Chapter 8 상수

날짜	@November 3, 2021
발표자	김하연

8.1 상수 선언

- 상수는 변하지 않는 값이다. 값을 수시로 바꿀 수 있는 변수와 달리 상수는 초기화된 값이 변하지 않는다.
- 기본 타입값들만 상수로 선언 가능하다. 구조체, 배열 등 기본 타입이 아닌 타입에는 상수를 사용할 수 없다.
- 가능한 타입: 불리언, 룬(rune), 정수, 실수, 복소수, 문자열
- 상수 선언 키워드: const

```
const ConstValue int = 10
```

- 상수명 규칙은 변수명과 같다.
- 값으로만 동작하기 때문에 대입문의 좌변에 올 수 없다.

```
package main

import "fmt"

func main() {
  const C int = 10 //상수 선언

  var b int = C * 20
  C = 20           //에러 발생 - 상수는 대입문 좌변에 올 수 없다.
  fmt.Println(&C) //상수 주소 출력
}
```

출력값

```
.\ex8.1.go:9:4: cannot assign to C (declared const)
```

- 상수를 대입문 좌변에 두어 에러가 발생한다.
- &연산자를 상수 앞에 사용하면 상수의 메모리 주솟값을 접근할 수 없기 때문에 에러가 발생한다.

8.2 상수는 언제 사용하나?

상수는 보통 1) 변하면 안되는 값에 사용하고 2) 코드값을 통해서 숫자에 의미를 부여할 때 사용한다.

8.2.1 변하면 안 되는 값에 상수 사용하기

- 상수로 변하는 값에 이름을 부여하면 매번 값을 쓰지 않고 편리하게 이용 가능하다.
- 예) 원주율 π = 3.141592... → 상수 선언
- 고정 불변의 값을 여러 번 사용할 때 변수 대신 상수로 정의하는 것이 더 안전하고 확실하다.

```
package main

import "fmt"

func main() {
  const PI1 float64 = 3.141592653589793238
  var PI2 float64 = 3.141592653589793238

  // PI1 = 4 //에러 발생
  PI2 = 4

  fmt.Printf("원주율: %f\n", PI1)
  fmt.Printf("원주율: %f\n", PI2)
}
```

출력값

```
원주율: 3.141593
원주율: 4.000000
```

- 상수를 사용하면 상수를 변경하는 시도를 할 때 컴파일 단계에서 에러가 출력되므로 의도치 않은 결과를 미연에 방지할 수 있다.

8.2.2 코드값으로 사용하기

- 코드값이란 어떤 숫자에 의미를 부여하는 것이다.
- 컴퓨터에서 코드는 매우 다양하게 사용된다.
- 예) ASCII 문자 코드에서 'A'는 65이다.
- 예) 월드와이드웹의 통신 프로토콜 HTTP에서 응답코드 200번은 OK를 의미, 404번은 NOT FOUND이다. 숫자 자체에 의미가 있는 게 아니라 통신을 편하게 하기 위해서 숫자값에 의미를 부여한 것이다.
- 문자열 처리가 번거롭고 실수 발생 가능하고 성능이나 메모리에 더 안 좋은 문제가 있다. 따라서 문자열 대신 숫자값에 의미를 부여하는 코드로 처리할 수 있다.

```
package main

import "fmt"

//코드 부여
const Pig int = 0
const Cow int = 1
const Chicken int = 2

func PrintAnimal(animal int) {
  if animal == Pig {
    fmt.Println("꿀꿀")
  } else if animal == Cow {
    fmt.Println("음메")
  } else if animal == Chicken {
    fmt.Println("꼬끼오")
  } else {
    fmt.Println("...")
  }
}

func main() {
  PrintAnimal(Pig)
  PrintAnimal(Cow)
  PrintAnimal(Chicken)
}
```

출력값

```
꿀꿀
음메
꼬끼오
```

8.2.3 iota로 간편하게 열거값 사용하기

- 코드값으로 사용하기 때문에 값이 그냥 1, 2, 3, ...처럼 1씩 증가하도록 정의할 때 iota 키워드를 사용하면 편리하다.
- iota는 0부터 1씩 증가하며 소괄호를 벗어나면 다시 초기화된다.
- 예) 빨강, 파랑, 초록에 숫자 코드 부여

```
const(
  Red   int = iota //0
  Blue  int = iota //1
  Green int = iota //2
)
```

- 만약 첫번째 값고 똑같은 규칙이 계속 적용된다면 타입과 iota를 생략할 수 있다

```
const (
  C1 uint = iota + 1 //1 = 0 + 1
  C2                 //2 = 1 + 1
  C3                 //3 = 2 + 1
)
```

```
const (
  BitFlag1 uint = 1 << iota //1 = 1 << 0
  BitFlag2                  //2 = 1 << 1
  BitFlag3                  //4 = 1 << 2
  BitFlag4                  //8 = 1 << 3
)

const (
  A int = iota //0
  B            //1
)
```

8.3 타입 없는 상수

상수 선언 시 타입을 명시 않을 수 있다. 타입 없는 상숫값은 타입이 정해지지 않은 상태로 사용된다.

```
package main

import "fmt"

const PI = 3.14               //타입 없는 상수
const FloatPI float64 = 3.14 //타입 상수

func main() {
  var a int = PI * 100      //오류 발생하지 않는다.
  var b int = FloatPI * 100 //타입 오류 발생

  fmt.Println(a)
  fmt.Println(b)
}
```

출력값

```
.\ex8.4.go:10:6: cannot use FloatPI * 100 (type float64) as type int in assignment
```

- PI 값은 타입이 없기 때문에 3.14 숫자로만 동작한다. → PI * 100 = 314는 정수 타입 변수에 대입 가능하다.
- FloatPI는 float64 타입이기 때문에 타입 오류가 발생한다.
- 타입 없는 상수는 변수에 복사될 때 타입이 정해지기 때문에 여러 타입에 사용되는 상수값을 사용할 때 편리하다.

8.4 상수와 리터럴

컴퓨터에서 리터럴이란 고정된 값, 값 자체로 쓰인 문구이다.

```
var str string = "Hello World"
var i int = 0
i = 30
```

"Hello World, 0, 30과 같이 고정된 값 자체로 쓰인 문구가 리터럴이다. Go 언어에서 상수는 리터럴과 같이 취급한다. 그래서 컴파일될 때 상수는 리터럴로 변환되어 실행 파일에 쓰인다.

상수 표현식 역시 컴파일 타임에 실제 결괏값 리터럴로 변환하기 때문에 상수 표현식 계산에 CPI 자원을 사용하지 않는다.

```
const PI = 3.14
var a int = PI * 100
```

위 구문은 컴파일 타임에 아래와 같이 변환된다.

```
var a int = 314
```

상수의 메모리 주솟값에 접근할 수 없는 이유 역시 컴파일 타임에 리터럴로 전환되어서 실행 파일에 값 형태로 쓰이기 때문이다. 그래서 동적 할당 메모리 영역을 사용하지 않는다.

연습문제

1.

```
const Gravity = 9.80665
```

2.

```
2 3
```

iota는 소괄호를 벗어나면 초기화된다.