Go语言学习(五)流程控制语句

原创 2016年03月26日 15:36:38 标签：go语言 996

Go语言支持如下的几种流程控制语句：

条件语句，对应的关键字为 if 、 else 和 else if ；

选择语句，对应的关键字为 switch 、 case 和 select （将在介绍channel的时候细说）；

循环语句，对应的关键字为 for 和 range ；

跳转语句，对应的关键字为 goto

在具体的应用场景中，为了满足更丰富的控制需求，Go语言还添加了如下关键字： break 、

continue 和 fallthrough 。

1.1条件语句

关于条件语句，需要注意以下几点：

a.条件语句不需要使用括号将条件包含起来 () ；

b.无论语句体内有几条语句，花括号 {} 都是必须存在的；

c.左花括号 { 必须与 if 或者 else 处于同一行；

d.在if之后，条件语句之前可以添加变量初始化语句，使用 ; 间隔；

e.在有返回值的函数中，不允许将“最终的” return 语句包含在 if...else... 结构中，

否则会编译失败："function ends without a return statement"

1

2

3

4

5

6

7

8

编译失败的案例如下：

1

2

func example(x int)int{

if x==0{

return 5

}else {

return x

}

}

//失败的原因在于，Go编译器无法找到终止该函数的return语句。

1

2

3

4

5

6

7

8

1.2选择语句

在使用 switch 结构时，我们需要注意以下几点：

a.左花括号 { 必须与 switch 处于同一行；

b.条件表达式不限制为常量或者整数；

c.单个 case 中，可以出现多个结果选项；

d.与C,java等语言规则相反，Go语言不需要用 break 来明确退出一个 case ；

e.只有在 case 中明确添加 fallthrough 关键字，才会继续执行紧跟的下一个 case ；

f.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。

1

2

3

4

5

6

7

8

9

根据传入条件的不同，选择语句会执行不同的语句。下面的例子根据传入的整型变量i的不

同而打印不同的内容：

1

2

3

switch i {

case 0:

fmt.Printf("0")

case 1:

fmt.Printf("1")

case 2:

fallthrough

case 3:

fmt.Printf("3")

case 4, 5, 6:

fmt.Printf("4, 5, 6")

default:

fmt.Printf("Default")

}

1

2

3

4

5

6

7

8

9

10

11

12

13

14

运行结果:

i = 0 时，输出 0 ；

i = 1 时，输出 1 ；

i = 2 时，输出 3 ；

i = 3 时，输出 3 ；

i = 4 时，输出 4, 5, 6 ；

i = 5 时，输出 4, 5, 6 ；

i = 6 时，输出 4, 5, 6 ；

i = 其他任意值时，输出 Default

比较有意思的是， switch 后面的表达式甚至不是必需的，比如下面的例子：

1

2

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")

}

1

2

3

4

5

6

7

8

1.3循环语句

与多数语言不同的是，Go语言中的循环语句只支持 for 关键字，而不支持while和do-while结构。

1

2

sum := 0

for i :=0;i<10;i++{

sum +=i

}

1

2

3

4

可以看到比较大的一个不同在于 for 后面的条件表达式不需要用圆括号 () 包含起来。Go语言

还进一步考虑到无限循环的场景，让开发者不用写无聊的 for (;;) {} 和 do {} while(1); ，

而直接简化为如下的写法：

1

2

3

4

sum := 0

for{

sum++

if sum>100{

break

}

}

1

2

3

4

5

6

7

在条件表达式中也支持多重赋值，如下所示：

1

2

a := []int{1,2,3,4,5,6}

for i,j := 0, len(a)-1;i<j;i,j = i+1,j-1{

a[i],a[j] = a[j],a[i]

}

1

2

3

4

使用循环语句时，需要注意的有以下几点。

a.左花括号 { 必须与 for 处于同一行。

b.Go语言中的 for 循环与C语言一样，都允许在循环条件中定义和初始化变量，唯一的区别

是，Go语言不支持以逗号为间隔的多个赋值语句，必须使用平行赋值的方式来初始化多

个变量。

c.Go语言的 for 循环同样支持 continue 和 break 来控制循环.但是它提供了一个更高级的

break ，可以选择中断哪一个循环，如下例：

1

2

3

4

5

6

7

8

for j := 0;j<5;j++{

for i :=0 ;i<10;i++{

if i>5 {

break Tag

}

fmt.Println(i)

}

}

Tag:

1

2

3

4

5

6

7

8

9

上面例子中的break tag是用来终止Tag标签处的外层循环的.

for循环还可以结合range关键字来使用,形如:

1

2

3

for k,v := range [array,slice,map]{ //range可遍历array,slice,map

fmt.Println("key=",k," value=",v)

}

1

2

3

4

1.4跳转语句

goto 语句的语义非常简单，就是跳转到本函数内的某个标签，如：

1

2

func myfunc() {

i := 0

HERE:

fmt.Println(i)

i++

if i < 10 {

goto HERE

}

}

1

2

3

4

5

6

7

8

9

文章来源：http://blog.csdn.net/mchenys/article/details/50987150

版权声明：本文为博主原创文章，未经博主允许不得转载。