Mảng (Array)

Mảng

Nhiều phần tử cùng kiểu, tập hợp lại với nhau thành một thứ tự tạo thành mảng

- Khởi tạo mảng:

```
// Một mảng các số nguyên 1, 2, 3
let myNumbers = [1, 2, 3]

// Mảng String rỗng
var emptyStrings: [String] = [] // hoặc var emptyStrings = [String]()

// Khởi tạo mảng có 10 phần tử số nguyên, các phần tử có giá trị giống nhau
var digits = [Int](repeating: 0, count: 10)
```

Thao tác với mảng

```swift
var numberArray = [1, 43, 23, 0]

// Kiểm tra mảng rỗng
numberArray.isEmpty

// Kiểm tra mảng có bao nhiêu phần tử
numberArray.count

// Truy cập phần tử trong mảng bằng index
print(numberArray[2])

// Truy cập nhanh đến phần tử đầu / cuối mảng
numberArray.first; numberArray.last

// Duyệt mảng
for i in numberArray {
    print(i)
}

// Duyệt mảng lấy index
for (index, value) in numberArray.enumerated() {
    print("Chỉ số \(index) có giá trị \(value)")
}
```

Thao tác với mảng

```
// Thêm một phần tử
numberArray.append(19)

// Thêm một mảng phần tử
numberArray += [12, 0, 5] // hoặc: numberArray.append(contentOf: [12, 0, 5])

// Chèn 1 phần tử vào vị trí index
numberArray.insert(28, at: 2)

// Xoá khỏi mảng một phần tử theo vị trí index
numberArray.remove(at: 3)

// Xoá phần tử đầu, cuối của mảng
numberArray.removeFirst(); numberArray.removeLast()

// Xoá tất cả phần tử của mảng
numberArray.removeAll()
```

Map

- Các operator chuyển đổi array trong swift gồm có : Map, Filter, Flatmap, Reduce, Compact Map

- **Map:** Lặp qua một collection và áp dụng một thao tác cho mỗi phần tử trong collection đó

map nhận các closure là các argument và trả về kết quả như cách thực hiện vòng lặp thông thường

Ví dụ về Map

```
let numberArray = [1, 2, 3, 4, 5]
var squareArray: [Int] = [ ]
// cách truyền thống
for number in numberArray{
    squareArray.append(number * number)
}

//Cach 1:
let squareArray1 = numberArray.map { (value: Int) -> Int in
    return value * value
}

//Cach 2
let squareArray2 = numberArray.map { (value: Int) in
    return value * value
}

//Cach 3
let squareArray3 = numberArray.map { value in
    value * value
}

//Cach 4
let squareArray4 = numberArray.map { $0 * $0 }

print(squareArray, squareArray1, squareArray2, squareArray3, squareArray4)
```

Filter

- ***Filter**: Duyệt qua collection và trả về một mảng chứa các phần tử đáp ứng điều kiện.*

```
// cách truyền thống
var filterArray1: [Int] = [ ]
for number in numberArray {
    if (number % 2 == 0) {
        filterArray1.append(number)
    }
}

// sử dụng filter
let filterArray2 = numberArray.filter { $0 % 2 == 0 }

print(filterArray1, filterArray2)
```

Reduce

- ***Reduce**: Kết hợp tất cả các phần tử trong collection để tạo một giá trị mới*

```
// cách truyền thống
var sum = 0
for number in numbers {
    sum += number
}

//Reduce
let sum1 = numbers.reduce(0, { $0 + $1 })
let sum2 = numbers.reduce(0, +)
print(sum, sum1, sum2)

// có thể sử dụng với string
let stringArray = ["one", "two"]
let oneTwo = stringArray.reduce("", +)
print(oneTwo)
```

Flatmap

- ***Flatmap*** *giúp chúng ta đưa tất cả các dữ liệu trong các tập con về 1 tập duy nhất*

```
let arrayInArray = [[1 ,2, 3], [4, 5, 6]]

// cách truyền thống
var resultArray: [Int] = [ ]
for array in arrayInArray {
    resultArray += array
}
print(resultArray)

// sử dụng flatMap
let flattenedArray = arrayInArray.flatMap{ $0 }
print(flattenedArray)

// hàm flatMap xóa nil khỏi collection
let people: [String?] = ["A", nil, "B", nil, "C"]
let validPeople = people.flatMap { $0 }

print(validPeople)
```

Compact Map

- **Compact Map:** Tương tự với Map, Compact Map cũng lặp qua một collection và áp dụng thao tác cho mỗi phần tử trong collection đó nhưng kết quả trả về là một mảng không có nil, còn map trả về kết quả là một mảng có chứa nil

```
let possibleNumbers = ["1", "2", "three", "///4///", "5"]

let mapped: [Int] = possibleNumbers.map { str in Int(str) ?? 0 }
print(mapped) // [Optional(1), Optional(2), nil, nil, Optional(5)]

let compactMapped: [Int] = possibleNumbers.compactMap { str in Int(str) }
print(compactMapped) // [1, 2, 5]

let flattenedArray = possibleNumbers.flatMap{ $0 }
print(flattenedArray)
```

Chaning

- Chúng ta có thể kết hợp nhiều phương thức chuyển đổi mảng để ra được kết quả mong muốn khi thao tác với mảng

```
// tính tổng bình phương của tất cả các số chẵn từ mảng của các mảng
let arrayInArray = [[1, 2, 3], [4, 5, 6]]
let arrayValue = arrayInArray.flatMap{ $0 }.filter{$0 % 2 == 0}
let sumArray = arrayValue.map{ $0 * $0}.reduce(0, +)
print(sumArray) // 56

// Bước làm:
// đầu tiên hàm flatMap{ $0 } gộp các mảng con: [1, 2, 3, 4, 5, 6]
// sau đó hàm filter{ $0 % 2 == 0} lấy ra các số chẵn: [2, 4, 6]
// tiếp theo map { $0 * $0 } thực hiện bình phương: [4, 16, 36]
// cuối cùng hàm reduce(0, +) sẽ tính tổng: 4 + 16 + 36 = 56
```