Function

Hàm (Function)

- Trong Swift, một hàm được định nghĩa bởi từ khoá **func,** hàm có tên cụ thể, hàm có thể có **không** hoặc **nhiều** tham số, và **có** hoặc **không có** kiểu trả về

- Hàm gồm 2 phần là **khai báo hàm** và **định nghĩa hàm**

- **Khai báo hàm** là thông báo với trình biên dịch về tên hàm, tham số truyền vào, kiểu trả về

- **Định nghĩa hàm** là phần thân hàm (xử lý của hàm)

Ví dụ về định nghĩa hàm

```
// Định nghĩa một hàm
// Tên hàm: sayHello
// Tham số: name, kiểu String
// Trả về (return): String

func sayHello(name: String) -> String {

    // Nếu name rỗng
    if name.isEmpty {
        return "Hello every body!"
    }

    // Nếu name có giá trị
    return "Hello" + name
}
```

Ví dụ 2 về định nghĩa hàm

```
// Định nghĩa một hàm, không có tham số, không có kiểu trả về
func testSayHello(){

    // Gọi hàm sayHello(), truyền vào một string rỗng
    let greeting1 = sayHello(name: "")
    print("greeting1: " + greeting1)

    // Gọi hàm sayHello(), truyền vào một string rỗng
    let greeting2 = sayHello(name: "Swift")
    print("greeting2: " + greeting2)
}


// Gọi hàm testSayHello()
testSayHello()
```

Hàm trả về 1 giá trị

```
// Định nghĩa một hàm tính tổng 3 số Int, trả về kiểu Int.
func sum(a: Int, b: Int, c: Int) -> Int {
    return a + b + c
}

// Định nghĩa một hàm để tìm số lớn nhất trong 3 số
func max3So(a: Int, b: Int, c: Int) -> Int {

    var m = a
    if m < b {
        m = b
    }
    if m > c {
        return m
    }

    return c
}
```

Hàm trả về nhiều giá trị (Tuples)

```
func getMinMax(arrs: [Int]) -> (min: Int, max: Int) {

    // Nếu mảng không có phần tử thì trả về (0, 0)
    if arrs.count == 0 {
        return (0, 0)
    }

    var min = arrs[0]
    var max = arrs[0]

    for a in arrs {
        if min > a {
            min = a
        }

        if max < a {
            max = a
        }
    }

    return (min, max)
}
```

Hàm với tham số Variadic

- **Swift** sử dung **variableName: DataType…** để đánh dấu một tham số là **Variadic**

```
// Một hàm với các tham số variadic: nums
// Tham số nums: giống như một mảng các số Int
func sum(nums: Int...) -> Int {
    var tong = 0
    for i in nums {
        tong += i
    }
    return tong
}

// in hàm truyền vào 3 số
print(sum(nums: 1, 2, 4))

// in hàm truyền vào 7 số
print(sum(nums: 3, 23, 1, 0, 58, 5, 9))
```

Hàm với tham số inout

- Tham số của hàm mặc định là hằng số, do đó nếu muốn thay đổi giá trị của các tham số và muốn nó tồn tại sau lời gọi hàm kết thúc thì chúng ta định nghĩa hàm với tham số **inout**

```
// Hàm hoán vị 2 số nguyên
func swap( a: inout Int, b: inout Int) {
    let t = a
    a = b
    b = t
}

var a = 10
var b = 17

// Gọi hàm
swap(&a, &b)

// Sau khi chạy hàm: a là 17, b là 10
print("a = \(a), b = \(b)")
```

Hàm lồng nhau

- Swift cho phép viết một hàm bên trong một hàm khác, hàm này được sử dụng trong nội bộ của hàm cha

```
// Hàm trả về tiền thuế, dựa trên mã quốc gia và lương
func getTaxAmount(countryCode: String, salaryAmount: Int) -> Int {
    func getUSATaxAmount(salaryAmount: Int) -> Int {
        return 15 * salaryAmount / 100
    }

    func getVietNamTaxAmount(salaryAmount: Int) -> Int {
        return 10 * salaryAmount / 100
    }

    if countryCode == "$" {
        // USA
        return getUSATaxAmount(salaryAmount: salaryAmount)
    } else if countryCode == "VND" {
        // VietNam
        return getVietNamTaxAmount(salaryAmount: salaryAmount)
    }

    // Các quốc gia khác
    return 5 * salaryAmount / 100
}
```