Swift

Wprowadzenie do programowania

Krótki wstęp

- Twórcy: Chris Latter wraz z Apple Inc.
- Data wydania: 9 września 2015
- rozszerzenie: „.swift”
- *„[…] drawing ideas from Objective-C, Rust, Haskell, Ruby, Python, C#, CLU, and for too many others to list.”*
- Zorientowany Obiektowo (Aleeeee zawiera również elementy programowania strukturalnego)
- Program: X-code

Spis treści:

- Zmienne i stałe
- tablice
- kolekcje
- Pętle
- Instrukcje warunkowe
- Funkcje
- Obiekty i klasy

Zmienne i stałe

UWAGA ! średnik pojawia się i znika taka rola … magika ???

Zmienne:

- słowo kluczowe - „var”
- deklaracja typu - niekonieczna, ale możliwa
- konwersja z jednego typu na drugi - możliwa.
- inny sposób konwersji dla typu String

Stałe:

- słowo kluczowe - „let”
- nadaje się im wartość tylko raz ! nie można zmieniać jej początkowo nadanej wartości
- używać można wiele razy w programie
- konwersja typu możliwa

```
5  var zmienna = 40; var zmienna2 = 2; // Oooooo średniki ?!
6  // Ale jak to tak ?!
7
8  zmienna2 = 5
9
10 var zmienna3: String = "Hello World" //jawna deklaracja typu
11
12 //uniwersalny sposób konwersji typu
13 zmienna3 += String(zmienna2)
14
15 //inny sposób dla typu String
16 zmienna3 += "\(zmienna)"
17
18 let stala = 40
19 //stala = 5 <-- tak nie wolno !!!!
20
21 zmienna3 += "\(stala)" //konwersja możliwa
```

Tablice

- deklaracja za pomocą „[]”
- operowanie na tablicy za pomocą indeksów (indeksowanie od 0)
- wyświetlanie całej zawartości tablicy - print(nazwa_tablicy)
- dodawanie nowych elementów za pomocą metody „append(el_dod)” lub operatora „+=”
- [wartosc…wartosc2] <— określenie zakresu tablicy

```swift
var tablica = [3,5,6,7] //deklaracja tablicy typu int
//kompilator sam rozpoznaje typ

tablica[1] = 4
print(tablica) // wyniki wyświetlony na ekranie: [4,5,6,7]

//jawna deklaracja typu
var tablica2: [String] = ["swift", "Objective-C", "C"]

var tablica3 = [Int]() // pusta tablica z określonym typem
var tablica4 = [] // pusta tablica bez określonego typu

tablica.append(3) // mega proste dodawanie elementów do tablicy
tablica += [4,5] //alernetywny sposob na dodanie elementow do tablicy

tablica[3...6]// określenie zakresu tablicy
```

Kolekcje

- Tworzone za pomocą składni:
  var/let nazwa_kolekcji =
  [key:val, …., key_n:val_n]

```
41 //Deklaracja kolekcji ze wstawionymi już elementami
42 var shoppingList = ["apples" : 4, "cheese" : 1, "milk" : 2 ]
43
44 //Deklaracja pustej kolekcji
45 var shoppingList2 = [:]
46
47 //jawna deklaracja typów
48 var shoppingList3: [String: Int] = ["eggs": 1]
49
50 print(shoppingList.count) //zwróci licze elementów kolekcji
51 print(shoppingList.isEmpty)// zwróci logiczną odpowiedź na pytanie
52 // czy kolekcja jest pusta
53
54 shoppingList["grapes"] = 2 //dodawanie elementow
55 print(shoppingList.count)//zwróci teraz wartosc 4
```

- jawne określenie typu var/let nazwa: [typ : typ]

- Przykładowe operacje na kolekcjach: .count i .isEmpty

- dodawanie elementu do kolekcji

Instrukcje warunkowe

- nawiasy niepotrzebne (dla przywiązanych możliwe)
- for-in dla iterowania po tablicach, kolekcjach itd.
- for, while oraz do-while zachowują się dokładnie tak samo jak w większości języków
- instrukcje warunkowe w pętlach muszą być typu bool

```
58  for index in tablica2 {
59      index
60  }
61
62  for index in 1...4 { index }
63
64  for var i = 0 ; i < 10 ; i++ {
65     print("show this messager \(i)")
66  }
67
68  var i = 0
69  while i < 4 {
70  print(i++)
71  }
72
73  do {
74   print(i++)
75  } while ( i < 4)
```

Instrukcje warunkowe c.d.

- if else jest prawie identyczne jak w większości języków programowania

- switch - swift wprowadza kilka nowych ciekawych rozwiązań m.in. użycie zakresu (np. 5…9) czy ciekawa funkcja „where”, która pozawala na dodatkowe warunki wykonywania instrukcji

- brak konieczności „break” na końcu case.

- Konieczny domyślny kod umieszczony po etykiecie „default”

```
77  i = 4
78  if i < 4 { print("to ja sie wyswietle")}
79  else if i == 4 { print("nieee bo ja")}
80  else {print("nie wyswietle sie")}
81
82  var wybor = 4
83  var zmienna100 = 7
84
85  switch wybor {
86      case 2,3: print("jakis kod")
87      case 5...99: print("przyklad zakresu")
88      case 4...99 where wybor + 3 == zmienna100:
89          print("przyklad uzycia where")
90  default: print("default jest obowiazakowe")
91  }
```

Funkcje

- Deklaracja słowem kluczowym „func”
- oznaczenie zwracanego typu po znaku „->”
- kilka zwracanych elementów
- odnoszenie się do danych zmiennych zwracanych po kropce
- możliwość zwracania funkcji przez funkcje
- deklaracja argumentów podobnie jak w Objective-C : nazwa: typ, …

```
1  func calculateStatistics(scores: [Int]) -> (min:
           Int, max: Int, sum: Int) {
2      var min = scores[0]
3      var max = scores[0]
4      var sum = 0
5
6      for score in scores {
7          if score > max {
8              max = score
9          } else if score < min {
10             min = score
11         }
12         sum += score
13     }
14
15     return (min, max, sum)
16 }
17 let statistics = calculateStatistics([5, 3, 100,
           3, 9])
18 statistics.sum
19 statistics.2
```

Obiekty i klasy

- Tworzenie klasy za pomocą słowa kluczowego „class”

- składniki i metody tworzone tak samo jak funkcje i zmienne oraz stałe

- tworzenie obiektu za pomocą składni var zm = nazwa_klasy()

- odnoszenie się do obiektów klasy oraz metod po kropce: nazwa_klasy.składnik

- tworzenie konstruktora za pomocą składni: init(argumenty){//blok kodu}

```
1  class Shape {
2      var numberOfSides = 0
3      func simpleDescription() -> String {
4          return "A shape with \(numberOfSides)
           sides."
5      }
6  }
```

```
1  var shape = Shape()
2  shape.numberOfSides = 7
3  var shapeDescription = shape.simpleDescription()
```

```
5      init(name: String) {
6          self.name = name
7      }
```

Teraz troszkę praktyki...