```swift
import Foundation
// 剑指 Offer 67. 把字符串转换成整数
// 写一个函数 StrToInt，实现把字符串转换成整数这个功能。不能使用 atoi 或者其他类似的库函数。
//
// 首先，该函数会根据需要丢弃无用的开头空格字符，直到寻找到第一个非空格的字符为止。
//
// 当我们寻找到的第一个非空字符为正或者负号时，则将该符号与之后面尽可能多的连续数字组合起来，
//
作为该整数的正负号；假如第一个非空字符是数字，则直接将其与之后连续的数字字符组合起来，形成整数。
//
//
该字符串除了有效的整数部分之后也可能会存在多余的字符，这些字符可以被忽略，它们对于函数不应该造成
影响。
//
// 注意：假如该字符串中的第一个非空格字符不是一个有效整数字符、字符串为空或
// 字符串仅包含空白字符时，则你的函数不需要进行转换。
//
// 在任何情况下，若函数不能进行有效的转换时，请返回 0。
//
// 说明：
// 假设我们的环境只能存储 32 位大小的有符号整数，那么其数值范围为
// [−2^31,  2^31 − 1]。如果数值超过这个范围，请返回
// INT_MAX (231 − 1) 或 INT_MIN (−231)。
class Solution {
    func strToInt(_ str: String) -> Int {
        var _str = str
        /// 删除开头与结尾的空格
        _str = _str.trimmingCharacters(in: .whitespaces)
        if let first = _str.first {
            if first.isNumber {
                _str = deletePrefixZero(_str)
                return getTheNumber(_str)
            } else if (first == "+") || (first == "-") {
                let mark = first
                _str = String(_str.dropFirst())
                if let second = _str.first {
                    if second.isNumber {
                        _str = deletePrefixZero(_str)
                        return (mark == "+") ? getTheNumber(_str):
                         getTheNumber(_str, true)
                    } else { return 0 }
                } else { return 0 }
            } else { return 0 }
        }
        return 0
    }

    private func deletePrefixZero(_ str: String) -> String {
        var _str = str
```

```
        var id = 0
        for c in str {
            if c != Character("0") { break }
            else { id += 1 }
        }
        let start = str.startIndex
        let end = str.endIndex
        return String(str[str.index(start, offsetBy: id)..<end])
    }

    private func getTheNumber(_ str: String, _ negative: Bool = false) -> Int {
        var nums: [Int] = []
        for c in str {
            if !c.isNumber {
                break
            } else {
                if let val = Int(String(c)) {
                    nums.append(val)
                }
            }
        }
        if nums.count > 10 { return negative ? Int(Int32.min) : Int(Int32.max)
         }
        var length: Int = 1
        var res: Int = 0
        for num in nums.reversed() {
            if (Int(Int32.max) - res) < num * powTen(length-1) {
                return negative ? Int(Int32.min) : Int(Int32.max)
            } else {
                res += num * powTen(length-1)
                length += 1
            }
        }
        return negative ? 0-res: res
    }

    private func powTen(_ n: Int) -> Int {
       var res: Int = 1
       var _n = n
       while _n != 0 {
           res *= 10
           _n -= 1
       }
       return res
    }
}
```