Chapter 10 switch문

날짜	@November 10, 2021
발표자	장예서

10.1 switch문 동작 원리

switch문은 값에 따라 다른 로직을 수행할 때 사용.

```
switch 비굣값 {
case 값1:
  문장
case 값2:
  문장
default:
  문장
}
```

switch 키워드 다음 비굣값. 첫번째 case부터 값 검사. 만약 비굣값과 case값이 같으면 해당 case 문장 수행 후 switch문 종료. 같은 값이 없으면 default 문장 수행. default 생략 가능

```
package main

import "fmt"

func main() {
  a := 3

  switch a {
  case 1:
    fmt.Println("a == 1")
  case 2:
    fmt.Println("a == 2")
  case 3:
    fmt.Println("a == 3")
  case 4:
    fmt.Println("a == 4")
  default:
```

```
    fmt.Println("a > 4")
  }
}
```

출력문

```
a == 3
```

10.2 switch문을 언제 쓰는가?

switch문을 이용해 복잡한 if else문을 보기 좋기 정리

```
package main

import "fmt"

func main() {

  day := 3

  if day == 1 {
    fmt.Println("첫째 날입니다")
    fmt.Println("오늘은 팀미팅이 있습니다.")
  } else if day == 2 {
    fmt.Println("둘째 날입니다")
    fmt.Println("새로운 팀원 면접이 있습니다.")
  } else if day == 3 {
    fmt.Println("셋째 날입니다.")
    fmt.Println("설계안을 확정하는 날입니다.")
  } else if day == 4 {
    fmt.Println("넷째 날입니다.")
    fmt.Println("예산을 확정하는 날입니다.")
  } else if day == 5 {
    fmt.Println("다섯째 날입니다.")
    fmt.Println("최종 계약하는 날입니다.")
  } else {
    fmt.Println("프로젝트를 진행하세요")
  }
}
```

출력문

```
셋째 날입니다.
설계안을 확정하는 날입니다.
```

switch문 활용

```
package main

import "fmt"

func main() {

  day := 3

  switch day {
  case 1:
    fmt.Println("첫째 날입니다")
    fmt.Println("오늘은 팀미팅이 있습니다.")
  case 2:
    fmt.Println("둘째 날입니다")
    fmt.Println("오늘은 면접이 있습니다.")
  case 3:
    fmt.Println("셋째 날입니다.")
    fmt.Println("설계안을 확정하는 날입니다.")
  case 4:
    fmt.Println("넷째 날입니다.")
    fmt.Println("예산을 확정하는 날입니다.")
  case 5:
    fmt.Println("다섯째 날입니다.")
    fmt.Println("최종 계약하는 날입니다.")
  default:
    fmt.Println("프로젝트를 진행하세요")
  }
}
```

출력문

```
셋째 날입니다.
설계안을 확정하는 날입니다.
```

10.3 다양한 switch문 형태

10.3.1 한 번에 여러 값 비교

- 하나의 case는 하나 이상의 값을 비교할 수 있음. (쉼표 , 로 구분)

```
package main

import "fmt"
```

```
func main() {

  day := "thursday"

  switch day {
  case "monday", "tuesday":
    fmt.Println("월, 화요일은 수업 가는 날입니다.")
  case "wednesday", "thursday", "friday":
    fmt.Println("수, 목, 금요일은 실습 가는 날입니다.")
  }
}
```

출력문

```
수, 목, 금요일은 실습 가는 날입니다.
```

10.3.2 조건문 비교

- switch문의 동작을 응용하면 단순히 값만 비교하는 게 아닌 if문처럼 true가 되는 조건문을 검사할 수 있음

```
package main

import "fmt"

func main() {
  temp := 18

  switch true {
  case temp < 10 || temp > 30:
    fmt.Println("바깥 활동하기 좋은 날씨가 아닙니다.")
  case temp >= 10 && temp < 20:
    fmt.Println("약간 추울 수 있으니 가벼운 겉옷을 준비하세요")
  // 이미 두 번째 case를 실행해서 검사X.
  case temp >= 15 && temp < 25:
    fmt.Println("야외활동 하기 좋은 날씨입니다")
  default:
    fmt.Println("따뜻합니다.")
  }
}
```

출력문

```
약간 추울 수 있으니 가벼운 겉옷을 준비하세요
```

- 비굣값을 true로 할 경우 case의 조건문이 true가 되는 경우가 실행됨
- 이미 case 로직이 실행되면 switch문 종료
- switch문 다음에 비굣값을 적지 않은 경우 default값으로 true 사용

10.3.3 switch 초기문

- if문과 마찬가지로 초기문 사용 가능

```
switch 초기문; 비굣값 {
case 값1:
  ...
case 값2:
  ...
default:
}
```

```
package main

import "fmt"

func getMyAge() int {
  return 22
}

func main() {
  switch age := getMyAge(); age { // getMyAge() 결과값 비교
  case 10:
    fmt.Println("Teenage")
  case 33:
    fmt.Println("Pair 3")
  default:
    fmt.Println("My age is", age) // age값 사용
  }

  fmt.Println("age is", age) // error - age 변수는 사라짐
}
```

출력문

```
.\ex10.6.go:19:24: undefined: age
```

- 초기문으로 getMyAge() 함수 실행 후 반환값으로 age값 초기화

- age는 switch문이 종료되기 전까지 접근 가능
- switch문 종료되면 age도 사라지기 때문에 에러 발생

```
package main

import "fmt"

func getMyAge() int {
  return 22
}

func main() {
  // age 변수 선언 및 초기화
  switch age := getMyAge(); true {
  case age < 10:
    fmt.Println("Child")
  case age < 20:
    fmt.Println("Teenager")
  case age < 30:
    fmt.Println("20s")
  default:
    fmt.Println("My age is", age) // age값 사용
  }
}
```

출력문

```
20s
```

- age 변수를 초기문을 통해서 선언 및 대입 후 switch의 비굣값으로 true 사용. true일 경우 생략 가능하므로 다음과 같이 작성 가능

```
switch age := getMyAge(); {
```

10.4 const 열거값과 switch

const 열거값에 따라 수행되는 로직을 변경할 때 switch문을 주로 사용.

```
package main

import "fmt"
```

```
type ColorType int // 별칭 ColorType 선언 & Const 열거값 정의
const (
  Red ColorType = iota
  Blue
  Green
  Yellow
)

// 각 ColorType 열거값에 따른 문자열을 반환하는 함수
func colorToString(color ColorType) string {
  switch color {
  case Red:
    return "Red"
  case Blue:
    return "Blue"
  case Green:
    return "Green"
  case Yellow:
    return "Yellow"
  default:
    return "Undefined"
  }
}

func getMyFavoriteColor() ColorType {
  return Blue
}

func main() {
  fmt.Println("My favorite color is", colorToString(getMyFavoriteColor()))
}
```

출력문

```
My favorite color is Blue
```

- ColorType 별칭 타입을 정의하고 열거값을 정의. switch문을 이용해 각 열거값에 따른 문자열을 반환하는 함수.
- 위 예에서 새로운 색깔 추가하면 colorToString() 함수도 수정해야함. 이런 경우 커플링되었다, 결함되어있다고 함. 열거값이 수정될 때 연관된 모든 switch case문도 수정해야함. 그래서 열거값에 연관된 switch case가 많아질수록 작은 수정에도 많은 코드가 변경되어야하는 **산탄총 수술 문제**가 발생. **하나의 열거값에 연관된 switch case는 최대한 줄이는 게 좋음.**

10.5 break와 fallthrough 키워드

일반적으로 다른 언어에서는 switch문의 각 case 종료 시에 break문을 사용해야 다음 case로 코드가 이어서 실행되지 않음. 하지만 Go 언어에서는 break를 사용하지 않아도 case 하나를 실행 후 자동으로 switch문을 빠져나감.

```
package main

import "fmt"

func main() {

  a := 3

  switch a {
  case 1:
    fmt.Println("a == 1")
    break
  case 2:
    fmt.Println("a == 2")
    break
  case 3:
    fmt.Println("a == 3")
  case 4:
    fmt.Println("a == 4")
  default:
    fmt.Println("a > 4")
  }
}
```

출력문

```
a == 3
```

- break를 사용하든 않든 상관없이 case 하나를 실행 후 switch문을 빠져나감.

만약 하나의 case문 실행 후 다음 case문까지 같이 실행하고 싶을 땐 → fallthrough 키워드 사용

```
package main

import "fmt"

func main() {

  a := 3
```

```
  switch a {
  case 1:
    fmt.Println("a == 1")
    break
  case 2:
    fmt.Println("a == 2")
  case 3:
    fmt.Println("a == 3")
    fallthrough
  case 4:
    fmt.Println("a == 4")
  case 5:
    fmt.Println("a == 5")
  default:
    fmt.Println("a > 4")
  }
}
```

출력문

```
a == 3
a == 4
```

연습문제

1.

```
package main

import "fmt"

func main(){
  day := 1

  switch day {
  case 1:
    fmt.Println("First day")
  case 2:
    fmt.Println("Second day")
  default:
    fmt.Println("Another day")
  }
}
```

2.

```
긍정적인 평가
```

3.

```
package main

import "fmt"

type Direction int

const (
  None Direction = iota
  North
  East
  South
  West
)

func DirectionToString(d Direction) string {
  switch d {
  case North:
    return "North"
  case East:
    return "East"
  case South:
    return "South"
  case West:
    return "West"
  default:
    return "None"
  }
}

func GetDirection(angle float64) Direction {
  switch {
  case angle >= 315, angle >= 0 && angle < 45:
    return North
  case angle >= 45 && angle < 135:
    return East
  case angle >= 135 && angle < 225:
    return South
  case angle >= 225 && angle < 315:
    return West
  default:
    return None
  }
}
```

```
func main() {
  fmt.Println(DirectionToString((GetDirection(38.3))))

}
```