```
import Foundation
/// 剑指 Offer 46. 把数字翻译成字符串
/// 给定一个数字, 我们按照如下规则把它翻译为字符串:
/// 0 翻译成 "a", 1 翻译成 "b", ....., 11 翻译成 "l", ....., 25 翻译成 "z"。
/// 一个数字可能有多个翻译。请编程实现一个函数,用来计算一个数字有多少种不同的翻译方法。
/// 示例 1:
/// 输入: 12258
/// 输出:5
/// 解释: 12258有5种不同的翻译, 分别是"bccfi", "bwfi", "bczi", "mcfi"和"mzi"
class Solution {
   func translateNum( num: Int) -> Int {
       if String(num).count == 1 { return 1 }
       /// F(0) = F(1) = 1
       var(a, b): (Int, Int) = (1, 1)
       /// 先将 num 转为字符串并计算 num 数字位数
       let (numStr, length): (String, Int) = (String(num), String(num).count)
       let startIndex = numStr.startIndex
       for id in 2...length {
           /// 比较直观的写法,但不是很推荐比较浪费内存
           let slice = String(Array(numStr)[id-2..<id])</pre>
           /// 由于题目限定当组合的 2 位数字大于 25 时候无法翻译
           /// 因此要扣除这些无法翻译的次数,此时获取到组合大于25的数字所在的索引,
           /// 其前面所有数字构成的翻译数目即为需要扣除的
           /// 同时需要注意可能存在 "05" 这种第一个字符为 "0" 的字符串
           var c = (slice.first! != "0" && slice <= "25") ? a + b : a</pre>
           b = a
           a = c
       }
       return a
   }
}
let solution = Solution()
print(solution.translateNum(12258))
                                     // 5
                                      // 2
print(solution.translateNum(25))
print(solution.translateNum(542))
                                     // 1
print(solution.translateNum(506))
                                     // 1
print(solution.translateNum(18822))
                                     // 4
print(solution.translateNum(18580))
                                     // 2
```