





如何判断整型的类型

fmt.Printf()可以用来做格式化输出

var n1 int =100

fmt.Printf(“n1的类型%T”，n1)

unsafe.Sizeof(n1)//返回n1变量的字节数

常用占位符

**定义示例类型和变量**

type Human struct {  
Name string  
}

var people = Human{Name:"zhangsan"}

|  |  |  |  |
| --- | --- | --- | --- |
| %v | 相应值的默认格式。 | Printf("%v", people) | {zhangsan} |
| %T | 类型 | Printf("%T", people) | main.Human |
| %t | true 或 false | Printf("%t", true) | true |
| %d | 十进制表示 | Printf("%d", 0x12) | 18 |
| %f | 有小数点而无指数，例如 123.456 | Printf("%f", 10.2) | 10.200000 |
| %p | 十六进制表示，前缀 0x | Printf("%p", &people) | 0x4f57f0 |
| %c | 输出单个字符 |  |  |

a.Go中没有专门的字符类型，如果要存储单个字符（字母），一般使用byte来保存。而byte就是单字符int

b.字符串就是一串固定长度的字符连接起来的字符序列，go的字符串是由单个字节连接起来的，也就是

说对于传统的字符串是由字符组成的，而go的字符串不同，它是由字节组成的

c.字符常量是用单引号括起来的单个字符，例如：var c1 byte=’a’ var c2 int=”中” var c3 byte=’9’

d.go语言是由utf-8编码

布尔类型只能是true和false

Go的字符串是常量 所以不能修改

``可以连接多行字符串

`eee

ee

`



字符类型转换

类型 B 的值 = 类型 B(类型 A 的值)

var str string="tancanghai"

    for index,val :=range str{

        fmt.Printf("%v %c \n",index,val)

    }

Go语言是utf-8的编码 ，汉子占有几个字节

基本语法：

Func 函数名（形参列表）（返回值列表）{

执行语句。。。。。。

return 返回值列表

}

\_init()初始话函数 main()函数 和全局变量 先执行全局变量-》初始话函数 –》main()函数；

引用传递因为只是传递地址 所以比值传递效率高

按照Go语言规范，任何类型在未初始化时都对应一个零值：布尔类型是false，整型是0，字符串是""，而指针、函数、interface、slice、channel和map的零值都是nil。

PS：这里没有说结构体struct的零值为nil，因为struct的零值与其属性有关

nil没有默认的类型，尽管它是多个类型的零值，必须显式或隐式指定每个nil用法的明确类型。