

作业

1、求n的阶乘。至少使用递归完成一次。

```
1 package main
2
3 import (
4     "fmt"
5 )
6
7 // 循环版
8 func Factorialv1(n int) int {
9     if n < 1 {
10         panic("不能小于1")
11     }
12     p := 1
13     for i := 1; i <= n; i++ {
14         p *= i
15     }
16     return p
17 }
18
19 // 递归1 公式
20 func Factorialv2(n int) int {
21     if n < 1 {
22         panic("不能小于1")
23     } else if n == 1 {
24         return 1
25     }
26     return n * Factorialv2(n-1)
27 }
28
29 // 递归2 循环改递归
30 // p是初始值1
31 func Factorialv3(n int, p int) int {
32     if n < 1 {
33         panic("不能小于1")
34     } else if n == 1 {
35         return p
36     }
37     return Factorialv3(n-1, p*n)
38 }
39
40 func main() {
41     fmt.Println(Factorialv1(6))
42     fmt.Println(Factorialv2(6))
43     fmt.Println(Factorialv3(6, 1))
44 }
```

2、编写一个函数，接受一个参数n，n为正整数。要求数字必须对齐

```

1  上三角
2
3      1
4      2 1
5      3 2 1
6      4 3 2 1
7      5 4 3 2 1
8      6 5 4 3 2 1
9      7 6 5 4 3 2 1
10     8 7 6 5 4 3 2 1
11     9 8 7 6 5 4 3 2 1
12    10 9 8 7 6 5 4 3 2 1
13    11 10 9 8 7 6 5 4 3 2 1
14    12 11 10 9 8 7 6 5 4 3 2 1

```

上三角所需要的所有数字，实现如下

```

1 package main
2
3 import "fmt"
4
5 func main() {
6     for i := 1; i < 13; i++ {
7         for j := i; j >= 1; j-- {
8             fmt.Print(j, " ")
9         }
10        fmt.Println()
11    }
12 }

```

上面代码至少接近了，只需要实现右对齐，似乎就可以了。但是右对齐最大的问题是一行的宽度，这个问题这么解决呢？

大概估计一下最后一行的宽度就行，给一个较大的宽度就可以了。或者计算出最后一行的宽度。

```

1 package main
2
3 import (
4     "fmt"
5 )
6
7 func main() {
8     width := 40
9     var line string
10    for i := 1; i < 13; i++ {
11        line = ""
12        for j := i; j >= 1; j-- {
13            line += fmt.Sprintf(j) + " "
14        }
15        fmt.Printf("[%2]*[1]s\n", line, width)
16        // fmt.Printf("[%1]*s\n", width, line)
17    }
18 }

```

如果能知道最后一行宽度，更加完美。那怎么做呢？

```
1 package main
2
3 import (
4     "fmt"
5 )
6
7 func PrintTriangle(n int) {
8     last := ""
9     for i := n; i >= 1; i-- {
10         last += fmt.Sprintf("%d ", i)
11     }
12
13     width := len(last)
14     var line string
15     for i := 1; i < n; i++ {
16         line = ""
17         for j := i; j >= 1; j-- {
18             line += fmt.Sprintf("%d ", j)
19         }
20         fmt.Printf("%*s\n", width, line)
21     }
22     fmt.Println(last)
23 }
24
25 func main() {
26     PrintTriangle(12)
27 }
```

两层循环太多了，根据上面的代码是否有所启发？由此，有另一种思路

```
1 package main
2
3 import (
4     "fmt"
5 )
6
7 func PrintTriangle(n int) {
8     last := ""
9     for i := n; i >= 1; i-- {
10         if i == 1 {
11             last += fmt.Sprint(i) // 去掉最后的空格
12         } else {
13             last += fmt.Sprintf("%d ", i)
14         }
15     }
16     width := len(last)
17     // fmt.Println(width, last)
18     for i := width - 1; i >= 0; i-- {
19         if last[i] == 32 {
20             fmt.Printf("%*s\n", width, last[i:])
21         }
22     }
23 }
```

```
22     }
23     fmt.Println(last)
24 }
25
26 func main() {
27     PrintTriangle(12)
28 }
```

