1、基本介绍

Go语言中strconv包实现了`基本数据类型和其字符串表示`的相互转换。
strconv包实现了基本数据类型与其字符串表示的转换，主要有以下常用函数：`Atoi()、Itoa()、parse系列、format系列、append系列`。

- func IsPrint(r rune) bool
- func CanBackquote(s string) bool
- func Quote(s string) string
- func QuoteToASCII(s string) string
- func QuoteRune(r rune) string
- func QuoteRuneToASCII(r rune) string
- func Unquote(s string) (t string, err error)
- func UnquoteChar(s string, quote byte) (value rune, multibyte bool, tail string, err error)
- func ParseBool(str string) (value bool, err error)
- func ParseInt(s string, base int, bitSize int) (i int64, err error)
- func ParseUint(s string, base int, bitSize int) (n uint64, err error)
- func ParseFloat(s string, bitSize int) (f float64, err error)
- func FormatBool(b bool) string
- func FormatInt(i int64, base int) string
- func FormatUint(i uint64, base int) string
- func FormatFloat(f float64, fmt byte, prec, bitSize int) string
- func Atoi(s string) (i int, err error)
- func Itoa(i int) string
- func AppendBool(dst []byte, b bool) []byte
- func AppendInt(dst []byte, i int64, base int) []byte
- func AppendUint(dst []byte, i uint64, base int) []byte
- func AppendFloat(dst []byte, f float64, fmt byte, prec int, bitSize int) []byte
- func AppendQuote(dst []byte, s string) []byte
- func AppendQuoteToASCII(dst []byte, s string) []byte
- func AppendQuoteRune(dst []byte, r rune) []byte
- func AppendQuoteRuneToASCII(dst []byte, r rune) []byte

2、Atoi和Itoa函数

func Atoi

```
func Atoi(s string) (i int, err error)
```

Atoi是ParseInt(s, 10, 0)的简写。

func Itoa

```
func Itoa(i int) string
```

Itoa是FormatInt(i, 10) 的简写。


```go
func testAtoi2() {

    // 测试Atoi，字符串转整数
    num1, err := strconv.Atoi("123.123")
    if err != nil {
        fmt.Println("转换错误 --> ", err)
    }
    fmt.Println("num1: ", num1)

    // 测试Itoa，整数转字符串
    str := strconv.Itoa(190)
    fmt.Println("str --> ", str)
}

func main() {

    testAtoi2()
}
```

```
问题 1  输出  调试控制台  终端  COMMENTS

PS F:\tools\golang\strconv> go run .\main.go
转换错误 -->  strconv.Atoi: parsing "123.123": invalid syntax
num1:  0
str -->  190
PS F:\tools\golang\strconv>
```

3、Parse系列

Parse类函数用于转换字符串为给定类型的值：`ParseBool()、ParseFloat()、ParseInt()、ParseUint()`。

func ParseBool

```
func ParseBool(str string) (value bool, err error)
```

返回字符串表示的bool值。它接受1、0、t、f、T、F、true、false、True、False、TRUE、FALSE；否则返回错误。

func ParseInt

```
func ParseInt(s string, base int, bitSize int) (i int64, err error)
```

返回字符串表示的整数值，接受正负号。

base指定进制（2到36），如果base为0，则会从字符串前置判断，"0x"是16进制，"0"是8进制，否则是10进制；

bitSize指定结果必须能无溢出赋值的整数类型，0、8、16、32、64 分别代表 int、int8、int16、int32、int64；返回的err是*NumErr类型的，如果语法有误，err.Error = ErrSyntax；如果结果超出类型范围err.Error = ErrRange。

func ParseUint

```
func ParseUint(s string, base int, bitSize int) (n uint64, err error)
```

ParseUint类似ParseInt但不接受正负号，用于无符号整型。

func ParseFloat

```
func ParseFloat(s string, bitSize int) (f float64, err error)
```

解析一个表示浮点数的字符串并返回其值。

如果s合乎语法规则，函数会返回最为接近s表示值的一个浮点数（使用IEEE754规范舍入）。bitSize指定了期望的接收类型，32是float32（返回值可以不改变精确值的赋值给float32），64是float64；返回值err是*NumErr类型的，语法有误的，err.Error=ErrSyntax；结果超出表示范围的，返回值f为±Inf，err.Error= ErrRange。

4、Format系列

func FormatBool

```
func FormatBool(b bool) string
```

根据b的值返回"true"或"false"。

func FormatInt

```
func FormatInt(i int64, base int) string
```

返回i的base进制的字符串表示。base 必须在2到36之间，结果中会使用小写字母'a'到'z'表示大于10的数字。

func FormatUint

```
func FormatUint(i uint64, base int) string
```

是FormatInt的无符号整数版本。

func FormatFloat

```
func FormatFloat(f float64, fmt byte, prec, bitSize int) string
```

函数将浮点数表示为字符串并返回。

bitSize表示f的来源类型（32：float32、64：float64），会据此进行舍入。

fmt表示格式：'f'（-ddd.dddd）、'b'（-ddddp±ddd，指数为二进制）、'e'（-d.ddde±dd，十进制指数）、'E'（-d.ddddE±dd，十进制指数）、'g'（指数很大时用'e'格式，否则'f'格式）、'G'（指数很大时用'E'格式，否则'f'格式）。

prec控制精度（排除指数部分）：对'f'、'e'、'E'，它表示小数点后的数字个数；对'g'、'G'，它控制总的数字个数。如果prec 为-1，则代表使用最少数量的、但又必需的数字来表示f。