基本数据类型

整型

整型分为以下两个大类： 按长度分为：int8、int16、int32、int64 对应的无符号整型：uint8、uint16、uint32、uint64

其中， `uint8` 就是我们熟知的 `byte` 型， `int16` 对应C语言中的 `short` 型， `int64` 对应C语言中的 `long` 型。

类型	描述
uint8	无符号 8位整型 (0 到 255)
uint16	无符号 16位整型 (0 到 65535)
uint32	无符号 32位整型 (0 到 4294967295)
uint64	无符号 64位整型 (0 到 18446744073709551615)
int8	有符号 8位整型 (-128 到 127)
int16	有符号 16位整型 (-32768 到 32767)
int32	有符号 32位整型 (-2147483648 到 2147483647)
int64	有符号 64位整型 (-9223372036854775808 到 9223372036854775807)

特殊整型

类型	描述
uint	32位操作系统上就是uint32，64位操作系统上就是uint64
int	32位操作系统上就是int32，64位操作系统上就是int64
uintptr	无符号整型，用于存放一个指针

注意： 在使用 `int` 和 `uint` 类型时，不能假定它是32位或64位的整型，而是考虑 `int` 和 `uint` 可能在不同平台上的差异。

注意事项 获取对象的长度的内建 `len()` 函数返回的长度可以根据不同平台的字节长度进行变化。实际使用中，切片或 map 的元素数量等都可以用 `int` 来表示。在涉及到二进制传输、读写文件的结构描述时，为了保持文件的结构不会受到不同编译目标平台字节长度的影响，不要使用 `int` 和 `uint` 。

八进制&十六进制

Go语言中无法直接定义二进制数，关于八进制和十六进制数的示例如下：

```go
package main

import "fmt"

func main(){
    // 十进制
    var a int = 10
    fmt.Printf("%d \n", a)  // 10
    fmt.Printf("%b \n", a)  // 1010  占位符%b表示二进制

    // 八进制  以0开头
    var b int = 077
    fmt.Printf("%o \n", b)  // 77

    // 十六进制  以0x开头
    var c int = 0xff
    fmt.Printf("%x \n", c)  // ff
    fmt.Printf("%X \n", c)  // FF
}
```

```go
func main() {
    var a int = 10
    fmt.Printf("%d\n", a)
    fmt.Printf("%b\n", a)
    //八进制
    var b int = 077
    fmt.Printf("%o\n", b)
    //十六进制
    var c int = 0xff
    fmt.Printf("%x\n", c)
    fmt.Printf("%X\n", c)
}
```

复数

complex64和complex128

```go
var c1 complex64
c1 = 1 + 2i
var c2 complex128
c2 = 2 + 3i
fmt.Println(c1)
fmt.Println(c2)
```

复数有实部和虚部，complex64的实部和虚部为32位，complex128的实部和虚部为64位。

布尔值

Go语言中以 `bool` 类型进行声明布尔型数据，布尔型数据只有 `true`（真） 和 `false`（假） 两个值。

注意：

1. 布尔类型变量的默认值为false。
2. Go 语言中不允许将整型强制转换为布尔型.
3. 布尔型无法参与数值运算，也无法与其他类型进行转换。