Go语言中的格式化输出

通用格式化输出

在Go语言中格式化输出通常使用 fmt 包，通用的输出格式如下表所示：

输出格式	输出内容
%v	值的默认格式表示
%+v	类似 %v，但输出结构体时会添加字段名
%#v	值的 Go 语法表示
%T	值的类型的 Go 语法表示

```
package main

import "fmt"

func main() {
    var s string = "ajwlf"
    var z int = 1

    fmt.Printf("s: %v\n", s)
    fmt.Printf("s: %+v\n", s)
    fmt.Printf("s: %#v\n", s)
    fmt.Printf("s: %T\n", s)

    fmt.Printf("z: %v\n", z)
    fmt.Printf("z: %T\n", z)
}

#结果
s: ajwlf
s: ajwlf
s: "ajwlf"
s: string
z: 1
z: int
```

布尔类型

输出格式	输出内容
%t	单词 true 或 false

```
package main
import "fmt"
func main() {
    var flag bool
    fmt.Printf("%T, %t \n", flag, flag)
    flag = true
    fmt.Printf("%T, %t \n", flag, flag)
}

# 结果
bool, false
bool, true
```

整数类型

输出格式	输出内容
%b	表示为二进制
%c	该值对应的 unicode 码值
%d	表示为十进制
%8d	表示该整型长度是 8，不足 8 则在数值前补空格；如果超出 8，则以实际为准
%08d	表示该整型长度是 8，不足 8 则在数值前补 0；如果超出 8，则以实际为准
%o	表示为八进制
%q	该值对应的单引号括起来的Go语言语法字符字面值，必要时会采用安全的转义表示
%x	表示为十六进制，使用 a~f
%X	表示为十六进制，使用 A~F
%U	表示为 unicode 格式：U+1234，等价于 U+%04X

```
package main
import "fmt"
func main() {
    fmt.Printf("%T, %d \n", 123, 123)
    fmt.Printf("%T, %5d \n", 123, 123)
    fmt.Printf("%T, %05d \n", 123, 123)
    fmt.Printf("%T, %b \n", 123, 123)
    fmt.Printf("%T, %o \n", 123, 123)
    fmt.Printf("%T, %c \n", 97, 97)
    fmt.Printf("%T, %q \n", 97, 97)
    fmt.Printf("%T, %x \n", 123, 123)
    fmt.Printf("%T, %X \n", 123, 123)
    fmt.Printf("%T, %U \n", '一', '一')
}

# 结果
int, 123
int,   123
```

```
int, 00123
int, 1111011
int, 173
int, a
int, 'a'
int, 7b
int, 7B
int32, U+4E00
```

浮点型与复数型

输出格式	输出内容
%b	无小数部分、二进制指数的科学计数法，如 -123456p-78
%e	（=%.6e）有 6 位小数部分的科学计数法，如 -1234.456e+78
%E	科学计数法，如 -1234.456E+78
%f	（=%.6f）有 6 位小数部分，如 123.456123
%F	等价于 %f
%g	根据实际情况采用 %e 或 %f 格式（获得更简洁、准确的输出）
%G	根据实际情况采用 %E 或 %F 格式（获得更简洁、准确的输出）

```
package main
import "fmt"
func main() {
    fmt.Printf("%b \n", 123.123456)
    fmt.Printf("%f \n", 123.1)
    fmt.Printf("%.2f \n", 123.125456)
    fmt.Printf("%e \n", 123.123456)
    fmt.Printf("%E \n", 123.123456)
    fmt.Printf("%.1e \n", 123.123456)
    fmt.Printf("%F \n", 123.123456)
    fmt.Printf("%g \n", 123.123456)
    fmt.Printf("%G \n", 123.123456)
}

# 结果
8664042977533870p-46
123.100000
123.13
1.231235e+02
1.231235E+02
1.2e+02
123.123456
123.123456
123.123456
```

字符串与字节数组

输出格式	输出内容
%s	直接输出字符串或者字节数组
%q	该值对应的双引号括起来的Go语法字符串字面值，必要时会采用安全的转义表示
%x	每个字节用两字符十六进制数表示，使用 a~f
%X	每个字节用两字符十六进制数表示，使用 A~F

```
package main
import "fmt"
func main() {
    arr := []byte{'x', 'y', 'z', 'z'}
    fmt.Printf("%s \n", "欢迎访问")
    fmt.Printf("%q \n", "欢迎访问")
    fmt.Printf("%x \n", "欢迎访问")
    fmt.Printf("%X \n", "欢迎访问")
    fmt.Printf("%T, %s \n", arr, arr)
    fmt.Printf("%T, %q \n", arr, arr)
    fmt.Printf("%T, %x \n", arr, arr)
    fmt.Printf("%T, %X \n", arr, arr)
}

#结果
欢迎访问
"欢迎访问"
e6aca2e8bf8ee8aebfe997aee5beaee5ada6e88b91
E6ACA2E8BF8EE8AEBFE997AEE5BEAEE5ADA6E88B91
[]uint8, xyzz
[]uint8, "xyzz"
[]uint8, 78797a7a
[]uint8, 78797A7A
```

指针类型

输出格式	输出内容
%p	指针类型

```
package main

import (
    "fmt"
)

func main() {
    x := 100
    p := &x
    fmt.Printf("p: %T,%p\n", p, p)
}
#结果
p: *int,0xc0000a6058
```