

# 循环和控制

## for

## 条件控制语句

与其他语言大致相同

有两点差别

- 表达式必须是bool型的

一下情况会报错

```
func TestCondition(t *testing.T) {  
    if 5 {  
        t.Log("flag")  
    }  
}
```

```
.\condition_test.go:6:5: non-boolean condition in if statement
```

- if 后面还可以跟赋值语句

```
if var a = b, condition {  
    //todo  
}
```

## switch

### switch 条件

```
switch os := runtime.GOOS; os {  
case "darwin":  
    fmt.Println("OS X.")  
    //break  
case "linux":  
    fmt.Println("Linux.")  
default:  
    // freebsd, openbsd,  
    // plan9, windows...  
    fmt.Printf("%s.", os)  
}
```

```
switch {  
case 0 <= Num && Num <= 3:  
    fmt.Printf("0-3")  
case 4 <= Num && Num <= 6:  
    fmt.Printf("4-6")  
case 7 <= Num && Num <= 9:  
    fmt.Printf("7-9")  
}
```

## 与其他主要编程语言的差异

1. 条件表达式不限制为常量或者整数；
2. 单个 case 中，可以出现多个结果选项, 使用逗号分隔；
3. 与 C 语言等规则相反，Go 语言不需要用break来明确退出一个 case；
4. 可以不设定 switch 之后的条件表达式，在此种情况下，整个 switch 结构与多个 if...else... 的逻辑作用等同

```
func TestCondition(t *testing.T) {  
    for i := 0; i < 5; i++ {  
        switch i {  
            case 0, 2: ← 满足一个条件就进入分支  
                t.Log("是4以内的偶数") ← 执行结束自动 break  
            case 1, 3:  
                t.Log("是4以内的奇数")  
            default:  
                t.Log("This should be four")  
        }  
    }  
}
```

Debug Console (Ctrl+Shift+Y)