Swift in Hinde :-

Operators एक Symbol है जो कम्पाइलर को विशिष्ट Mathematical या Logical Manipulations करने के लिए कहता है.

|  |  |
| --- | --- |
| **Operators** | **Description** |
| + | Addition |
| – | Subtraction |
| \* | Multiplication |
| / | Division |
| % | Modulus |
| ++ | Increment |
| −− | Decrement |

xAssignment Operator Variables को Value प्रदान करने के लिए उपयोग किया जाता है.

|  |  |
| --- | --- |
| **Operators** | **Description** |
| = | Assign |
| += | Increments, then assigns |
| -+ | Decrements, then assigns |
| \*= | Multiplies, then assigns |
| /= | Divides, then assigns |
| %= | Modulus, then assigns |

## xLogical Operators

Swift भाषा Logical Operators को Support करती है.

|  |  |
| --- | --- |
| **Operators** | **Description** |
| && | *And* और Operators दो Expressions पर एक Logical Conjunction का निष्पादन करते है (यदि दोनों Expressions True का Evaluation करते हैं, तो परिणाम True है .अगर किसी भी अभिव्यक्ति का Evaluation False है, तो परिणाम False है). |
| || | *Or* Operators दो Expressions पर एक Logical Disjunction का निष्पादन करता है (यदि दोनों या Expressions दोनों True का Evaluation करते हैं, तो परिणाम True है). |
| ! | *Not* Operator एक Expression पर Logical Negation करता है. |

Swift में एक Character String Literal है जिसे Data Type Character द्वारा संबोधित किया गया है.

**let** char1: Character = "X" **let** char2: Character = "Y" println("Value of char1 \(char1)") println("Value of char2 \(char2)")

Loops एक विशिष्ट Block Codes के खंड को दोहराने के लिए एक कार्यक्रम में जाना जाता है. इससे एक बार फिर से लिखने का कार्य कम हो जाता है.

**For-in Loops**

आप अनुक्रम या चीज़ों के संग्रह पर पुनरावृति करने के लिए For-in Loops का उपयोग करते हैं .जैसे कि सरणी या शब्दकोश के तत्व String में Characters या Numbers की Category यहाँ सामान्य रूप है.

**Syntax**

**for** index **in** sequence {

statements

}

उपर्युक्त उदाहरण में, यह कई संख्याओं से अधिक चलती है Loop Index Variable (i) प्रत्येक बार Loop के माध्यम से Category में अगली संख्या के मान लेता है.

**import** Cocoa

**var** someInts:[Int] = [10, 20, 30]

**for** index **in** someInts {

println( "Value of index is \(index)")

**import** Cocoa **var** someInts:[Int] = [10, 20, 30] **for** **var** index = 0; index < 3; ++index { println( "Value of someInts[\(index)] is \(someInts[index])")

## While Loop

Swift Programming भाषा में Loop Statement में एक बार जब तक दी गई Condition True है तब तक एक Target Statement का Execution किया जाता है.

## Syntax

**while** condition {

statement(s)

}

Programmers Loop का उपयोग कर सकते हैं जो कि For-condition-increment Loop की कार्यक्षमता को दोहराने के लिए निम्नानुसार हो.

import Cocoa

var **index** = 10

**while** **index** < 25 {

println( "Value of index is \(index)")

**index** = **index** + 1

}

## do...while Loop

Loop के Opposite और Loop के Opposite जो Loop के शीर्ष पर Loop की स्थिति का परीक्षण करते हैं. जबकि Loop पाश के निचले भाग में स्थित इसकी स्थिति जांचता है.

एक Do...While Loop एक While Loop के समान है, सिवाय इसके कि Do...While Loop जबकि Loop को कम से कम एक बार गारंटी के साथ निष्पादित नहीं कर देता है.

## Syntax

**do** {

statement(s);

}**while**( condition );

## For Example

import Cocoa

var **index** = 10

**do** {

println( "Value of index is \(index)")

**index** = **index** + 1

}**while** **index** < 20

Swift Arrays को किसी विशेष प्रकार के Sequential सूचियों में कई मानों को संग्रहीत करने के लिए लागू किया जाता है.

Swift ने सख्त जांच तंत्र की स्थापना की, जो Programmer को गलती से किसी भी गलत प्रकार में प्रवेश करने या सम्मिलित करने की अनुमति नहीं देता है. अर्थात Swift Programming में Arrays को वे प्रकार के मूल्यों के बारे में विशिष्ट हैं जो वे स्टोर कर सकते हैं.

एक ही Value कई स्थानों पर कई बार Arrays में प्रकट हो सकता है

Swift Arrays के types सुरक्षित होते हैं.

Arrays को Arrays के अंत में नई वस्तुओं को सम्मिलित करने के लिए Append () विधि या अतिरिक्त Assignment Operator (+ =) का उपयोग करके आसानी से संशोधित किया जा सकता है.

var arr = [Int]()

arr.append(100)

arr.append(200)

arr += [300]

var arr2 = arr[0]

print( "First element of array is \(arr2)" )

print( "Second element of array is \(arr[1])" )

print( "Third element of array is \(arr[2])" )

## Iterating Over an Array

आप Array के संपूर्ण सेट पर Iterate करने के लिए For in Loop का उपयोग कर सकते हैं.

import Cocoa

var someStrs = [String]()

someStrs.append("Facebook")

someStrs.append("YouTube")

someStrs += ["Google"]

for item in someStrs {

println(item)

}

**The Empty Property**

Programmers सामान्यत केवल पढ़ने के लिए खाली Array की Property का पता लगाने के लिए उपयोग करते हैं कि क्या कोई Array खाली है या नहीं.

.isempty

Decision Making वाली संरचनाओं के लिए आवश्यक है कि Programmer एक या अधिक शर्तों को निर्दिष्ट करता है कि Programmer द्वारा Evaluation या परीक्षण किया जा सकता है साथ ही एक Statement या Description के साथ निष्पादित किया जाता है. यदि स्थिति True होने के लिए निर्धारित होती है और वैकल्पिक रूप से, अन्य Statement का निष्पादन Condition False होना निर्धारित है.

Swift अपने Programming के अंतर्गत निम्नलिखित प्रकार के Decision Making Statements प्रदान करता है.

* If Statement
* If….else Statement
* Else if Statement
* Nested if Statement
* Switch Statement

## If Statement

सभी Programming भाषा में सबसे पहले और सबसे आम है अगर यह Users के अनुमति के माध्यम से कार्यक्रम के प्रवाह को नियंत्रित करता है. यह Conditional Statement केवल तभी काम करता है जब Condition True हो.

## if…else Statement

अगर यहां एक वैकल्पिक और Description के साथ अगर Statement का पालन किया जाता है और यह Description केवल Boolean Expression (i.e. the condition) false हो जाती है तो उसे केवल Execute किया जाता है.

## else if Statement

एक Statement अगर एक वैकल्पिक और else if…else Statement, जो एक से अधिक Conditions का परीक्षण करने के लिए बहुत ही उपयोगी है और इसे पीछा किया जा सकता है अगर if…else if Statement है.

## Nested if Statement

यह हमेशा Swift में Legal होता है यदि Nest if-else Statement जिसका मतलब है कि Programmers एक if or else if Statement कर सकते हैं यदि Statement अगर if or else if Statement के अंदर हो.

{ ka anter hota h isme sara statement ek hi bracket k ander hota h }