Go 语言函数方法 Go 语言数组

Go 语言变量作用域

作用域为已声明标识符所表示的常量、类型、变量、函数或包在源代码中的作用范围。

Go 语言中变量可以在三个地方声明：

- 函数内定义的变量称为局部变量
- 函数外定义的变量称为全局变量
- 函数定义中的变量称为形式参数

接下来让我们具体了解局部变量、全局变量和形式参数。

局部变量

在函数体内声明的变量称之为局部变量，它们的作用域只在函数体内，参数和返回值变量也是局部变量。

以下实例中 main() 函数使用了局部变量 a, b, c：

```
package main

import "fmt"

func main() {
   /* 声明局部变量 */
   var a, b, c int

   /* 初始化参数 */
   a = 10
   b = 20
   c = a + b

   fmt.Printf ("结果：  a = %d, b = %d and c = %d\n", a, b, c)
}
```

以上实例执行输出结果为：

```
结果：  a = 10, b = 20 and c = 30
```

全局变量

在函数体外声明的变量称之为全局变量，全局变量可以在整个包甚至外部包（被导出后）使用。

全局变量可以在任何函数中使用，以下实例演示了如何使用全局变量：

```
package main

import "fmt"

/* 声明全局变量 */
var g int

func main() {

   /* 声明局部变量 */
   var a, b int

   /* 初始化参数 */
   a = 10
   b = 20
   g = a + b

   fmt.Printf("结果:  a = %d, b = %d and g = %d\n", a, b, g)
}
```

以上实例执行输出结果为：

```
结果:  a = 10, b = 20 and g = 30
```

Go 语言程序中全局变量与局部变量名称可以相同，但是函数内的局部变量会被优先考虑。实例如下：

```
package main

import "fmt"

/* 声明全局变量 */
var g int = 20

func main() {
   /* 声明局部变量 */
   var g int = 10

   fmt.Printf ("结果:  g = %d\n",  g)
}
```

以上实例执行输出结果为：

```
结果:  g = 10
```

形式参数

形式参数会作为函数的局部变量来使用。实例如下：

```
package main

import "fmt"

/* 声明全局变量 */
var a int = 20;

func main() {
   /* main 函数中声明局部变量 */
   var a int = 10
   var b int = 20
   var c int = 0

   fmt.Printf("main()函数中 a = %d\n",  a);
   c = sum( a, b);
   fmt.Printf("main()函数中 c = %d\n",  c);
}

/* 函数定义-两数相加 */
func sum(a, b int) int {
   fmt.Printf("sum() 函数中 a = %d\n",  a);
   fmt.Printf("sum() 函数中 b = %d\n",  b);

   return a + b;
}
```

以上实例执行输出结果为：

```
main()函数中 a = 10
sum() 函数中 a = 10
sum() 函数中 b = 20
main()函数中 c = 30
```

初始化局部和全局变量

不同类型的局部和全局变量默认值为：

数据类型	初始化默认值
int	0
float32	0

数据类型	初始化默认值
pointer	nil

Go 语言函数方法 Go 语言数组

2 篇笔记

写笔记

形参使用，比较 sum 函数中的 a 和 main 函数中的 a，sum 函数中虽然加了 1，但是 main 中还是原值 10:

```
package main

import "fmt"

/* 声明全局变量 */
var a int = 20

func main() {
    /* main 函数中声明局部变量 */
    var a int = 10
    var b int = 20
    var c int = 0

    fmt.Printf("main()函数中 a = %d\n", a)
    c = sum(a, b)
    fmt.Printf("main()函数中 a = %d\n", a)
    fmt.Printf("main()函数中 c = %d\n", c)
}

/* 函数定义-两数相加 */
func sum(a, b int) int {
    a = a + 1
    fmt.Printf("sum() 函数中 a = %d\n", a)
    fmt.Printf("sum() 函数中 b = %d\n", b)
    return a + b
}
```

输出为：

```
main()函数中 a = 10
sum() 函数中 a = 11
sum() 函数中 b = 20
main()函数中 a = 10
main()函数中 c = 31
```

草原 9个月前 (06-25)

```
package main

import "fmt"

func main(){
  var a int = 0
  fmt.Println("for start")
  for a:=0; a < 10; a++ {
    fmt.Println(a)
  }
  fmt.Println("for end")

  fmt.Println(a)
}
```

输出为：

```
for start
0
1
2
3
4
5
6
7
8
9
for end
0
```

在 for 循环的 initialize（a:=0） 中，此时 initialize 中的 a 与外层的 a 不是同一个变量，initialize 中的 a 为 for 循环中的局部变量，因此在执行完 for 循环后，输出 a 的值仍然为 0。

package main import "fmt" func main(){ var a int = 0 fmt.Println("for start") for a = 0; a < 10; a++ { fmt.Println(a) } fmt.Println("for end") fmt.Println(a) }

输出为：

```
for start
0
1
2
3
4
5
6
7
8
9
```

```
for end
10
```

此时 initialize 中的 a 便于外层的 a 为同一个变量，因此在执行完 for 循环后，输出 a 的值为 10。

所以大伙们在使用 for 循环的时候千万要注意呀。。。

李克森 5个月前 (10-25)