Go 变量、常量、变量命名规则

主讲教师：（大地）
合作网站：www.itying.com （IT 营）
我的专栏：https://www.itying.com/category-79-b0.html

1、变量的来历

程序运行过程中的数据都是保存在内存中，我们想要在代码中操作某个数据时就需要去内存上找到这个变量，但是如果我们直接在代码中通过内存地址去操作变量的话，代码的可读性会非常差而且还容易出错，所以我们就利用变量将这个数据的内存地址保存起来，以后直接通过这个变量就能找到内存上对应的数据了。

2、变量类型

变量（Variable）的功能是存储数据。不同的变量保存的数据类型可能会不一样。经过半个多世纪的发展，编程语言已经基本形成了一套固定的类型，常见变量的数据类型有：整型、浮点型、布尔型等。

Go 语言中的每一个变量都有自己的类型，并且变量必须经过声明才能开始使用。

3、GO 语言中变量的声明

Go 语言变量名由字母、数字、下划线组成，其中首个字符不能为数字。Go 语言中关键字和保留字都不能用作变量名。

Go 语言中的变量需要声明后才能使用，同一作用域内不支持重复声明。 并且 Go 语言的变量声明后必须使用。

1、var 声明变量

```
var  变量名称  type
```

```
var name string

var age int

var isOk bool
```

```
package main

import "fmt";

func main()   {

      var username="张三"

      var age int =20

      fmt.Println(username,age)

}
```

2、一次定义多个变量

```
var identifier1, identifier2 type
```

```
package main

import "fmt"

func main() {

      var username, sex string
```

```
    username = "张三"
    sex = "男"
    fmt.Println(username, sex)
}
申明变量的时候赋值
 var a, b, c, d = 1, 2, 3, false
```

3、批量声明变量的时候指定类型

```
var (
    a string
    b int
    c bool
)
a = "张三"
b = 10
c = true
fmt.Println(a,b,c)
```

批量声明变量并赋值

```
var (
        a string = "张三"
        b int    = 20
        c bool   = true
)
fmt.Println(a, b, c)
fmt.Println(a,b,c)
```

4、变量的初始化

Go 语言在声明变量的时候，会自动对变量对应的内存区域进行初始化操作。每个变量会被初始化成其类型的默认值，例如： 整型和浮点型变量的默认值为 0。 字符串变量的默认值为空字符串。 布尔型变量默认为 false。 切片、函数、指针变量的默认为 nil。

当然我们也可在声明变量的时候为其指定初始值。变量初始化的标准格式如下：

```
var 变量名 类型 = 表达式
```

举个例子：

```
var name string = "zhangsan"

var age int = 18
```

或者一次初始化多个变量并赋值

```
var name, age = "zhangsan", 20
```

5、类型推导

有时候我们会将变量的类型省略，这个时候编译器会根据等号右边的值来推导变量的类型完成初始化。

```
var name = "Q1mi"

var age = 18
```

6、短变量声明法

在函数内部，可以使用更简略的 := 方式声明并初始化变量。

注意：短变量只能用于声明局部变量，不能用于全局变量的声明

```
变量名 := 表达式
```

```
package main

import (
	"fmt"
)
// 全局变量 m
var m = 100


func main() {
	n := 10
	m := 200 // 此处声明局部变量 m
	fmt.Println(m, n)
}
```

使用变量一次声明多个变量，并初始化变量

```
m1, m2, m3 := 10, 20, 30
fmt.Println(m1, m2, m3)
```

7、匿名变量

在使用多重赋值时，如果想要忽略某个值，可以使用**匿名变量**（anonymous variable）。 匿名变量用一个下划线_表示，例如：

```
func getInfo() (int, string) {
	return 10, "张三"
}
```

```
func main() {
	_, username := getInfo()
	fmt.Println(username)
}
```

匿名变量不占用命名空间，不会分配内存，所以匿名变量之间不存在重复声明。

注意事项：

1、函数外的每个语句都必须以关键字开始（var、const、func 等）

2、:=不能使用在函数外。

3、_多用于占位，表示忽略值。

4、Go 语言中的常量

相对于变量，常量是恒定不变的值，多用于定义程序运行期间不会改变的那些值。 常量的声明和变量声明非常类似，只是把 var 换成了 const，常量在定义的时候必须赋值。

1、使用 const 定义常量

```
const pi = 3.1415
const e = 2.7182
```

声明了 pi 和 e 这两个常量之后，在整个程序运行期间它们的值都不能再发生变化了。

多个常量也可以一起声明：

```
const (
	pi = 3.1415
	e = 2.7182
)
```

const 同时声明多个常量时，如果省略了值则表示和上面一行的值相同。 例如：

```
const (
	n1 = 100
	n2
	n3
)
```

上面示例中，常量 n1、n2、n3 的值都是 100。

打印 Pi 的值

```
package main
import (
	"fmt"
	"math"
)
func main() {
	const pi=math.Pi
	fmt.Println(pi);
}
```

2、const 常量结合 iota 的使用（了解）

iota 是 golang 语言的常量计数器,只能在常量的表达式中使用。

iota 在 const 关键字出现时将被重置为 0(const 内部的第一行之前)，const 中每新增一行常量声明将使 iota 计数一次(iota 可理解为 const 语句块中的行索引)。

1、iota 只能在常量的表达式中使用。

```
fmt.Println(iota)

编译错误：  undefined: iota
```

2、每次 const 出现时，都会让 iota 初始化为 0.【自增长】

```
const a = iota // a=0

const (

  b = iota        //b=0

  c                 //c=1

)
```

3、const iota 使用_跳过某些值

```
const (

            n1 = iota //0

            n2              //1

            _

            n4              //3

)
```

4、iota 声明中间插队

```
const (
        n1 = iota //0
        n2 = 100  //100
        n3 = iota //2
        n4        //3
    )
const n5 = iota //0
```

4、多个 iota 定义在一行

```
const (
        a, b = iota + 1, iota + 2 //1,2
        c, d                      //2,3
        e, f                      //3,4
)
```

5、Go 语言变量、常量命名规则

1、变量名称必须由数字、字母、下划线组成。

2、标识符开头不能是数字

3、标识符不能是保留字和关键字。

4、变量的名字是区分大小写的如: age 和 Age 是不同的变量。在实际的运用中,也建议,不要用一个单词大小写区分两个变量。

5、标识符(变量名称)一定要见名思意 ：变量名称建议用名词，方法名称建议用动词

6、变量命名一般采用驼峰式，当遇到特有名词（缩写或简称，如 DNS）的时候，特有名词根据是否私有全部大写或小写。

6、Go 语言代码风格

1、代码每一行结束后不用写分号（ ;）

2、运算符左右建议各加一个空格

```
var    username string = "itying"
```

3、Go 语言程序员推荐使用驼峰式命名

当名字有几个单词组成的时优先使用大小写分隔

4、强制的代码风格

左括号必须紧接着语句不换行，这个特性刚开始会使开发者不习惯，但随着对 Go 语言的不断熟悉，就会发现风格统一让大家在阅读代码时把注意力集中到了解决问题上，而不是代码风格上

5、go fmt 主要用于格式化文档，让所有人的代码风格保持一致

```
D:\golang\src\demo01>go fmt main.go

main.go
```