@[toc]

# Go 前言

## 参考资料

[go中文社区](https://studygolang.com)  
[go中文在线文档](https://studygolang.com/pkgdoc)

## 注意事项

* Go语言严格区分大小写。
* Go方法由一条条语句构成，每个语句后不需要分号Go语言会在每行后自动加分号），这也体现出 Golan的简洁性。
* Go编译器是一行行进行编译的，因此我们一行就写一条语句，**不能把多条语句写在同一个**，否则报错
* go语言**定义的变量**或者 **import的包**如果没有使用到，代码不能编译通过。
* 每个Go源代码文件的开头都是一个package声明，表示该Go代码所属的包。包是Go语言里最基本的分发单位，也是工程管理中依赖关系的体现。
* 要生成Go可执行程序，必须建立一个名字为main的包，并且在该包中包含一个叫main()的函数（该函数是Go可执行程序的执行起点）。

Tips ： 通常以新建文件夹的形式新建文件以避免mian重复问题

* Go语言的main()函数不能带参数，也不能定义返回值。
* 在包声明之后，是一系列的import语句，用于导入该程序所依赖的包。由于本示例程序用到了Println()函数，所以需要导入该函数所属的fmt包。

# Go 语法

## 命名与关键字

函数名、变量名、常量名、类型名、语句标号和包名等所有的命名，都遵循一个简单的命名**规则**：

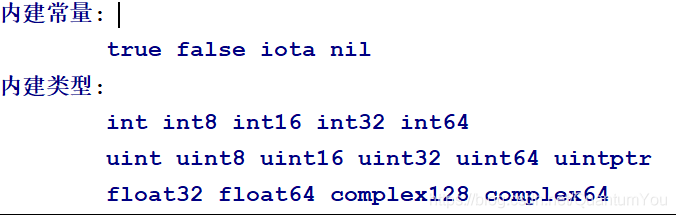
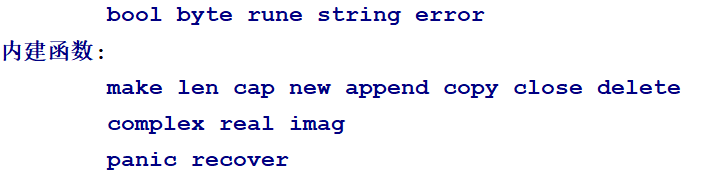
* 一个名字必须以一个字母（Unicode字母）或下划线开头，后面可以跟任意数量的字母、数字或下划线。
* 大写字母和小写字母是不同的：heapSort和Heapsort是两个不同的名字。

**关键字**

按字母顺序排序为：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| break | default | func | interface | select |
| case | defer | go | map | struct |
| chan | else | goto | package | switch |
| const | fallthrough | if | range | type |
| continue | for | import | return | var |

此外，还有大约30多个预定义的名字，比如int和true等，主要对应内建的常量、类型和函数。

## 基础类型

### 变量

一次定义多个变量

注意: 变量与数据类型的位置，与C语言不同

var v3, v4 int   
  
 var (  
 v5 int  
 v6 int  
 )

注意 ： 多个变量的赋值，为 （ ）

#### 变量初始化与自动推导类型

var v1 int = 10 // 方式1  
 var v2 = 10 // 方式2，编译器自动推导出v2的类型  
 v3 := 10 // 方式3，编译器自动推导出v3的类型  
 fmt.Println("v3 type is ", reflect.TypeOf(v3)) //v3 type is int

出现在 := 左侧的变量不应该是已经被声明过，:=定义时必须初始化

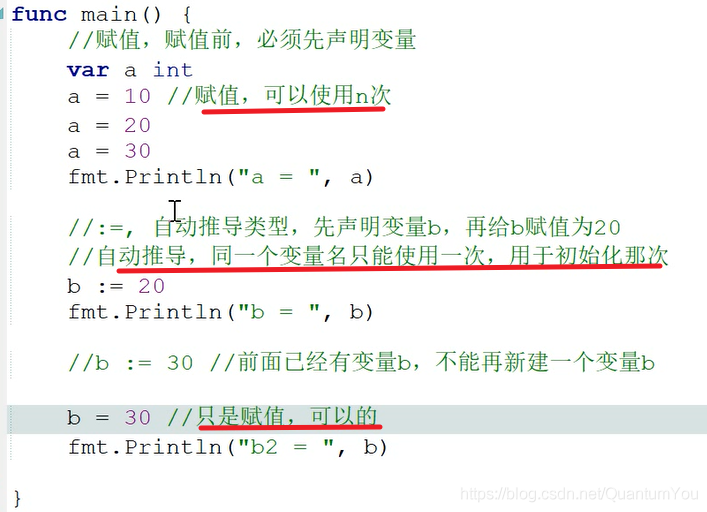
var v4 int  
 v4 := 2 //err

#### 变量赋值

var v1 int  
 v1 = 123  
  
 var v2, v3, v4 int  
 v2, v3, v4 = 1, 2, 3 //多重赋值

i := 10  
 j := 20  
 i, j = j, i //多重赋值与交换

注意：

1、可以多次赋值，但是不能多次类型推导  
2、 i, j = j, i 变量的交换，优化传统需要第三方变量（temp）的缺点  


#### 匿名变量

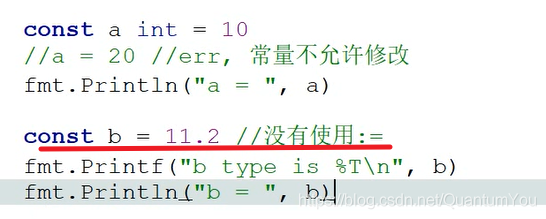
\_（下划线）是个特殊的变量名，任何赋予它的值都会被丢弃：

\_, i, \_, j := 1, 2, 3, 4  
  
func test() (int, string) {  
 return 250, "sb"  
}  
  
\_, str := test()

注意： 通常与函数返回值相使用，因为Go 函数可以传多值

### 常量

* 变量：程序运行期间，可以改变的量，变量声明需要var
* 常量：程序运行期间，不可以改变的量，常量声明需要 const

  
注意： 常量不使用类型推导，因为其值固定

### iota 枚举类型

* 常量声明可以使用iota常量生成器初始化，它用于生成一组以相似规则初始化的常量，但是不用每行都写一遍初始化表达式。
* 在一个const声明语句中，在第一个声明的常量所在的行，iota将会被置为0，然后在每一个有常量声明的行加一。

const (  
 x = iota // x == 0  
 y = iota // y == 1  
 z = iota // z == 2  
 w // 这里隐式地说w = iota，因此w == 3。其实上面y和z可同样不用"= iota"  
 )  
  
 const v = iota // 每遇到一个const关键字，iota就会重置，此时v == 0  
  
 const (  
 h, i, j = iota, iota, iota //h=0,i=0,j=0 iota在同一行值相同  
 )  
  
 const (  
 a = iota //a=0  
 b = "B"  
 c = iota //c=2  
 d, e, f = iota, iota, iota //d=3,e=3,f=3  
 g = iota //g = 4  
 )  
   
 const (  
 x1 = iota \* 10 // x1 == 0  
 y1 = iota \* 10 // y1 == 10  
 z1 = iota \* 10 // z1 == 20  
 )

## 数据类型

Go 内置如下基础数据类型

  
注意： 零值表示的未赋值时的值（初始值）

### 布尔类型

var v1 bool  
v1 = true  
v2 := (1 == 2) // v2也会被推导为bool类型  
  
//布尔类型不能接受其他类型的赋值，不支持自动或强制的类型转换  
var b bool  
b = 1 // err, 编译错误  
b = bool(1) // err, 编译错误

### 浮点类型

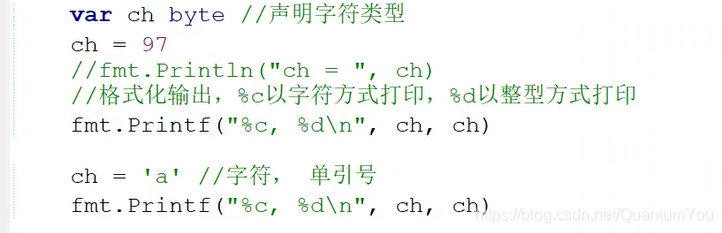
var f1 float32  
 f1 = 12  
 f2 := 12.0 // 如果不加小数点， fvalue2会被推导为整型而不是浮点型，float64

### 字符类型

* 在Go语言中支持两个字符类型，一个是byte（实际上是uint8的别名），代表utf-8字符串的单个字节的值；另一个是rune，代表单个unicode字符。（包含中文）

package main  
  
import (  
 "fmt"  
)  
  
func main() {  
 var ch1, ch2, ch3 byte  
 ch1 = 'a' //字符赋值  
 ch2 = 97 //字符的ascii码赋值  
 ch3 = '\n' //转义字符  
 fmt.Printf("ch1 = %c, ch2 = %c, %c", ch1, ch2, ch3)

注意： 输出行的转义字符的书写

实例：  
  
输出

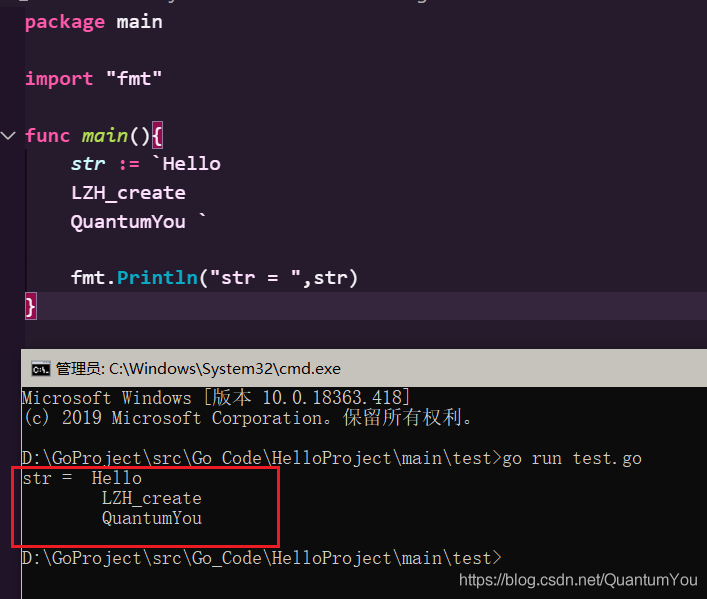
a,97  
a,97

### 字符串类型

var str string // 声明一个字符串变量  
 str = "abc" // 字符串赋值  
 ch := str[0] // 取字符串的第一个字符  
 fmt.Printf("str = %s, len = %d\n", str, len(str)) //内置的函数len()来取字符串的长度  
 fmt.Printf("str[0] = %c, ch = %c\n", str[0], ch)

注意： 内置的函数len()来取字符串的长度

`(反引号)括起的字符串为Raw字符串，即字符串在代码中的形式就是打印时的形式，它没有字符转义，换行也将原样输出。



### 复数类型

* 复数实际上由两个实数（在计算机中用浮点数表示）构成，一个表示实部（real），一个表示虚部（imag）。

var v1 complex64 // 由2个float32构成的复数类型  
 v1 = 3.2 + 12i  
 v2 := 3.2 + 12i // v2是complex128类型  
 v3 := complex(3.2, 12) // v3结果同v2  
  
 fmt.Println(v1, v2, v3)  
 //内置函数real(v1)获得该复数的实部  
 //通过imag(v1)获得该复数的虚部  
 fmt.Println(real(v1), imag(v1))

## 类型转换与别名

### 类型转换

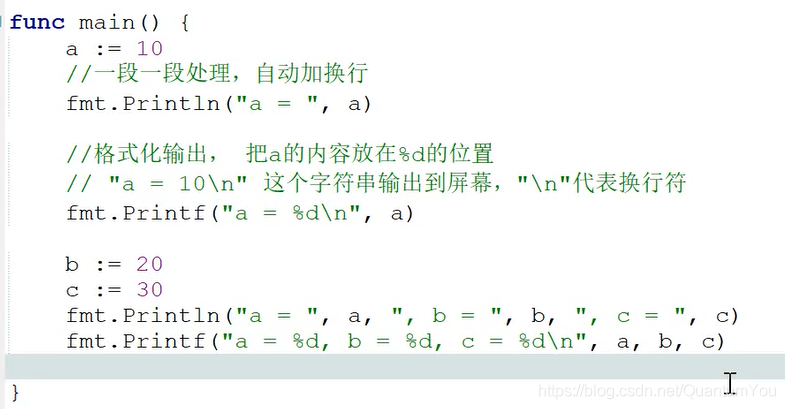
Go语言中不允许隐式转换，所有类型转换必须显式声明，而且转换只能发生在两种相互兼容的类型之间。



### 类型别名

    type bigint int64 //int64类型改名为bigint  
    var x bigint = 100  
  
    type (  
        myint int //int改名为myint  
        mystr string //string改名为mystr  
    )

## 输出格式

Println 与 Printf 的区别  


### 占位符

#### 普通占位符



#### 布尔占位符



#### 整数占位符



#### 浮点数与复数



#### 字符串与字节切片



#### 指针



### 转义字符

1、\t：表示一个制表符，通常使用它可以排版。  
2、\n：换行符  
3、\\ 一个 \  
4、\" 一个 "  
5、\r 换行但是会覆盖本行

fmt.Println("LZH\_Create\rQJY")

输出

QJY\_Create

## 输入语句

var v int  
 fmt.Println("请输入一个整型：")  
 fmt.Scanf("%d", &v)  
 //fmt.Scan(&v)  
 fmt.Println("v = ", v)