

algorithm

何书文

1201220707@pku.edu.cn

2020 年 1 月 7 日

# 目录

第一章	list	2
1.1	list . . . . .	2

# Chapter

## list

### list

```
1    package main
2
3    import (
4        "fmt"
5    )
6
7    type ListNode struct{
8        Val int
9        Next *ListNode
10    }
11
12    func main1() {
13        headNode := &ListNode{}
14        listData := headNode
15
16        InsertTail(1, listData, headNode)
17        PrintList(listData)
18
19        InsertTail(2, listData, headNode)
20        PrintList(listData)
21
22        InsertTail(3, listData, headNode)
23        PrintList(listData)
24    }
25
26    func InsertTail(value int, list, position *ListNode) {
27        tempCell := new(ListNode)
28
29        if tempCell == nil{
```

```
30         fmt.Println("out of space")
31     }
32
33     tempCell.Val = value
34     tempCell.Next = position.Next
35     position.Next = tempCell
36 }
37
38 func PrintList(list *ListNode) {
39     if list.Next != nil {
40         fmt.Print(list.Val, "->")
41         PrintList(list.Next)
42     } else {
43         fmt.Println(list.Val)
44     }
45 }
46
47 // 给出两个非空的链表用来表示两个非负的整数。其中，它们
48 // 各自的位数是按照逆序的方式存储的，
49 // 并且它们的每个节点只能存储一位数字。
50 // 如果，我们将这两个数相加起来，则会返回一个新的链表来
51 // 表示它们的和。
52 // 您可以假设除了数字0之外，这两个数都不会以0开头。
53 // 示例：
54 // 输入：(2 -> 4 -> 3) + (5 -> 6 -> 4)
55 // 输出：7 -> 0 -> 8
56 // 原因：342 + 465 = 807
57
58 // type ListNode2 struct {
59 //     Val int
60 //     Next *ListNode2
61 // }
62
63 type List struct {
64     headNode2 *ListNode // head node
65 }
```

```
65     func Insert2(value int, list *ListNode, position *
        ListNode) {
66         tempCell := new(ListNode)
67         if tempCell == nil {
68             fmt.Println("out of space")
69         }
70         tempCell.Val = value
71         tempCell.Next = position.Next
72         position.Next = tempCell
73     }
74
75     func PrintList2(list *ListNode) {
76         if list.Next != nil {
77             fmt.Println(list.Val)
78             PrintList2(list.Next)
79         } else {
80             fmt.Println(list.Val)
81         }
82     }
83
84     func main() {
85         l1 := new(ListNode)
86         listDate := l1
87         // insert data to l1
88         Insert2(9, listDate, l1)
89         Insert2(7, listDate, l1)
90         Insert2(5, listDate, l1)
91         l2 := new(ListNode)
92         //
93         listDate2 := l2
94         // insert data to l1
95         Insert2(4, listDate2, l2)
96         Insert2(2, listDate2, l2)
97         Insert2(8, listDate2, l2)
98         l3 := addTwoNumbers(l1, l2)
99         PrintList(l3)
100     }
101
```

```
102     func addTwoNumbers(l1 *ListNode, l2 *ListNode) *ListNode
103     {
104         promotion := 0      // 进位值，只可能为0或1
105         var head *ListNode // 结果表的头结点
106         var rear *ListNode // 保存结果表的尾结点
107         for nil != l1 || nil != l2 {
108             sum := 0
109             if nil != l1 {
110                 sum += l1.Val
111                 l1 = l1.Next
112             }
113             if nil != l2 {
114                 sum += l2.Val
115                 l2 = l2.Next
116             }
117             sum += promotion
118             promotion = 0
119
120             if sum >= 10 {
121                 promotion = 1
122                 sum = sum % 10
123             }
124
125             node := &ListNode{
126                 sum,
127                 nil,
128             }
129
130             if nil == head {
131                 head = node
132                 rear = node
133             } else {
134                 rear.Next = node
135                 rear = node
136             }
137         }
138     }
```

```
139         if promotion > 0 {
140             rear.Next = &ListNode{
141                 promotion,
142                 nil,
143             }
144         }
145         return head
146     }
```