Go基本类型

- 布尔型：bool
    - 长度：1字节
    - 取值范围：true, false
    - 注意事项：不可以用数字代表true或false
- 整型：int/uint
    - 根据运行平台可能为32或64位
- 8位整型：int8/uint8
    - 长度：1字节
    - 取值范围：-128~127/0~255
- 字节型：byte（uint8别名）

Go基本类型

- 16位整型：int16/uint16
    - 长度：2字节
    - 取值范围：-32768~32767/0~65535
- 32位整型：int32（rune）/uint32
    - 长度：4字节
    - 取值范围：-2^32/2~2^32/2-1/0~2^32-1
- 64位整型：int64/uint64
    - 长度：8字节
    - 取值范围：-2^64/2~2^64/2-1/0~2^64-1
- 浮点型：float32/float64
    - 长度：4/8字节
    - 小数位：精确到7/15小数位

## Go基本类型

- 复数：complex64/complex128
  - 长度：8/16字节
- 足够保存指针的 32 位或 64 位整数型：uintptr

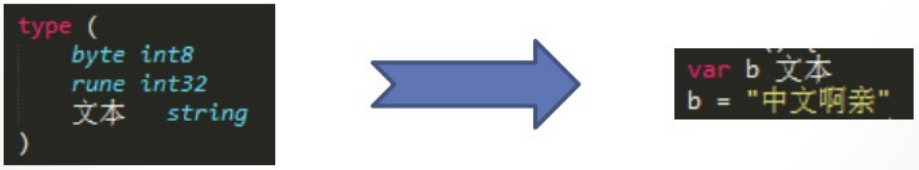
- 其它值类型：
  - array、struct、string
- 引用类型：
  - slice、map、chan

- 接口类型：inteface
- 函数类型：func

## 类型零值

零值并不等于空值，而是当变量被声明为某种类型后的默认值，通常情况下值类型的默认值为0，bool为false，string为空字符串

## 类型别名




```go
// 当前程序的包名
package main

// 导入其它的包
import (
    "fmt"
)

func main() {
    var a int
    var b int32
    var c float32
    var d float64
    var e bool
    var f string
```

```
    var g [1]int
    var h [1]bool
    var i [1]byte

    fmt.Println(a)
    fmt.Println(b)
    fmt.Println(c)
    fmt.Println(d)
    fmt.Println(e)
    fmt.Println(f)
    fmt.Println(g)
    fmt.Println(h)
    fmt.Println(i)
}
```

```
// 当前程序的包名
package main

// 导入其它的包
import (
    "fmt"
    "math"
)

func main() {
    fmt.Println(math.MinInt8)
}
```

```
// 当前程序的包名
package main

// 导入其它的包
import (
    "fmt"
)

type (
    byte int8
    rune int32
    文本   string
)

func main() {
    var b 文本
    b = "中文类型名"
    fmt.Println(b)
}
```

```
成功: 进程退出代码 0.
/home/jiemin/code/GOlang/go/src/basic_structure/basic_structure  [/home/jiemin/code/GOlang/go/src/basic_structure]
中文类型名
成功: 进程退出代码 0.
```

```go
// 当前程序的包名
package main

// 导入其它的包
import (
    "fmt"
)

type (
    byte     int8
    rune     int32
    ByteSize int64
)

func main() {
    var b ByteSize
    b = 213213242412412
    fmt.Println(b)
}
```

```
/home/jiemin/code/GOlang/go/src/basic_structure/basic_structure  [/home/jiemin/code/GOlang/go/src/basic_structure]
213213242412412
成功: 进程退出代码 0.
```

## 单个变量的声明与赋值

- 变量的声明格式：var <变量名称> <变量类型>
- 变量的赋值格式：<变量名称> = <表达式>
- 声明的同时赋值：var <变量名称> [变量类型] = <表达式>

```go
var a int // 变量的声明
a = 123   // 变量的赋值

// 变量声明的同时赋值
var b int = 321
// 上行的格式可省略变量类型，由系统推断
var c = 321
// 变量声明与赋值的最简写法
d := 456
```

```go
// 当前程序的包名
package main

```

```go
// 导入其它的包
import (
    "fmt"
)

type (
    byte     int8
    rune     int32
    ByteSize int64
)

func main() {
    var b int
    b = 1
    fmt.Println(b)
}
```

```go
// 当前程序的包名
package main

// 导入其它的包
import (
    "fmt"
)

type (
    byte     int8
    rune     int32
    ByteSize int64
)

func main() {
    var b int = 1
    fmt.Println(b)
}
```

```go
// 当前程序的包名
package main

// 导入其它的包
import (
    "fmt"
)

type (
    byte     int8
    rune     int32
    ByteSize int64
)

func main() {
    var b = 1
    fmt.Println(b)
```

```
18. }
```

执行结果全部为：

```
/home/jiemin/code/GOlang/go/src/basic_structure/basic_structure [/home/jiemin/code/GOlang/go/src/basic_structure]
1
成功: 进程退出代码 0.
```

```
1.  // 当前程序的包名
2.  package main
3.  // 导入其它的包
4.  import (
5.      "fmt"
6.  )
7.  type (
8.      byte     int8
9.      rune     int32
10.     ByteSize int64
11. )
12.
13. // 类型的自动判断
14. func main() {
15.     b := 1
16.     fmt.Println(b)
17. }
```

这时候自动类型判断为int型

```
1.  // 当前程序的包名
2.  package main
3.  // 导入其它的包
4.  import (
5.      "fmt"
6.  )
7.  type (
8.      byte     int8
9.      rune     int32
10.     ByteSize int64
11. )
12. // 类型的自动判断
13. func main() {
14.     b := false
15.     fmt.Println(b)
16. }
```

这时候就是一个bool型

```
/home/jiemin/code/GOlang/go/src/basic_structure/basic_structure [/home/jiemin/code/GOlang/go/src/basic_structure]
false
成功: 进程退出代码 0.
```

## 多个变量的声明与赋值

- 全局变量的声明可使用 var() 的方式进行简写
- 全局变量的声明不可以省略 var，但可使用并行方式
- 所有变量都可以使用类型推断
- 局部变量不可以使用 var() 的方式简写，只能使用并行方式

```
var (
    // 使用常规方式
    aaa = "hello"
    // 使用并行方式以及类型推断
    sss, bbb = 1, 2
    // ccc := 3 // 不可以省略 var
)
```

```
// 多个变量的声明
var a, b, c, d int
// 多个变量的赋值
a, b, c, d = 1, 2, 3, 4

// 多个变量声明的同时赋值
var e, f, g, h int = 5, 6, 7, 8
// 省略变量类型，由系统推断
var i, j, k, l = 9, 10, 11, 12
// 多个变量声明与赋值的最简写法
i, m, n, o := 13, 14, 15, 16
```

```go
// 当前程序的包名
package main

// 导入其它的包
import (
    "fmt"
)

// 并行方式的声明
func main() {
    var a, b, c, d int = 1, 2, 3, 4
    fmt.Println(a)
    fmt.Println(b)
    fmt.Println(c)
    fmt.Println(d)
}
```

```go
// 当前程序的包名
package main

// 导入其它的包
import (
    "fmt"
)

// 并行方式的声明,以及省略var关键字
func main() {
    var a, b, c, d = 1, 2, 3, 4
    fmt.Println(a)
    fmt.Println(b)
    fmt.Println(c)
```

```
    fmt.Println(d)
}
```

```
// 当前程序的包名
package main

// 导入其它的包
import (
    "fmt"
)

// 使用冒号替代var关键字，进行简写方式声明与赋值
func main() {
    a, b, c, d := 1, 2, 3, 4
    fmt.Println(a)
    fmt.Println(b)
    fmt.Println(c)
    fmt.Println(d)
}
```

```
// 当前程序的包名
package main

// 导入其它的包
import (
    "fmt"
)

type (
    byte     int8
    rune     int32
    ByteSize int64
)

var (

)

//
func main() {
    a, _, c, d := 1, 2, 3, 4
    fmt.Println(a)
    fmt.Println(c)
    fmt.Println(d)
}
```

忽略2的数值

```
/home/jiemin/code/GOlang/go/src/basic_structure/basic_structure  [/home/jiemin/code/GOlang/go/src/basic_structure]
1
3
4
成功: 进程退出代码 0.
```

## 变量的类型转换

- Go中不存在隐式转换，所有类型转换必须显式声明
- 转换只能发生在两种相互兼容的类型之间
- 类型转换的格式：

  <ValueA> [:]= <TypeOfValueA>(<ValueB>)

```
// 在相互兼容的两种类型之间进行转换
var a float32 = 1.1
b := int(a)

// 以下表达式无法通过编译
var c bool = true
d := int(c)
```

```
1.  // 当前程序的包名
2.  package main
3.
4.  // 导入其它的包
5.  import (
6.      "fmt"
7.  )
8.
9.  type (
10.     byte     int8
11.     rune     int32
12.     ByteSize int64
13. )
14.
15. var ()
16.
17. func main() {
18.     var a float32 = 100.1
19.     fmt.Println(a)
20.     b := int(a)
21.     fmt.Println(b)
22. }
```

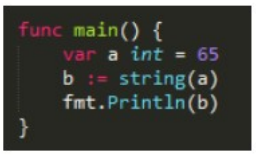
```
/home/jiemin/code/GOlang/go/src/basic_structure/basic_structure  [/home/jiemin/code/GOlang/go/src/basic_structure]
100.1
100
成功: 进程退出代码 0.
```

```go
// 当前程序的包名
package main

// 导入其它的包
import (
    "fmt"
)

type (
    byte     int8
    rune     int32
    ByteSize int64
)

var ()

func main() {
    var a float32 = 100.1
    fmt.Println(a)
    b := bool(a)
    fmt.Println(b)
}
```

```
# _/home/jiemin/code/GOlang/go/src/basic_structure
./basic_structure.go:20: cannot convert a (type float32) to type bool
错误: 进程退出代码 2.
```

课堂作业

- 请尝试运行以下代码，看会发生什么，并思考为什么。




string() 表示将数据转换成文本格式，因为计算机中存储的任何东西本质上都是数字，因此此函数自然地认为我们需要的是用数字65表示的文本 A。

作业：

```go
package main

import (
```

```
    "fmt"
)

type (
    byte     int8
    rune     int32
    ByteSize int64
)

func main() {
    var a int = 65
    fmt.Println(a)
    b := string(a)
    fmt.Println(b)
}
```

```
/home/jiemin/code/GOlang/go/src/temp1/temp1  [/home/jiemin/code/GOlang/go/src/temp1]
65
A
成功: 进程退出代码 0.
```

引入 strconv包

```
package main

import (
    "fmt"
    "strconv"
)

type (
    byte     int8
    rune     int32
    ByteSize int64
)

func main() {
    var a int = 65
    fmt.Println(a)
    b := strconv.Itoa(a)
    fmt.Println(b)
    a, _ = strconv.Atoi(b)
    fmt.Println(a)
}
```

```
/home/jiemin/code/GOlang/go/src/temp1/temp1  [/home/jiemin/code/GOlang/go/src/temp1]
65
65
65
成功: 进程退出代码 0.
```