6 Опциональные типы данных

Опциональные типы данных — это потрясающее нововведение языка Swift, которое вы, вероятно, никогда не встречали в других языках программирования и которое значительно расширяет возможности работы с типами данных.

6.1. Опционалы

Опциональные типы данных, также называемые *опционалами*, — это особый тип данных, который говорит о том, что некоторая переменная или константа либо имеет значение определенного типа, либо вообще не имеет никакого значения.

ПРИМЕЧАНИЕ Важно не путать отсутствие какого-либо значения в опциональном типе данных с пустой строкой или нулем. Пустая строка — это обычный строковый литерал, то есть вполне конкретное значение переменной типа String, а нуль — вполне конкретное значение числового типа данных. В случае же отсутствия значения в опциональном типе данных имеет место полное отсутствие значения как такового.

Рассмотрим абстрактный пример. Представьте, что у вас есть бесконечная плоскость. На ней устанавливают точку с определенными координатами (x, y). В любой момент времени мы можем говорить об этой точке и получать информацию о ней. Теперь уберем данную точку с плоскости. Несмотря на это вы все еще можете говорить о данной точке, но получить информацию о ней нельзя, поскольку точка уже не существует на плоскости. В данном примере точка — это некоторый объект (переменная, константа и т. д.), а ее координаты — опциональный тип данных, они могут иметь определенное значение, а могут отсутствовать в принципе.

Теперь рассмотрим опционал на практике. Вспомните метод-инициализатор класса `Int`, обозначающийся как `Int(_:)`. Данный метод

предназначен для создания целочисленного значения или конвертации некоторого числового значения в целочисленное. Не каждый передаваемый литерал может быть преобразован в целочисленный тип данных: например, строку `"1945"` можно конвертировать и вернуть в виде целого числа, а вот строку `"Привет, Дракон!"` вернуть в виде числа не получится (листинг 6.1).

Листинг 6.1

```
// переменная с числовым строковым литералом
let possibleString = "1945"                                    "1945"
/* при попытке конвертации
 преобразуется в целочисленный тип */
let convertPossibleString = Int(possibleString)                1 945
// переменная со строковым литералом
let unpossibleString = "Привет, Дракон!"                       "Привет,
                                                               Дракон!"
/* при попытке конвертации
 не преобразуется в целочисленный тип */
let convertUnpossibleString = Int(unpossibleString)            nil
```

В результате своей работы функция `Int(_:)` возвращает опциональный тип данных, то есть такой тип данных, который в данном случае может либо содержать целое число (`Int`), либо не содержать совершенно ничего.

Опциональный тип данных обозначается с помощью постфикса в виде знака вопроса после имени основного типа данных (то есть типа данных, на котором основан опционал). Для примера из предыдущего листинга опциональный тип `Int` обозначается как `Int?`. Опционалы могут быть основаны на любом типе данных, включая `Bool`, `String`, `Float` и `Double`.

Для того чтобы сообщить Swift о том, что значение в некотором объекте отсутствует, используется ключевое слово `nil`, указываемое в качестве значения этого объекта. Если для переменной указан опциональный тип данных, то она в любой момент времени может принять либо значение основного типа данных, либо `nil`. Значение опционала-константы задается единожды (листинг 6.2).

Листинг 6.2

```
/* переменная с опциональным типом Int
 и с установленным значением */
var dragonAge: Int? = 230                          230
// уничтожаем значение переменной
dragonAge = nil                                    nil
```

Переменная `dragonAge` является переменной опционального типа данных. Изначально ей присваивается значение, соответствующее основному для опционала типу данных (типу `Int` в данном случае). Так как `dragonAge` — это переменная, то мы можем изменить ее значение в любой момент. В результате мы присваиваем ей `nil`, после чего `dragonAge` не содержит никакого значения.

ПРИМЕЧАНИЕ В Swift ключевое слово nil можно использовать только с переменными опционального типа данных. При этом, если вы объявите такую переменную, но не инициализируете ее значение, Swift по умолчанию считает ее равной nil.

Для того чтобы объявить параметр опционального типа данных, можно использовать функцию `Optional(_:)`, как показано в листинге 6.3.

Листинг 6.3

```
// опциональная переменная с установленным значением
var optionalVar = Optional ("stringValue")            "stringValue"
optionalVar                                           "stringValue"
// уничтожаем значение опциональной переменной
optionalVar = nil                                     nil
optionalVar                                           nil
```

Так как функции `Optional(_:)` в качестве входного аргумента передано значение типа `String`, то возвращаемое ею значение имеет опциональный строковый тип данных.

В качестве значения данной функции необходимо передавать значение того типа данных, который должен стать основным для создаваемого опционала.

6.2. Извлечение опционального значения

Для того чтобы со значениями, содержащимися в опционалах, можно было работать, их необходимо специальным образом извлекать.

Принудительное извлечение значения

Запомните, что опциональный тип данных — это совершенно новый тип данных: `Int?` и `Int`, `String?` и `String`, `Bool?` и `Bool` — все это разные типы данных. Поэтому несмотря на то, что опционалы могут принимать значения основных типов данных, остальные свойства

этих типов к опционалам не относятся. Например, тип `Int?` не может использоваться в качестве операнда при выполнении арифметических операций (листинг 6.4).

Листинг 6.4

```
/* опциональная переменная
 с установленным значением */
var trollAge: Int? = 95                          95
trollAge = trollAge + 10 // ОШИБКА
```

Swift предлагает механизм решения данной проблемы, который называется *принудительным извлечением опционального значения* (forced unwrapping). При этом с помощью специального оператора значение опционального типа данных преобразуется в значение основного (для этого опционала) типа данных, например `Int?` преобразуется в `Int`. Для принудительного извлечения используется знак восклицания в качестве постфикса названия параметра (переменной или константы), содержащего значение опционального типа.

Исправим код, приведенный в предыдущем примере, для корректного подсчета суммы целых чисел (листинг 6.5).

Листинг 6.5

```
/* опциональная переменная
 с установленным значением */
var trollAge: Int? = 95                          95
// проведение арифметической операции
trollAge = trollAge! + 10                        105
trollAge                                         105
```

Для того чтобы преобразовать опциональное значение переменной `trollAge` в значение типа `Int`, к имени переменной добавим оператор принудительного извлечения опционального значения (!). В итоге выражение `trollAge! + 10` будет складывать два однотипных числа, и результат можно записать в качестве значения опционального типа `Int` в переменную `trollAge`.

При принудительном извлечении значения вы должны гарантировать, что параметр с опциональным типом данных содержит какое-либо значение, а не равен `nil`. В противном случае будет иметь место попытка преобразовать в основой тип данных несуществующее значение, поэтому Xcode сообщит об ошибке.

Косвенное извлечение значения

В противовес принудительному извлечению опционального значения Swift поддерживает *косвенное извлечение опционального значения* (implicitly unwrapping).

Если при инициализации значения опционала вы уверены, что данный параметр гарантированно будет иметь значение и никогда не будет равен `nil`, имеет смысл отказаться от принудительного извлечения значения с помощью знака восклицания всякий раз, когда это значение требуется. Для этой цели используется косвенное извлечение опционального значения.

При косвенном извлечении в качестве постфикса к типу данных (при указании типа данных) необходимо указывать не знак вопроса, а знак восклицания (например, `Int!` вместо `Int?`). В листинге 6.6 показана разница в использовании принудительного и косвенного извлечения опционального значения.

Листинг 6.6

```
var type: String
// принудительное извлечение опционального значения
let monsterOneType: String? = "Дракон"                    "Дракон"
type = monsterOneType!                                    "Дракон"
type                                                      "Дракон"
// косвенное извлечение опционального значения
let monsterTwoType: String! = "Тролль"                    "Тролль"
type = monsterTwoType                                     "Тролль"
type                                                      "Тролль"
```

При попытке косвенно извлечь несуществующее (то есть равное `nil`) опциональное значение Xcode сообщит об ошибке (листинг 6.7).

Листинг 6.7

```
let pointCoordinates: (Int, Int)! = nil
coordinates = pointCoordinates // ОШИБКА
```

В качестве основного типа для опционала можно использовать любой тип данных. Так как кортеж представляет собой отдельный тип данных, соответствующий типу входящих в него элементов, то и его тип можно брать за основу опционала, что и продемонстрировано в предыдущем примере.