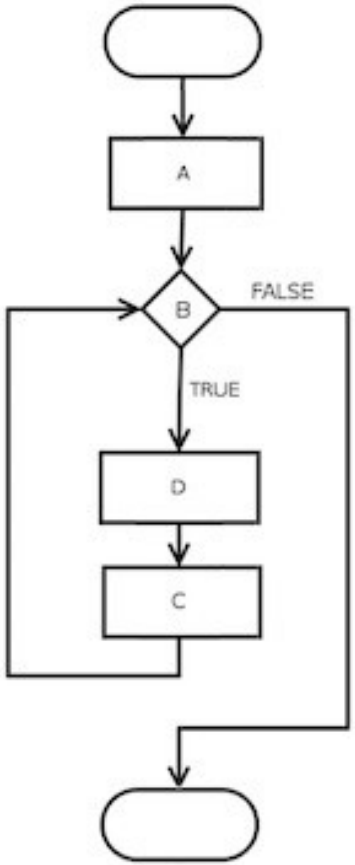
Kotlin – Control Statements

Tushar B. Kute,
http://tusharkute.com

Control Flow



Control Flow Statements

- if...else Expression
- ehen Expression
- for Loop
- while Loop
- break and continue

if-else Statements

- The syntax of a traditional if...else expression is as follows:

```
if (condition) {
   // code block A to be executed if condition is true
} else {
  // code block B to be executed if condition is false
}
```

if-else Statements

- if...else can also be used as an expression which returns a value and this value can be assigned to a variable.

```
val result = if (condition) {
    // code block A to be executed if condition is true
} else {
  // code block B to be executed if condition is false
}
```

if-else Statements

- You can ommit the curly braces { } when if has only one statement.
- You can include multiple statements in if...else block, in this case the last expression is returned as the value of the block.

if...else...if Ladder

```
if (condition1) {
  // code block A to be executed if condition1 is true
} else if (condition2) {
  // code block B to be executed if condition2 is true
} else {
  // code block C to be executed if condition1 and
condition2 are false
}
```

Nested if Expression

```
if(condition1) {
   // code block A to be executed if condition1 is true
   if( (condition2) {
      // code block B to be executed if condition2 is true
   } else{
      // code block C to be executed if condition2 is fals
   }
} else {
  // code block D to be executed if condition1 is false
}
```

when Statement

- Consider a situation when you have large number of conditions to check.
- Though you can use if..else if expression to handle the situation, but Kotlin provides when expression to handle the situation in nicer way.
- Using when expression is far easy and more clean in comparison to writing many if...else if expressions.
- Kotlin when expression evaluates a section of code among many alternatives as explained in example.

when Statement

- Kotlin when can be used either as an expression or as a statement, simply like a switch statement in Java.
- If it is used as an expression, the value of the first matching branch becomes the value of the overall expression.

when Statement

- We can combine multiple when conditions into a single condition.
- Kotlin ranges are created using double dots .. and we can use them while checking when condition with the help of in operator.
- Kotlin when can use arbitrary expressions instead of a constant as branch condition.
- Kotlin when branches can be put as block of code enclosed within curly braces.

Thank you

@mitu_skillologies

@mITuSkillologies

@mitu_group

@mitu-skillologies

@MITUSkillologies

kaggle

@mituskillologies

Web Resources
https://mitu.co.in
http://tusharkute.com

@mituskillologies

contact@mitu.co.in

tushar@tusharkute.com