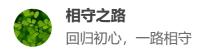
Golang学习——常量const和iota



Golang中常量const和iota

-.const

常量是一个简单值的标识符,在程序运行时,不会被修改的量。

Java编程规范中,常量一般都是全大写字母,但是在**Golang**中,大小写是具有一定含义的,所以不一定所有常量都得全大写

Golang中,大写字母开头表示 public ,小写字母开头表示 private

1.语法

```
const 常量名 [数据类型] = value
```

数据类型可以忽略不写, Golang编译器会自动推断出数据类型。

在使用时,要注意以下几点:

- 1. 数据类型只可以是布尔型、数字型(整数型、浮点型和复数)和字符串型
- 2. 满足多重赋值
- 3. 常量只定义不使用,编译不会报错
- 4. 常量可以作为枚举, 常量组
- 5. 常量组中如不指定类型和初始化值,则与上一行非空常量右值相同
- 6. 显示指定类型的时候,必须确保常量左右值类型一致,需要时可做显示类型转换。这与变量就不一样了,变量是可以是不同的类型值

2.实例

1.定义单个常量

```
const monday int = 1 // 显式类型定义
const tuesday = "2" // 隐式类型定义
const wednesday = 3 // 不使用 编译不会报错
```

```
fmt.Printf("monday 的数据类型为: %T, 值为: %d\n", monday, monday)
fmt.Printf("tuesday 的数据类型为: %T, 值为: %s\n", tuesday, tuesday)
```

输出为:

```
monday 的数据类型为: int, 值为: 1
tuesday 的数据类型为: string, 值为: 2
```

2.定义一组常量

Golang中没有枚举类型,不像Java或Python中有 Enum 类可以实现枚举,所以Golang通过 const 来实现枚举

```
const monday, tuesday, wednesday = 1, 2, 3
// 更推荐以下定义方式
const (
  one = 1
  two = 2
  three = 3
)
```

一组常量中,如果某个常量没有初始值,默认和上一行一致

```
const (
  one = 1
  two = 2
  three = 3
  four
  five
)
fmt.Printf("four 的数据类型为: %T, 值为: %d\n", four, four)
fmt.Printf("five 的数据类型为: %T, 值为: %d\n", five, five)
```

输出为:

```
four 的数据类型为: int, 值为: 3 five 的数据类型为: int, 值为: 3
```

写到这里,不知道有没有这样一个想法,如果我的常量很多,难道每一个常量都要——赋值吗?

二.iota

iota,特殊常量,可以认为是一个可以被编译器修改的常量

1.知识点

在使用 iota 时,需要注意以下几点:

- 1. 每当定义一个const, iota的初始值为0
- 2. 每当定义一个常量,就会自动累加1
- 3. 直到下一个const出现,清零
- 4. 如果中断iota自增,则必须显式恢复。且后续自增值按行序递增
- 5. 自增默认是int类型,可以自行进行显示指定类型
- 6. iota 可以参与运算

加粗的部分是需要重点关注的,现在实例演示一下

2.实例

1.定义const枚举, 首项为iota

```
const (
one = iota //iota初始值为0
two = iota //自增1
three = iota //再自增1
)
fmt.Println("one 的值为: ", one)
fmt.Println("two 的值为: ", two)
fmt.Println("three 的值为: ", three)
```

输出:

```
one 的值为: 0
two 的值为: 1
three 的值为: 2
```

可以看到,每一项都自增了1。 在定义时,不需要每次都写 iota ,结合 const 的特性, iota 除了首项之外,后续常量不显示写 iota ,都会自增

2.不显示写 iota , 常量也会自增

```
const (
  one = iota //iota初始值为0
  two
  three
  four
)
fmt.Println("one 的值为: ", one)
fmt.Println("two 的值为: ", two)
fmt.Println("three 的值为: ", three)
fmt.Println("four 的值为: ", four)
```

输出:

```
one 的值为: 0
two 的值为: 1
three 的值为: 2
four 的值为: 3
```

难道 iota 只能定义 0,1,2,3吗? 如果有别的值想用怎么办?

Golang中 iota 有一个非常非常好用的特性: iota 可以参与计算

3. iota 参与计算

比如,定义b,kb,mb...等等

输出:

```
1 1024 1048576 1073741824 1099511627776 1125899906842624
```

是不是非常好用? 虽然Golang中没有枚举类,但是通过 const 和 iota 也可以方便的实现枚举

三.结合 type, 更能表现实际意义

在实际开发中,枚举类通常都是有一定意义的,比如:月份,vip等级...

上面的例子中,只是简单的定义了数字常量,我们可以定义一个具体类型,来表示这些常量,实现 一个有意义的枚举

```
type VipLevel int
const (
  vipOne VipLevel = 1 + iota
  vipTwo
  vipThree
  vipFour
  vipFive
)
fmt.Println(one, two, three, four, five)
```

输出:

```
1 2 3 4 5
```

总结一下, const 和 iota 通常结合使用来定义枚举,为了使枚举更有意义,会自定义 type 类型。

要多注意 const 和 iota 的特性,在使用的时候留心下,熟能生巧,勤练习,勤思考。

编辑于 2020-05-04

Go 语言