Параметры функции и возвращаемые значения

В языке Swift параметры функций и возвращаемые значения реализованы очень гибко. Разработчик может объявлять любые функции – от простейших, с одним безымянным параметром, до сложных, со множеством параметров и составными именами.

Функции без параметров

В некоторых случаях функции могут не иметь входных параметров. Вот пример функции без входных параметров, которая при вызове всегда возвращает одно и то же значение типа `String`:

```
func sayHelloWorld() -> String {
    return "hello, world"
}
print(sayHelloWorld())
// Выведет "hello, world"
```

Обратите внимание, что несмотря на отсутствие параметров, в объявлении функции все равно нужно ставить скобки после имени. При вызове после имени функции также указываются пустые скобки.

Функции с несколькими входными параметрами

У функции может быть несколько параметров, которые указываются через запятую в скобках.

Эта функция принимает два параметра: имя человека и булево значение, приветствовали ли его уже, и возвращает соответствующее приветствие для этого человека:

```
func greet(person: String, alreadyGreeted: Bool) -> String {
    if alreadyGreeted {
        return greetAgain(person: person)
    } else {
        return greet(person: person)
    }
}
print(greet(person: "Tim", alreadyGreeted: true))
// Выведет "Hello again, Tim!"
```

Вы вызываете функцию `greet(person:alreadyGreeted:)`, передавая значение типа `String` параметру с ярлыком `person` и булево значение с ярлыком `alreadyGreeted`, взятое в скобки через запятую. Обратите внимание, что эта функция отличается от функции `greet(person:)`, которую вы видели в предыдущем разделе. Хотя имена обеих функций начинаются с `greet`, функция `greet(person:alreadyGreeted:)` принимает два аргумента, а `greet(person:)` принимает только один.

Функции, не возвращающие значения

В некоторых случаях функции могут не иметь возвращаемого типа. Вот другая реализация функции `greet(person:)`, которая выводит свое собственное значение типа `String`, но не возвращает его:

```
func greet(person: String) {
    print("Hello, \(person)!")
}
greet(person: "Dave")
// Выведет "Hello, Dave!"
```

Так как у функции нет выходного значения, в ее объявлении отсутствует результирующая стрелка (->) и возвращаемый тип.

Заметка

Строго говоря, функция `greet(person:)` *все же возвращает значение, хотя оно нигде и не указано. Функции, для которых не задан возвращаемый тип, получают специальный тип* `Void`*. По сути, это просто пустой кортеж, т. е. кортеж с нулем элементов, который записывается как ().*

Выходное значение функции может быть игнорировано:

```
func printAndCount(string: String) -> Int {
    print(string)
    return string.count
}
func printWithoutCounting(string: String) {
    let _ = printAndCount(string: string)
}
printAndCount(string: "hello, world")
// Выведет "hello, world" и возвращает значение 12
printWithoutCounting(string: "hello, world")
// Выведет "hello, world", но не возвращает значения
```

Первая функция, `printAndCount(string:)` выводит строку, а затем возвращает подсчет символов в виде целого (`Int`). Вторая функция, `printWithoutCounting(string:)` вызывает первую, но игнорирует ее возвращаемое значение. При вызове второй функции первая функция по-прежнему печатает сообщение, но ее возвращаемое значение не используется.

Заметка

Хотя возвращаемые значения можно игнорировать, функция все же должна возвратить то, что задано в ее объявлении. Функция, для

которой указан возвращаемый тип, не может заканчиваться оператором, который ничего не возвращает, иначе произойдет ошибка во время компиляции.

Функции, возвращающие несколько значений

Вы можете использовать кортежный тип в качестве возвращаемого типа для функции для возврата нескольких значений в виде составного параметра.

В следующем примере объявлена функция `minMax(array:)`, которая ищет минимальный и максимальный элементы в массиве типа `Int`:

```
func minMax(array: [Int]) -> (min: Int, max: Int) {
    var currentMin = array[0]
    var currentMax = array[0]
    for value in array[1..<array.count] {
        if value < currentMin {
            currentMin = value
        } else if value > currentMax {
            currentMax = value
        }
    }
    return (currentMin, currentMax)
}
```

Функция `minMax(array:)` возвращает кортеж из двух значений типа `Int`. Этим значениям присвоены имена `min` и `max`, чтобы к ним можно было обращаться при запросе возвращаемого типа функции.

Тело функции `minMax(array:)` начинается с инициализации двух рабочих переменных `currentMin` и `currentMax` значением первого целого элемента в массиве. Затем функция последовательно проходит по всем остальным значениям в массиве и сравнивает их со значениями `currentMin` и `currentMax` со-

ответственно. И наконец, самое маленькое и самое большое значения возвращаются внутри кортежа типа `Int`.

Так как имена элементов кортежа указаны в возвращаемом типе функции, к ним можно обращаться через точку и считывать значения:

```
let bounds = minMax(array: [8, -6, 2, 109, 3, 71])
print("min is \(bounds.min) and max is \(bounds.max)")
// Выведет "min is -6 and max is 109"
```

Обратите внимание, что элементам кортежа не нужно давать название в момент возвращения кортежа из функции, так как их имена уже указаны как часть возвращаемого типа функции.