Урок 2

Базовые операторы

Операторы - это специальные символы, при помощи которых можно производить, как арифметические действия с величинами, так и операции сравнения, изменения или проверки. Язык Swift поддерживает большинство стандартных операторов C, а также ряд возможностей для устранения типичных ошибок в коде.

Виды основных операторов:

- Арифметические операторы
- Операторы присваивания
- Операторы сравнения
- Операторы диапазона
- Логические операторы

Арифметические операторы

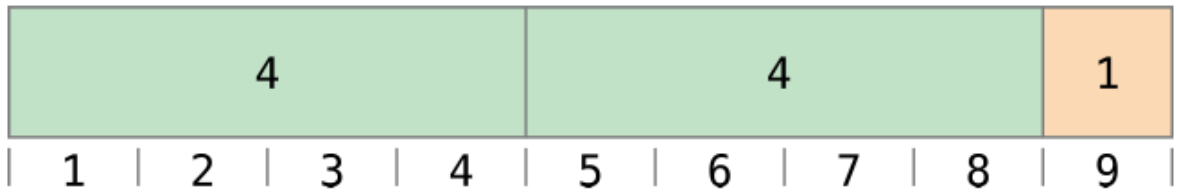
Swift поддерживает четыре стандартных арифметических оператора для всех числовых типов:

- (+) сложение
- (–) вычитание
- (*) умножение
- (/) деление

Кроме этих стандартных операторов существует еще один интересный оператор, который не встречается в математике. Это оператор остатка от деления или оператор целочисленного деления (**%**). Т.е. выражение (**a % b**) показывает остаток от деления **a** на **b**.

Пример:

Для вычисления выражения **9 % 4** сначала определяется, сколько **четверок** содержится в **девятке**:



В одной **девятке** содержатся две **четверки**, а остатком будет **1** (выделено оранжевым цветом).

Чтобы получить остаток от деления деления **a** на **b**, оператор **%** вычисляет следующее выражение:

a = (b × множитель) + остаток

где **множитель** показывает, сколько целых раз **b** содержится в **a**.

Подставляя в это выражение **9** и **4**, получим:

9 = (4 × 2) + 1

В Swift оператор остатка от деления работает как с положительными, так и с отрицательными числами, поэтому остаток, когда **a** отрицательно рассчитывается точно так же:

Подставляя в наше выражение **-9** и **4**, получим:

-9 = (4 × -2) + -1. Соответственно в этом случае остаток будет равен **-1**.

Если **b** отрицательно, его знак отбрасывается. Это означает, что выражения **a % b** и **a % -b** всегда будут давать одинаковый результат.

Оператор присваивания

Оператор присваивания (**a = b**) инициализирует или изменяет значение переменной **a** на значение **b**:

В отличии от математики, где этот оператор говорит о равенстве величин или выражает результат каких либо операций над величинами, в Swift этот оператор именно присваивает значение одной переменной или константы другой переменной. Т.е. переменная **a** до выражения (**a = b**) может и не равняться **b**. В программировании, что бы получить результат, например от сложения двух величин (**a + b**) не нужно писать оператор (**=**) после выражения сложения. Компилятор и так вычислит результат. Но при этом мы можем переменной **c** присвоить значение суммы двух других переменных (**c = a + b**).

Операторы присваивания могут иметь следующие формы записи

(**a += b**) сокращено от (**a = a + b**)

(**a -= b**) сокращено от (**a = a - b**)

(**a *= b**) сокращено от (**a = a * b**)

(**a /= b**) сокращено от (**a = a / b**)

Операторы сравнения

Операторы сравнения, как это видно из названия, могут сравнивать между собой два значения. Ранее, говоря о типах переменных, мы познакомились с логическим типом `Bool`, который может хранить `true` или `false`. Так вот операторы сравнения возвращают эти значения. Т.е. Когда мы сравниваем два объекта между собой, мы можем получить либо истину, либо ложь:

- равно (**a == b**)
- не равно (**a != b**)
- больше (**a > b**)
- меньше (**a < b**)

- больше или равно (**a >= b**)
- меньше или равно (**a <= b**)

Обратите внимание на первый оператор равенства (**==**). Этот оператор можно соотнести с математическим оператором равенства (**=**), который говорит о том, что сравниваемые величины равны между собой.

Операторы сравнения в основном используют в условных выражениях, например в конструкции `if`.

Операторы диапазона

В Swift есть два оператора диапазона, которые задают диапазон значений от и до.

Оператор замкнутого диапазона

Оператор замкнутого диапазона или оператор закрытого диапазона (**a...b**) определяет диапазон значений, который идет от **a** до **b** включая сами значения **a** и **b**. При этом значение **a** не должно быть больше значения **b**. Этот оператор используется для последовательного перебора значений из заданного диапазона.

Оператор полузамкнутого диапазона

Оператор полузамкнутого или оператор полуоткрытого диапазона (**a..<b**) задает диапазон от **a** до **b**, исключая значение **b**. Такой диапазон называется полузамкнутым, потому что он включает первое значение, но исключает последнее. Так же, как и для оператора замкнутого диапазона, значение **a** не должно превышать **b**. Если значение **a** равно значению **b**, то итоговый диапазон будет пустым.

Односторонние диапазоны

Операторы замкнутого диапазона имеют себе альтернативу - диапазон, который продолжается насколько возможно, но только в одну сторону (**a...**). В этих случаях вы можете пропустить значение с одной стороны оператора диапазона. Этот тип диапазона называется односторонним, потому что оператор имеет значение только с одной стороны.

Логические операторы

Логические операторы используются, как правило, со значениями логического типа `Bool`. В этом случае результатом работы оператора является значение типа **Boolian**. Swift поддерживает три стандартных логических оператора:

- (`!`) Логическое НЕ
- (`&&`) Логическое И
- (`||`) Логическое ИЛИ

Оператор логического НЕ

Данный оператор является префиксным оператором. Это значит, что он ставится непосредственно перед значением, без пробела (`!a`). Его можно интерпретировать, как НЕ **a**.

Оператор логического И

Оператор логического И (`&&`) записывается между двух операндов и возвращает логическое значение **true**, если оба операнда имеют значение **true**; в противном случае он возвращает значение **false**.

Оператор логического ИЛИ

Оператор логического ИЛИ (||) записывается между двух операндов и возвращает логическое значение **true**, если один или оба операнда имеют значение **true**; в противном случае он возвращает значение **false**.

Комбинирование логических операторов

Логические операторы можно комбинировать, составляя более сложные выражения. Например:

```
Если enteredDoorCode && passedRetinaScan || hasDoorKey || knowsOverridePassword, то "Welcome", Иначе - "ACCESS DENIED"
```

В этом примере с помощью нескольких операторов `&&` и || составляется более длинное и сложное выражение. Однако операторы `&&` и || по-прежнему применяются только к двум величинам, поэтому все выражение можно разбить на три простых условия. Алгоритм работы будет следующим:

- если пользователь правильно ввел код дверного замка и прошел сканирование сетчатки или если он использовал действующую ключ-карту или если он ввел код экстренного доступа, то дверь открывается.

Использование круглых скобок

Иногда имеет смысл использовать дополнительные круглые скобки, чтобы сложное логическое выражение стало проще для восприятия. В примере с открытием двери можно заключить в круглые скобки первую часть составного выражения, что сделает его нагляднее:

```
Если (enteredDoorCode && passedRetinaScan) || hasDoorKey || knowsOverridePassword
```

Круглые скобки показывают, что первые две величины составляют одно из возможных значений всего логического выражения. Хотя результат составного выражения не изменится, такая запись сделает код понятнее. Читаемость кода всегда важнее краткости, поэтому желательно ставить круглые скобки везде, где они облегчают понимание.

Строки и символы

Строки в Swift представлены типом `String`. Все строки состоят из символов, которые в свою очередь определяются типом `Character`.

Значения строковых данных должны браться в кавычки. Если присвоить переменной значение **10** то получится переменная с типом `Int`. Если же число взять в кавычки **«10»**, то переменная уже будет иметь тип `String`. Каждая строка состоит из независимых от кодировки символов Unicode, и обеспечивает поддержку доступа к этим символам в различных Unicode представлениях.

Объявление строковой переменной (константы):

```
let name: String
var surname = String()
let email = ""
```

NEXT

Работа с символами

Тип **String** в Swift представляет собой коллекцию значений **Character** в указанном порядке. Вы можете получить доступ к отдельным символу в строке с помощью перебора этой строки в цикле. В этом случает компилятор будет

вычленять каждый символ из всей строки, пока не закончиться вся строка. Таким образом, например можно подсчитать количество символов в строке.

Кроме того, можно создать отдельную **Character** константу или переменную из односимвольного строкового литерала с помощью присвоения типа **Character**:

```
let exclamationMark: Character = "!"
```

Из отдельных символов можно собрать строку, путем передачи массива символов в инициализатор строки

Конкатенация строк

Конкатенация означает сложение нескольких строк в одну. Сложение происходит при помощи оператора сложения (**+**).

Так же для объединения строк можно использовать комбинированный оператор сложения (+=):

К строке можно прибавить и символ, но для этого уже используется специальный метод **append**.

При этом вы не сможете то же самое повторить по отношению к переменной с типом **Character**, т.к. этот тип может содержать в себе только один символ.

Интерполяция строк

Интерполяция строк - это способ объединить переменные и константы различных типов внутри строки.

Для того, что бы добавить любую переменную или константу в строку, нужно вставить обратный слэш, открыть круглую скобку, вписать имя нашей переменной, закрыть круглую скобку.