


Спринт 1, тема «Перечисления»

Яндекс Практикум

Мы сделали для вас шпаргалку по теме «Перечисления». Здесь вы найдёте короткое изложение пройденного в уроке материала и ключевые фрагменты кода. Используйте шпаргалку, чтобы быстро восстановить в памяти пройденный материал.

 Скачайте документ, чтобы обращаться к нему при необходимости, пока не доведёте навыки до автоматизма.

Определение перечисления

Чтобы определить перечисление, используют синтаксис:

```
enum <ИмяПеречисления> {  
    case <кейс 1>  
    case <кейс 2>  
    ...  
}
```

Например:

```
enum FlyingSpecie {  
    case bird  
    case insect  
}
```

В коде выше:

1. `enum` — ключевое слово, с которого начинают объявление перечисления.
2. `FlyingSpecie` — название перечисления.

3. Внутри фигурных скобок перечисляют все значения перечисления, каждое с новой строки. Эти значения называют **кейсами** (от англ. "case" — случай).
4. Каждый новый кейс начинают с ключевого слова `case` и далее пишут название этого кейса. Названия кейсов принято писать с маленькой буквы.

Оператор Switch

Если нужно напечатать «Это птица» в случае, если передано значение `.bird`, и «Это насекомое», если передано `.insect`, то понадобится оператор `switch`:

```
func print(specie: FlyingSpecie) {
    switch specie {
        case .bird:
            print("Это птица")
        case .insect:
            print("Это насекомое")
    }
}
```

Оператор `switch` должен обрабатывать все возможные значения перечисления. Если не задан код для одного или нескольких значений, Swift выдаёт ошибку `Switch must be exhaustive`.

Оператор `default`

Если все кейсы, кроме явно перечисленных, нужно обработать одинаково, то используют ключевое слово `default`:

```
func print(specie: FlyingSpecie) {
    switch specie {
        case .bird:
            print("Это птица")
        default:
            print("Неизвестный вид")
    }
}
```

Объявление функции внутри перечисления

Функции можно объявлять внутри тела `enum`. В этом случае к перебираемому значению можно обращаться с использованием `self`:

```
enum FlyingSpecie {
    case bird
    case insect

    func printSelf() {
        switch self {
            case .bird:
                print("Это птица")
            case .insect:
                print("Это насекомое")
        }
    }
}

let myBird = FlyingSpecie.bird
myBird.printSelf() // Напечатает - Это птица
```

Перечисление с типом значений

Для перечисления можно указать тип значений. Например, чтобы с каждым значением из перечисления `FlyingSpecie` была связана строка, нужно написать:

```
enum FlyingSpecie: String {
    case bird = "птица"
    case insect = "насекомое"
}
```

Чтобы получить связанное значение, используют `rawValue`:

```
let myBird = .bird
print(myBird.rawValue) // Выведет на экран "птица"
```

Когда задают связанный тип для перечисления, можно создавать перечисление из значения связанного типа:

```
let myBird = FlyingSpecie(rawValue: "птица")
```

Ассоциированные значения

С значением перечисления можно связать (ассоциировать) дополнительный тип:

```
enum FlyingSpecie {  
    case bird(Int, String)  
    case insect(Int, String)  
}
```

Теперь при создании значения перечисления можно связывать с ним дополнительные данные:

```
let newBird = FlyingSpecie.bird(21, "Колибри")
```

Для обработки связанного типа нужно использовать разыменовывание в операторе switch:

```
for specie in catalog {  
    switch specie {  
    case .bird(let width, let name):  
        print("Это птица: \$(name). Размах крыльев: \$(width)")  
    case .insect(let size, let name):  
        print("Это насекомое: \$(name). Размер: \$(size)")  
    }
```

Яндекс Практикум