SWIFT
Operator

Bill Kim(김정훈) | ibillkim@gmail.com

목차

Basic Operators

Swift에서도 다른 언어들처럼 다양한 타입의 연산자를 제공합니다. 지원하는 연산자는 아래와 같습니다.

1) 할당 연산자
2) 산술 연산자
3) 비교 연산자
4) 논리 연산자
5) 범위 연산자
6) 삼항 조건 연산자
7) Nil 병합 연산자

Basic Operators

또한 연산자는 대상에 수에 따라서 아래와 같이 분류할 수 있습니다.

1) 단항 연산자
    -a, !b, c!와 같이 대상 앞뒤에 바로 붙여서 사용하는 연산자

2) 이항 연산자
    2 + 3 과 같이 두 대상 사이에 위치하는 연산자

3) 삼항 연산자
    a ? b : c 형태로 참과 거짓에 대한 조건을 나타내는 연산자
    Swift에서는 제공하는 삼항 연산자는 위 형태 하나만 존재

Assignment Operator

할당 연산자는 값을 초기화하거나 변경할 때 사용합니다.
아래와 같이 상수, 변수 모두 사용 가능합니다.

```swift
let b = 10
var a = 5

a = b // a 값은 10
```

```swift
let (x, y) = (1, 2) // x 는 1, y 값은 2 가 됩니다.

print("x : \(x), y : \(y)") // x : 1, y : 2

if x = y {
    //  x = y 는 값을 반환하지 않기 때문에 이 문법은 올바르지 않습니다.
}
```

Arithmetic Operators

사칙 연산자는 덧셈(+), 뺄셈(-), 곱셉(*), 나눗셈(/) 등에서 사용하는 연산자입니다.

```
1 + 2 // 3
5 - 3 // 2
2 * 3 // 6
10.0 / 2.5 // 4.0

// 아래와 같이 나머지 연산자를 사용할 수 있습니다.
9 % 4 // 1
-9 % 4 // -1

// 단항 음수 연산자(Unary Minus Operator)
let three = 3
let minusThree = -three  // minusThree는 -3
let plusThree = -minusThree // plusThree는 3, 혹은 "minus minus 3"

// 단항 양수 연산자(Unary Plus Operator)
let minusSix = -6
let alsoMinusSix = +minusSix // alsoMinusSix는 -6

// 합성 할당 연산자 (Compound Assignment Operators)
var a = 1
a += 2 // a는 3

// 덧셈 연산을 통하여 문자를 합칠 수 있습니다.
"hello, " + "world"  // equals "hello, world"
```

Comparison Operators

Swift에서는 비교를 위한 **비교 연산자**로서 아래와 같은 연산자를 지원합니다.

1) 같다 (a == b)
2) 같지않다 (a != b)
3) 크다 (a > b)
4) 작다 (a < b)
5) 크거나 같다 (a >= b)
6) 작거나 같다 (a <= b)

```
1 == 1   // true
2 != 1   // true
2 > 1    // true
1 < 2    // true
1 >= 1   // true
2 <= 1   // false
```

Comparison Operators

```swift
let name = "world"

// if-else 조건 구문에서도 비교 연산을 할 수 있다.
if name == "world" {
    print("hello, world")
} else {
    print("I'm sorry \(name), but I don't recognize you")
}
// Prints "hello, world", because name is indeed equal to "world".
```

```swift
(1, "zebra") < (2, "apple") // true, 1이 2보다 작고; zebra가 apple은 비교하지 않기 때문
(3, "apple") < (3, "bird") // true 왼쪽 3이 오른쪽 3과 같고; apple은 bird보다 작기 때문
(4, "dog") == (4, "dog") // true 왼쪽 4는 오른쪽 4와 같고 왼쪽 dog는 오른쪽 dog와 같기 때문

("blue", -1) < ("purple", 1) // 비교가능, 결과 : true
("blue", false) < ("purple", true) // 에러, Boolean 값은 < 연산자로 비교할 수 없다.
```

Logical Operators

Swift에서는 세가지 표준 논리 연산자를 지원합니다.

1) 논리 부정 NOT (!a)
2) 논리 곱 AND (a && b)
3) 논리 합 (a || b)

Logical Operators

```swift
// 논리 부정 연산자(Logical NOT Operator)
let allowedEntry = false
if !allowedEntry {
    print("ACCESS DENIED")
} // Prints "ACCESS DENIED"

// 논리 곱 연산자(Logical AND Operator)
let enteredDoorCode = true
let passedRetinaScan = false
if enteredDoorCode && passedRetinaScan {
    print("Welcome!")
} else {
    print("ACCESS DENIED")
} // Prints "ACCESS DENIED"

// 논리 합(OR) 연산자(Logical OR Operator)
let hasDoorKey = false
let knowsOverridePassword = true
if hasDoorKey || knowsOverridePassword {
    print("Welcome!")
} else {
    print("ACCESS DENIED")
} // Prints "Welcome!"
```

Logical Operators

```swift
let enteredDoorCode = true
let passedRetinaScan = false

let hasDoorKey = false
let knowsOverridePassword = true

// 논리 연산자의 조합(Combining Logical Operators)
if enteredDoorCode && passedRetinaScan || hasDoorKey || knowsOverridePassword {
    print("Welcome!")
} else {
    print("ACCESS DENIED")
} // Prints "Welcome!"

// 명시적 괄호(Explicit Parentheses)
if (enteredDoorCode && passedRetinaScan) || hasDoorKey || knowsOverridePassword {
    print("Welcome!")
} else {
    print("ACCESS DENIED")
} // Prints "Welcome!"
```

Range Operators

Swift에서 지원하는 특별한 연산자로서 특정 범위에 대해서 지정할 수 있는 **범위 연산자**입니다.

```
// 닫힌 범위 연산자(Closed Range Operator)
// (a..b)의 형태로 범위의 시작과 끝이 있는 연산자 입니다. for-in loop에 자주 사용됩니다.

for index in 1...5 {
    print("\(index) times 5 is \(index * 5)”)
}
// 1 times 5 is 5
// 2 times 5 is 10
// 3 times 5 is 15
// 4 times 5 is 20
// 5 times 5 is 25

// 반 닫힌 범위 연산자(Half-Open Range Operator)
// (a..<b)의 형태로 a부터 b보다 작을 때까지의 범위를 갖습니다. 즉, a부터 b-1까지 값을 갖습니다.

let names = ["Anna", "Alex", "Brian", "Jack"]
let count = names.count
for i in 0..<count {
    print("Person \(i + 1) is called \(names[i])")
}
// Person 1 is called Anna
// Person 2 is called Alex
// Person 3 is called Brian
// Person 4 is called Jack
```

Range Operators

```swift
// 단방향 범위(One-Side Ranges)
// [a..] [..a]의 형태로 범위의 시작 혹은 끝만 지정해 사용하는 범위 연산자 입니다.

let names = ["Anna", "Alex", "Brian", "Jack"]

for name in names[2...] {
    print(name)
}
// Brian
// Jack

for name in names[...2] {
    print(name)
}
// Anna
// Alex
// Brian

for name in names[..<2] {
    print(name)
}
// Anna
// Alex

// 단방향 범위 연산자는 subscript뿐만 아니라 아래와 같이 특정 값을 포함하는지 여부를 확인할 때도 사용 가능합니다.
let range = ...5

print(range.contains(7))   // false
print(range.contains(4))   // true
print(range.contains(-1))  // true
```

Other Operators

삼항 조건 연산자(Ternary Conditional Operator)

```
// 삼항 조건 연산자는 question ? answer1 : answer2의 구조를 갖습니다.
// question 조건이 참인경우 answer1이 거짓인 경우 answer2가 실행됩니다.

let contentHeight = 40
let hasHeader = true
let rowHeight = contentHeight + (hasHeader ? 50 : 20)

print(rowHeight) // rowHeight는 90 (40 + 50)

// 위에서 삼항 조건 연산을 아래과 같이 풀어 쓸 수 있습니다.
let contentHeight = 40
let hasHeader = true
let rowHeight: Int
if hasHeader {
    rowHeight = contentHeight + 50
} else {
    rowHeight = contentHeight + 20
} // rowHeight는 90입니다.
```

Other Operators

Nil 병합 연산자(Nil-Coalescing Operator)

```
// nil 병합 연산자는 a ?? b 형태를 갖는 연산자 입니다.
// 옵셔널 a를 벗겨서(unwraps) 만약 a가 nil 인 경우 b를 반환합니다.

// 사용 예) a != nil ? a! : b

let defaultColorName = "red"
var userDefinedColorName: String?   // 이 값은 defaults 값 nil입니다.

var colorNameToUse = userDefinedColorName ?? defaultColorName
print(colorNameToUse) // red
// userDefinedColorNam이 nil이므로 colorNameToUse 값은 defaultColorName인 "red"가 설정
됩니다.

userDefinedColorName = "green"
colorNameToUse = userDefinedColorName ?? defaultColorName
print(colorNameToUse) // green
// userDefinedColorName가 nil이 아니므로 colorNameToUse 는 "green"이 됩니다.
```

References

[1] 기본 연산자 (Basic Operators) : https://jusung.gitbook.io/the-swift-language-guide/language-guide/02-basic-operators

[2] [Swift]Basic Operators 정리 : http://minsone.github.io/mac/ios/swift-basic-operators-summary

[3] Swift 5.2: Basic Operators (기본 연산자) : https://xho95.github.io/swift/language/grammar/basic/operators/2016/04/27/Basic-Operators.html

[4] Basic Operators : https://docs.swift.org/swift-book/LanguageGuide/BasicOperators.html

[5] 오늘의 Swift 상식 (Custom Operator, Generic) : https://medium.com/@jgj455/오늘의-swift-상식-custom-operator-generic-58e783742b

References

[6] Swift Operators : https://www.programiz.com/swift-programming/operators

[7] 범위 연산자 (Range Operators) : https://m.blog.naver.com/PostView.nhn?blogId=badwin&logNo=221177227872&proxyReferer=https%3A%2F%2Fwww.google.com%2F

[8] Swift - Operators : https://www.tutorialspoint.com/swift/swift_operators.htm

[9] Operator Declarations : https://developer.apple.com/documentation/swift/swift_standard_library/operator_declarations

[10] (Swift) 스위프트 연산자(Operator) 기초 : https://googry.tistory.com/37

Thank you!