Go语言学习(六)函数

原创 2016年03月26日 17:55:38 760

函数

1.1函数的定义

前面我们已经大概介绍过函数，这里我们用一个最简单的加法函数来进行详细说明：

1

2

package main

import (

"fmt"

"errors"

)

func main(){

//直接输出会输出2个值

fmt.Println(Add(1,2))

//如下方式可以只输出一个值

v,\_ := Add(1,2)

fmt.Println(v)

}

func Add(a int,b int)(ret int,err error){

if a<0 || b<0{

err = errors.New("Should be non-negative numbers!")

return

}

return a+b,nil //支持多重返回

}

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

运行结果:

3 <nil>

3

1

2

3

4

如果参数列表中若干个相邻的参数类型的相同，比如上面例子中的a和b,则可以在参数列表

中省略前面变量的类型声明，如下所示：

1

2

3

func Add(a,b int)(ret int, err error){

// ....

}

1

2

3

如果返回值列表中多个返回值的类型相同，也可以用同样的方式合并。

1

2

func Add(a,b int)int{ //如果函数只有一个返回值，也可以这么写：

// ....

}

func Add(a,b int)(res1,res2 int){ //多个相同类型的返回值,可以合并

// ....

}

1

2

3

4

5

6

1.2函数的调用

函数调用非常方便，只要事先导入了该函数所在的包，就可以直接按照如下所示的方式调用函数：

import "mymath"// 假设Add被放在一个叫mymath的包中

1

2

3

c := mymath.Add(1, 2)

注意:

Go语言中函数名字的大小写不仅仅是风格，更直接体现了该函数的可见性，大写表示在其他包可见,

小写则只在本包中可见.这个规则也适用于类型和变量的可见性。

1

2

3

4

1.3不定参数

前面我们经常使用的fmt.Println(),其实就是一个不定参数的函数,你可以输出任意类型和个数的值,用

逗号隔开即可.

1

2

3

func Myfunc(args ...int){

for k,arg := range args{

fmt.Println("k=",k," arg=",arg)

}

}

1

2

3

4

5

这段代码的意思是，函数 myfunc() 接受不定数量的参数，这些参数的类型全部是 int ，所

以它可以用如下方式调用：

1

2

3

Myfunc(2, 3, 4)

Myfunc(1, 3, 7, 13)

1

2

形如 ...type 格式的类型只能作为函数的参数类型存在，并且必须是最后一个参数,这点和java类似;

从内部实现机理上来说，类型 ...type 本质上是一个数组切片,这也是为什么上面的参数args可以用for 循环来获得每个传入的参数。

之前的例子中将不定参数类型约束为 int ，如果你希望传任意类型，可以指定类型为

interface{} 。下面是Go语言标准库中 fmt.Printf() 的函数原型：

1

2

3

4

5

6

func Printf(format string, args ...interface{}) {

// ...

}

1

2

3

用 interface{} 传递任意类型数据是Go语言的惯例用法,通过下面的例子样式多种类型的区分:

1

2

func main(){

var v1 int = 10

var v2 int64 = 234

var v3 string = "abc"

var v4 float32 = 1.234

Myfunc(v1,v2,v3,v4)

}

func Myfunc(args ...interface{}){

for \_,arg := range args{

switch arg.(type){

case int:

fmt.Println(arg," is an int value")

case string:

fmt.Println(arg," is an string value")

case int64:

fmt.Println(arg," is an int64 value")

default:

fmt.Println(arg," is anunknown type.")

}

}

}

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

运行结果:

10 is an int value

234 is an int64 value

abc is an string value

1.234 is anunknown type.

1

2

3

4

5

6

1.4多返回值

比如 File.Read() 函数就可以同时返回读取的字节数和错误信息。如果读取文件成功，则返回值中的n为读取的字节

数，err为nil ，否则 err 为具体的出错信息：

func (file \*File) Read(b []byte) (n int, err Error)

同样，从上面的方法原型可以看到，我们还可以给返回值命名，就像函数的输入参数一样。

返回值被命名之后，它们的值在函数开始的时候被自动初始化为空。在函数中执行不带任何参数

的 return 语句时，会返回对应的返回值变量的值。

Go语言并不需要强制命名返回值，但是命名后的返回值可以让代码更清晰，可读性更强，

同时也可以用于文档。

如果调用方调用了一个具有多返回值的方法，但是却不想关心其中的某个返回值，可以简单

地用一个下划线“ \_ ”来跳过这个返回值，比如下面的代码表示调用者在读文件的时候不想关心

Read() 函数返回的错误码：

n, \_ := f.Read(buf)

1

2

3

4

5

6

7

8

9

10

11

12

13

14

1.5匿名函数

匿名函数是指不需要定义函数名的一种函数实现方式，它并不是一个新概念,在Go里面，

函数可以像普通变量一样被传递或使用,Go语言支持随时在代码里定义匿名函数。

匿名函数由一个不带函数名的函数声明和函数体组成，如下所示：

1

2

3

4

func(a,b int,c float64) bool{

return a\*b < int(z)

}

1

2

3

匿名函数可以直接赋值给一个变量或者直接执行:

1

2

f := func(x,y int)int{

return x+y

}

func(ch chan int){

}

1

2

3

4

5

匿名函数花括号后直接跟参数列表表示函数调用,例如:

1

2

f := func(a,b int)int{

return a+b

}(1,2) //f最终的值为3

1

2

3

文单来源：http://blog.csdn.net/mchenys/article/details/50988015

版权声明：本文为博主原创文章，未经博主允许不得转载。