Основы Swift

Pattern Matching

Wildcard pattern

```
func someCalculation() -> Int { return 5 }

_ = someCalculation()

for _ in 1...10 {
    // ...
}
```

Identifier pattern

```
let someString = "string"
```

Value binding pattern

```
let tuple = (1, 2)

if case (let a, let b) = tuple {
    print("a: \(a), b: \(b)")
}

if case let (a, b) = tuple {
    print("a: \(a), b: \(b)")
}
```

Tupel pattern

```
let (a, b): (Int, Int) = (1, 2)
```

Enumeration case pattern

```
enum SomeType {
    case type1
    case type2
    case type3
}
let type = SomeType.type1
```

```
switch type {
case .type1:
    // ...
    break
case .type2:
    // ...
    break
case .type3:
    // ...
    break
}
```

Optional pattern

```
let optValue: Int? = 654

if case let value? = optValue {
    // ...
}

if case let .some(value) = optValue {
    // ...
}
```

Type casting pattern

```
let someValue: Any = 0.0

switch someValue {
case 0 as Double:
    // ...
    break

case 0 as Int:
    // ...
    break

case let anotherInt as Int:
    // ...
    break

default:
    // ...
    break
}
```

Expression pattern

```
func ~= (pattern: Int, value: String)
-> Bool {
    if let intValue = Int(value) {
        return intValue == pattern
    }
    return false
}
```

```
switch "123" {
case 321:
    // ...
    break
case 123:
    // ...
    break
default:
    // ...
    break
}
```