Основы Swift

# Перечисления

# Введение.
# Перечисления (Enumeration)

Перечисление или Enumeration — определяет тип для группы схожих значений.

В отличие от C/Objective-C значения не ограничены типом Int, могут использоваться String, Double и другие типы.

Enumeration — тип высшего класса, поддерживают вычисляемые свойства, инициализаторы, методы, расширения и протоколы.

# Синтаксис создания enumeration

Int значения не присваиваются по умолчанию

```
enum CompassPoint {
    case north
    case south
    case east
    case west
}

enum CompassPoint {
    case north, south, east, west
            // для удобства в 1 строку
}
```

# Enumeration определяет новый тип

Именование в единственном числе.
UpperCamelCase

```
var directionToWest = CompassPoint.west
```

Если тип переменной известен, можно использовать краткую запись

```
var directionToEast: CompassPoint
directionToEast = .east
```

# Enumeration в Switch выражении

```
directionToAntarctica = .south
switch directionToAntarctica {
        // знач. переменной
    case .north:
        // значение перечисления
        print("Север")
    case .south:
        print("Юг")
    case .east:
        print("Восток")
    case .west:
        print("Запад")
} // Напечатает “Юг”
```

Можно использовать default для того, чтобы не перечислять все значения.

```
directionToAntarctica = .south

switch directionToAntarctica {
        // знач. переменной
    case .north:
        // значение перечисления
        print("Север")
    default:
        print("Не север")
} // Напечатает "Не север"
```

# Associated Values в Enumerations

Ассоциированные значения

```
enum ColorCode {
    case Color(UIColor)
    case RGB(Int, Int, Int)
    case name(String)
}
```

```
var webColor = ColorCode.RGB(0, 0, 128)
webColor = ColorCode.name("Navy")
```

# Associated Values
# в Switch выражении

```
switch webColor {
    case let .Color(color):
        print("Color: \(color)")
    case .RGB(let red, var green, let blue):
        print("RGB: \(red), \(green), \(blue)")
    case let .name(colorName):
        print("Color is called \(colorName).")
}
// Напечатает "Color is called Navy"
```

# Raw Values

RAW значения могут быть представлены типом String, Character, Int или Float.

```
enum ASCIIControlCharacter: Character {
    case tab = "\t"
    case lineFeed = "\n"
    case carriageReturn = "\r"
}
```

В отличие от Associated, Raw устанавливаются в момент определения перечисления.

# Неявное присвоение значение Enumerations

Для Int и String перечислений Raw значения присваиваются неявно.

```
enum Planet: Int {
    case mercury = 1, venus, earth, mars, jupiter,
    saturn, uranus, neptune
}
let earthsOrder = Planet.earth.rawValue
                                    // earthsOrder is 3

enum CompassPoint: String {
    case north, south, east, west
}
let sunsetDirection = CompassPoint.west.rawValue
                                    // sunsetDirection is "west"
```

# Инициализация с помощью Raw значений

При инициализации Enumeration с помощью Raw значения получаем перечисление с соответствующим значением или nil.
Проверка при выполнении кода.
Failable инициализатор.

```
let possiblePlanet = Planet(rawValue: 7)
    // possiblePlanet is of type Planet?
                                and equals Planet.uranus
   // Обратите внимание на конструкцию if let
let positionToFind = 11
if let somePlanet = Planet(rawValue: positionToFind) {
    print(somePlanet.rawValue)
} else {
    print("nil value")
}
```

# Рекурсивное перечисление

Перечисления в качестве типа
для значения перечисления.

Динамическое выделение памяти.
Рекурсивное использование.

```
enum ArithmeticExpression {
    case number(Int)
    indirect case addition(ArithmeticExpression,
                           ArithmeticExpression)
    indirect case multiplication(ArithmeticExpression,
                           ArithmeticExpression)
}
```