链滴

Sequence1

作者：someone756
原文链接：https://ld246.com/article/1526799441086
来源网站：链滴

```
import Foundation

print("Hello, World!")

//protocol IteratorProtocol {
// associatedtype Element
// mutating func next() -> Element?
//}

struct Ones: IteratorProtocol {
    typealias Element = Int
    mutating func next() -> Int? {
        return 1
    }
}

var ones = Ones()
print(ones.next())
print(ones.next())

//struct Fibonacci: IteratorProtocol {
// typealias Element = Int
//
// private var state = (0, 1)
//
// mutating func next() -> Int? {
```

原文链接：Sequence1

```
// let nextNumber = state.0

// state = (state.1, state.0 + state.1)

// return nextNumber

// }

//

//}

//var fibonacci = Fibonacci()
//print(fibonacci.next())
//print(fibonacci.next())
//print(fibonacci.next())
//print(fibonacci.next())
//print(fibonacci.next())
//print(fibonacci.next())
//
//var fibonacci1 = fibonacci
//print(fibonacci1.next())
//protocol Sequence {
// associatedtype Element

// associatedtype Iterator: IteratorProtocol where Iterator.Element == Element

//

// func makeIterator() -> Iterator

//}



struct FiboIter: IteratorProtocol {

    typealias Element = Int

    var state = (0, 1)

    mutating func next() -> Int? {

        let nextNumber = state.0

        self.state = (state.1, state.0 + state.1)

        return nextNumber

    }

}
```

原文链接：Sequence1

```
struct Fibonacci: Sequence {
    typealias Element = Int
    func makeIterator() -> FiboIter {
        return FiboIter()
    }
}

var fibs = Fibonacci()
var fib1 = fibs.makeIterator()
print(fib1.next())
print(fib1.next())
print(fib1.next())
print(fib1.next())
print(fib1.next())
print(fib1.next())
print(fib1.next())

for n in  Fibonacci() {
    if n <= 5 {
        print(n)
    } else {
        break
    }
}

var arrayInt: [Int] = []
var sumInt = arrayInt.reduce(0, +)
```

原文链接：Sequence1

```
print(sumInt)


var arrStr: [String] = []
var sumStr = arrStr.reduce("", +)
print(sumStr)
func reduce1(of sequence: S, _ partical: (E, E) -> E) -> E?
    where S: Sequence, E == S.Element {
        var iter: S.Iterator = sequence.makeIterator()
        guard var accumulated = iter.next() else {
            return nil
        }
        while let element = iter.next() {
            accumulated = partical(accumulated, element)
        }
        return accumulated
}
extension  Sequence {
    //FixedWidthInteger, SignedInteger , 先就这么着吧,:slightly_smiling:
    func reduceFromZero(of sequence: S, _ partical: (E, E) -> E) -> E where S: Sequence, E: Fixe
WidthInteger, E: SignedInteger, E == S.Element {
        var iter = sequence.makeIterator()
        guard var accumulated = iter.next() else { return 0 }
        while let element = iter.next() {
            accumulated = partical(accumulated, element)
        }
        return accumulated
    }
}
```

原文链接：Sequence1

```
print(reduce1(of: arrayInt, +))

print([1].elementsEqual([1]))
```