指针

Go虽然保留了指针，但与其它编程语言不同的是，在Go当中不支持指针运算以及”->”运算符，而直接采用”.”选择符来操作指针目标对象的成员

- 操作符”&”取变量地址，使用”*”通过指针间接访问目标对象
- 默认值为 nil 而非 NULL

递增递减语句

在Go当中，++ 与 -- 是作为语句而并不是作为表达式

列子1：

```go
package main

import (
    "fmt"
)

func main() {
    a := 1
    var p *int = &a
    fmt.Println(p)
    fmt.Println(*p)
}
```

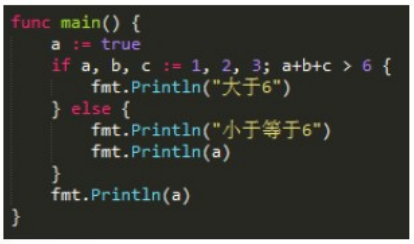
列子2：

```go
package main

import (
    "fmt"
)

func main() {
    a := 1
    a++
    var p *int = &a
    b := 1
    b--
    var f *int = &b
    fmt.Println(p)
    fmt.Println(*p)
    fmt.Println(f)
```

```
    fmt.Println(*f)
}
```

判断语句if

- 条件表达式没有括号
- 支持一个初始化表达式（可以是并行方式）
- 左大括号必须和条件语句或else在同一行
- 支持单行模式
- 初始化语句中的变量为block级别，同时隐藏外部同名变量
- 1.0.3版本中的编译器BUG



if 语句列子1：

```go
package main

import (
     "fmt"
)

func main(){
      a := 10
      if a := 1; a > 0 {
           fmt.Println(a)
      }
      fmt.Println(a)
}
```

if 语句列子2：

```go
package main

import (
    "fmt"
)

func main() {
    if 7 % 2 == 0 {
        fmt.Println("7 is even.")
    } else {
        fmt.Println("7 is odd.")
```

```
    }

    if 8 % 4 == 0 {
        fmt.Println("8 is divisible by 4")
    }

    if num := 9; num < 0 {
        fmt.Println(num, "is negative")
    } else if num < 10 {
        fmt.Println(num, "has 1 digit")
    } else {
        fmt.Println(num, "has multiple digits")
    }
}
```

循环语句for

- Go只有for一个循环语句关键字，但支持3种形式
- 初始化和步进表达式可以是多个值
- 条件语句每次循环都会被重新检查，因此不建议在条件语句中使用函数，尽量提前计算好条件并以变量或常量代替
- 左大括号必须和条件语句在同一行

```
func main() {
    a := 1
    for {
        a++
        if a > 3 {
            break
        }
    }
    fmt.Println(a)
}
```

```
func main() {
    a := 1
    for a <= 3 {
        a++
    }
    fmt.Println(a)
}
```

```
func main() {
    a := 1
    for i := 0; i < 3; i++ {
        a++
    }
    fmt.Println(a)
}
```

for 语句列子1:

```
package main

import (
    "fmt"
)

func main() {
    a := 1
    for {
        a++
        if a > 3 {
            break
        }
        fmt.Println(a)
    }
    fmt.Println("Over")
}
```

for 语句列子2：

```go
package main

import (
    "fmt"
)

func main() {
    a := 1
    for a <= 3 {
        a++
        fmt.Println(a)
    }
    fmt.Println("Over")
}
```

for 语句列子3：

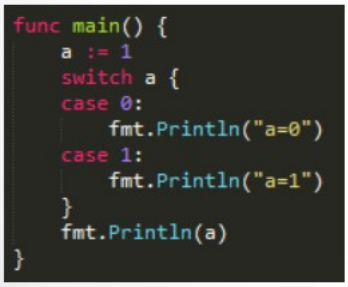
```go
package main

import (
    "fmt"
)

func main() {
    a := 1
    for i := 0; i < 3; i++ {
        a++
        fmt.Println(a)
    }
    fmt.Println("Over")
}
```

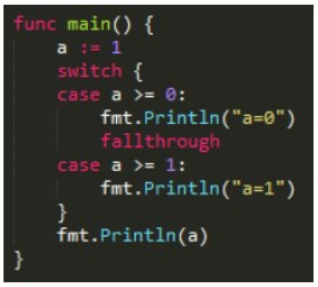
for 语句列子4：

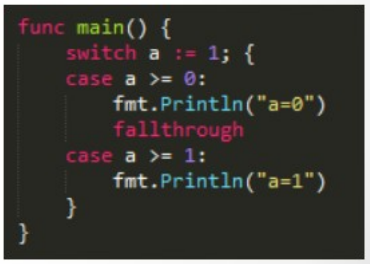
```go
package main

import (
    "fmt"
)

func main() {
    a := "string"
    l := len(a)
    for i := 0; i < l; i++ {
        // a++
        fmt.Println(a)
    }
    fmt.Println("Over")
}
```

选择语句switch

- 可以使用任何类型或表达式作为条件语句
- 不需要写break，一旦条件符合自动终止
- 如希望继续执行下一个case，需使用fallthrough语句
- 支持一个初始化表达式（可以是并行方式），右侧需跟分号
- 左大括号必须和条件语句在同一行







switch 语句列子1：

```go
package main

import (
    "fmt"
)

func main() {
    a := 1
    switch a {
    case 0:
        fmt.Println("a == 0")
    case 1:
        fmt.Println("a == 1")
    default:
        fmt.Println("None")
    }
}
```

switch 语句列子2：

```go
package main

import (
    "fmt"
)

func main() {
    a := 1
    switch {
    case a >= 0:
        fmt.Println("a == 0")
```

```
    case a >= 1:
        fmt.Println("a == 1")
    default:
        fmt.Println("None")
    }
}
```

switch 语句列子3：

```
package main

import (
    "fmt"
)

func main() {
    a := 1
    switch {
    case a >= 0:
        fmt.Println("a == 0")
        fallthrough
    case a >= 1:
        fmt.Println("a == 1")
    default:
        fmt.Println("None")
    }
}
```

switch 语句列子4：

```
package main

import (
    "fmt"
)

func main() {
    // a := 1
    switch a := 1; {
    case a >= 0:
        fmt.Println("a == 0")
        fallthrough
    case a >= 1:
        fmt.Println("a == 1")
    default:
        fmt.Println("None")
    }
}
```

switch 语句列子5：

```
package main

```

```
import (
    "fmt"
)

func main() {
    a := 10
    switch a := 1; {
    case a >= 0:
        fmt.Println("a == 0")
        fallthrough
    case a >= 1:
        fmt.Println("a == 1")
    default:
        fmt.Println("None")
    }
    fmt.Println(a)
}
```

```
package main

import (
    "fmt"
)

func main() {
    // a := 10
    switch a := 1; {
    case a >= 0:
        fmt.Println("a == 0")
        fallthrough
    case a >= 1:
        fmt.Println("a == 1")
    default:
        fmt.Println("None")
    }
    // fmt.Println(a)
}
```

跳转语句goto, break, continue

- 三个语法都可以配合标签使用
- 标签名区分大小写，若不使用会造成编译错误
- Break与continue配合标签可用于多层循环的跳出
- Goto是调整执行位置，与其它2个语句配合标签的结果并不相同

```go
func main() {
LABEL:
    for {
        for i := 0; i < 10; i++ {
            if i > 2 {
                break LABEL
            } else {
                fmt.Println(i)
            }
        }
    }
}
```

```go
func main() {
LABEL:
    for i := 0; i < 10; i++ {
        for {
            fmt.Println(i)
            continue LABEL
        }
    }
}
```

break 语句

break 语句例子1：

```go
package main

import (
    "fmt"
)

func main() {
LABEL1:
    for {
        for i := 0; i < 10; i++ {
            if i > 3 {
                break LABEL1
            }
        }
    }
    fmt.Println("OK")
}
```

goto语句

goto 语句例子1：

```go
package main

import (
    "fmt"
)

func main() {
    for {
        for i := 0; i < 10; i++ {
```

```
            if i > 3 {
                goto LABEL1
            }
        }
    }
LABEL1:
    fmt.Println("OK")
}
```

continue 语句

continue 语句例子1：

```
package main

import (
    "fmt"
)

func main() {
LABEL1:
    for i := 0; i < 10; i++ {
        for {
            continue LABEL1
            fmt.Println(i)
        }
    }
    fmt.Println("OK")
}
```

课堂作业

- 将下图中的continue替换成goto，程序运行的结果还一样吗？
- 请尝试并思考为什么。

```
func main() {
LABEL:
    for i := 0; i < 10; i++ {
        for {
            fmt.Println(i)
            continue LABEL
        }
    }
}
```

Goto是调整执行位置

作业：

```
package main

import (
```

```
    "fmt"
)

func main() {
LABEL1:
    for i := 0; i < 10; i++ {
        for {
            fmt.Println(i)
            continue LABEL1
        }
    }
}
```

```
/home/jiemin/code/GOlang/go/src/break/break  [/home/jiemin/code/GOlang/go/src/break]
0
1
2
3
4
5
6
7
8
9
成功: 进程退出代码 0.
```

把continue更换为goto

```
package main

import (
    "fmt"
)

func main() {
LABEL1:
    for i := 0; i < 10; i++ {
        for {
            fmt.Println(i)
            goto LABEL1
        }
    }
}
```

结果是死循环，一直打印0数值
把标签放到下面

```
package main

import (
    "fmt"
)

func main() {
```

```
8.      for i := 0; i < 10; i++ {
9.          for {
10.             fmt.Println(i)
11.             goto LABEL1
12.         }
13.     }
14. LABEL1:
15.     fmt.Println("OK")
16. }
```

```
/home/jiemin/code/GOlang/go/src/break/break  [/home/jiemin/code/GOlang/go/src/break]
0
OK
成功: 进程退出代码 0.
```