Chapter 12 배열

날짜	@November 17, 2021
발표자	김선영

12.1 배열

배열은 같은 타입의 데이터들로 이루어진 타입. 배열을 이루는 각 값은 요소, 요소를 가리키는 위치값은 인덱스.

```
var 변수명 [요소 개수]타입
```

```
var t [5]float64
```

```
package main

import "fmt"

func main() {
  var t [5]float64 = [5]float64{24.0, 25.9, 27.8, 26.9, 26.2}

  for i := 0; i < 5; i++ {
    fmt.Println(t[i])
  }
}
```

출력문

```
24
25.9
27.8
26.9
26.2
```

- float64 배열 선언 & 초기화. 중괄호 {} 를 이용해 각 요소의 값을 초기화.
- 배열 길이가 5이므로 5회 반복 for문 → i값에 따라 배열 요소에 접근해 값 출력
- 다른 값과 달리 24.0은 24로 출력 → 포맷을 지정하지 않은 경우 실수는 최소 소숫점 자릿수로 표시
- **배열의 인덱스는 0부터 시작**, 마지막 요소 인덱스는 배열 길이 - 1
- **배열의 인덱스 범위를 벗어나서 접근하면 에러 발생**

12.2 배열 사용법

배열 변수 선언과 초기화 방법에 대해서 알아보자

12.2.1 배열 변수 선언 및 초기화

```
var nums [5]int
```

int 타입 요소 5개 갖는 배열 nums. 별도로 초깃값을 지정하지 않으면 각 요솟값은 int 타입의 기본값인 0으로 초기화

```
days := [3]string{"monday", "tuesday", "wednesday"}
```

string 타입 요소 3개 갖는 배열 days.

```
var temps [5]float64 = [5]float64{24.3, 26.7}
```

float64 타입 요소 5개 갖는 배열 temps. 1, 2번째 요솟값 초기화, 나머지는 float64 기본값인 0.0으로 초기화

```
var s = [5]int{1:10, 3:30}
```

int 타입 요소 5개 갖는 배열 s. 인덱스가 1인 요솟값 10, 3인 요솟값 30으로 초기화, 나머지는 0으로 초기화

```
x := [...]int{10, 20, 30}
```

...를 사용해 배열 요소 개수 생략 가능. 이때 배열 요소 개수는 초기화되는 요소 개수와 같음.

12.2.1 배열 변수 선언 시 개수는 항상 상수

배열 선언 시 개수는 항상 상수. 변숫값을 사용할 수 없음.

```
package main

const Y int = 3

func main() {
  x := 5
  a := [x]int{1, 2, 3, 4, 5}

  b := [Y]int{1, 2, 3}

  var c [10]int
}
```

출력문

```
.\ex12.2.go:7:7: non-constant array bound x
```

- 변수로 선언된 x는 배열 길이로 사용 불가
- Y는 상수로 선언됐기 때문에 b 배열 선언 시 에러 발생하지 않음

12.2.3 배열 요소 읽고 쓰기

```
package main

import "fmt"

func main() {
  nums := [...]int{10, 20, 30, 40, 50}

  nums[2] = 300

  for i := 0; i < len(nums); i++ {
    fmt.Println(nums[i])
  }
}
```

출력문

```
10
20
300
40
50
```

- len() 함수를 이용해 배열 길이를 알 수 있음

12.2.3 range 순회

for 반복문에서 range 키워드를 이용해 배열 요소 순회

```
package main

import "fmt"

func main() {
  var t [5]float64 = [5]float64{24.0, 25.9, 27.8, 26.9, 26.2}

  for i, v := range t {
    fmt.Println(i, v)
  }
}
```

출력문

```
0 24
1 25.9
2 27.8
3 26.9
4 26.2
```

- range는 배열의 각 요소를 순회하면서 인덱스와 요솟값 두 값을 반환
- 인덱스가 필요 없고 요솟값만 필요하면 밑줄_을 이용해서 이덱스를 무효화할 수 있음

```
for _, v:= range t{
  fmt.Println(v)
}
```

- 선언하고 사용하지 않는 변수가 있으면 컴파일 에러 발생. 따라서 range을 사용할 때 인덱스를 사용하지 않으면 밑줄_을 사용해서 반드시 무효화함
- range 순회는 배열 뿐 아니라 문자열, 슬라이스, 맵 채널 등에서 사용 가능

12.3 배열은 연속된 메모리

배열을 선언하면 컴퓨터는 연속된 메모리 공간을 확보한다.

예) var a[10]int32 배열 선언 → 컴퓨터는 int32값 10개 저장할 수 있는 연속된 메모리 공간을 찾아 할당 (4바이트 * 10)

컴퓨터는 배열 시작 주소에 '인덱스 x 타입 크기'를 더해서 찾아간다.

- 요소 위치 = 배열 시작 주소 + (인덱스 x 타입 크기)

예) a 배열 시작 주소가 100번지라면 a[3] 주소는 100 + (3 * 4) = 112번지가 된다.

a[3] = 300 → 112부터 4바이트 메모리 공간에 300을 대입

배열의 핵심

1. 배열은 연속된 메모리다.
2. 컴퓨터는 인덱스와 타입 크기를 사용해서 메모리 주소를 찾는다.

12.3.1 배열 복사

```
package main

import "fmt"

func main() {
  a := [5]int{1, 2, 3, 4, 5}
  b := [5]int{500, 400, 300, 200, 100}

  for i, v := range a { // 배열 a 원소 출력
    fmt.Printf("a[%d] = %d\n", i, v)
  }

  fmt.Println()
  for i, v := range b { // 배열 b 원소 출력
    fmt.Printf("b[%d] = %d\n", i, v)
  }

  b = a // a 배열을 b변수에 복사

  fmt.Println()
```

```
  for i, v := range b { // 배열 b 원소 출력
    fmt.Printf("b[%d] = %d\n", i, v)
  }
}
```

출력문

```
a[0] = 1
a[1] = 2
a[2] = 3
a[3] = 4
a[4] = 5

b[0] = 500
b[1] = 400
b[2] = 300
b[3] = 200
b[4] = 100

b[0] = 1
b[1] = 2
b[2] = 3
b[3] = 4
b[4] = 5
```

- a 배열의 모든 요솟값을 b 배열에 복사
- Go 언어에서 대입 연산자는 우변의 값을 좌변의 메모리 공간에 복사. 복사되는 크기는 타입 크기와 같다. 배열의 대입도 마찬가지.
- a타입은 5 * 8 = 40 바이트, b타입도 40 바이트

```
var a = [5]int{1, 2, 3, 4, 5}
var b = [5]float32{100, 200, 300, 400, 500}

b = a //에러 발생
```

- Go언어에서 모든 연산자의 각 항의 타입은 같아야 함. b = a 코드는 a, b의 타입이 같지 않기 때문에 에러 발생

12.4 다중 배열

다중 배열이란 중첩된 배열. 이중, 삼중 배열처럼 여러번 중첩해서 사용 가능.

이중 배열은 X, Y 좌표계의 위치 데이터들을 위해 주로 사용되어 이차원 배열이라 부르고 삼중 배열은 삼차원 배열이라고 부르기도 함.

var a[5]int는 int값을 5개 가지는 배열.

var b[2][5]int는 [5]int가 두 개인 배열. b[0]은 첫번째 [5]int 배열, b[1]은 두번째 [5]int 배열

배열은 몇 개가 중첩됐든 **[개수]타입** 형태

[2][5][100][200]int : [5][100][200]int 타입이 2개인 배열

이중 배열의 초기화

```
var b[2][5]int{ //b배열용 {}
  {1, 2, 3, 4, 5}, //b[0] 초기화용 {}
  {6, 7, 8, 9, 10},//b[1] 초기화용 {}
}
```

```
package main

import "fmt"

func main() {
  a := [2][5]int{ //이중 배열 선언
    {1, 2, 3, 4, 5},
    {5, 6, 7, 8, 9},
  }
  for _, arr := range a { // arr값은 순서대로 a[0]의 배열 a[1]의 배열
    for _, v := range arr { // v값은 순서대로 a[0]과 a[1] 배열의 각 원소
      fmt.Print(v, " ") // v값 출력
    }
    fmt.Println()
  }
}
```

출력문

```
1 2 3 4 5
5 6 7 8 9
```

- 초기화 시 닫는 중괄호 } 가 마지막 요소와 같은 줄에 있지 않은 경우 마지막 항목 뒤에 쉼표 , 를 찍어줘야함.

```
a := [2][5]int{
  {1, 2, 3, 4, 5},
  {5, 6, 7, 8, 9} }  // 쉼표 없음
```

```
a := [2][5]int{
  {1, 2, 3, 4, 5},
  {5, 6, 7, 8, 9},  // 쉼표 찍어야함
}
```

```
a := [2][5]int{
  {1, 2, 3, 4, 5},
    {5, 6, 7, 8, 9,  //닫는 중괄호가 같은 줄에 있지 않아서 쉼표
  }} //닫는 중괄호가 같은 줄에 있어서 쉼표 없음
```

- 추후 항목이 늘어날 경우 쉼표를 찍지 않아서 생길 수 있는 오류를 방지하기 위해 존재하는 규칙

12.4.1 배열 크기

배열 크기는 타입 크기 요소 개수가 된다.

- **배열 크기 = 타입 크기 x 항목 개수**

이중 배열 크기?

[2][5]int = [2]([5]int) = 2 * (5 * 4) = 80
[3][2][5]int = 3 * 80 = 240

연습문제

1. 6
2. 3 * 2 * 5 * 8 = 240 바이트
3. 4

ChangeArray() 함수로 인수로 a값 복사, arr와 a는 서로 다른 메모리 주소를 가진 다른 배열이므로 arr[3] 값을 바꿔도 a[3]값은 그대로임.