을 지정해야한다. 모두 지정하거나 모두 지정하지 않거나!

7.4 재귀 호출

재귀 호출이란 함수 안에서 자기 자신 함수를 다시 호출하는 것을 말한다.

```
package main

import "fmt"

func printNo(n int) {
  if n == 0 { //재귀 호출 탈출 조건
    return
  }
  fmt.Println(n)
  printNo(n - 1)          //재귀 호출
  fmt.Println("After", n) //재귀 호출 이후 출력
}

func main() {
  printNo(3)
}
```

출력값

```
3
2
1
After 1
After 2
After 3
```

- printNo() 함수 호출
- 탈출 조건인지 확인 → n이 0이 아니면 printNo() 함수 다시 호출
- 호출 순서: printNo(3), printNo(2), printNo(1), printNo(0)
- return 순서: printNo(0), printNo(1), printNo(2), printNo(3)

주의 : 재귀 호출을 사용할 때는 항상 탈출 조건을 정해야 한다. 재귀 호출이 종료되는 시점을 명확히 하지 않으면 재귀 호출이 무한히 반복되어 프로그램이 비정상 종료된다.

연습문제

1.

```
func Multiple (a, b int) int {
  return a * b
}
```

2.

```
start AAA()
BBB()
end AAA()
```

3.

```
package main

import "fmt"

func F(n int) int {
  if n < 2 {
    return n
  }
  return F(n-2) + F(n-1)
}

func main() {
  fmt.Println(F(9))
}
```