链滴

Go 边看边练 -《Go 学习笔记》系列（二）

作者：88250

原文链接：https://ld246.com/article/1437558810339

来源网站：链滴

上一篇： https://hacpai.com/article/1437497122181

ToC

- Go 边看边练 -《Go 学习笔记》系列（一）- 变量、常量
- Go 边看边练 -《Go 学习笔记》系列（二）- 类型、字符串
- Go 边看边练 -《Go 学习笔记》系列（三）- 指针
- Go 边看边练 -《Go 学习笔记》系列（四）- 控制流1
- Go 边看边练 -《Go 学习笔记》系列（五）- 控制流2
- Go 边看边练 -《Go 学习笔记》系列（六）- 函数
- Go 边看边练 -《Go 学习笔记》系列（七）- 错误处理
- Go 边看边练 -《Go 学习笔记》系列（八）- 数组、切片
- Go 边看边练 -《Go 学习笔记》系列（九）- Map、结构体
- Go 边看边练 -《Go 学习笔记》系列（十）- 方法
- Go 边看边练 -《Go 学习笔记》系列（十一）- 表达式
- Go 边看边练 -《Go 学习笔记》系列（十二）- 接口
- Go 边看边练 -《Go 学习笔记》系列（十三）- Goroutine
- Go 边看边练 -《Go 学习笔记》系列（十四）- Channel

1.3 基本类型

更明确的数字类型命名，支持 Unicode，支持常用数据结构。

```
<table style="width: 80%;">
  <tr>
    <th>类型</th>
    <th>长度</th>
    <th>默认值</th>
    <th>说明</th>
  </tr>
  <tr>
    <td>bool</td>
    <td>1</td>
    <td>false</td>
    <td></td>
  </tr>
  <tr>
    <td>byte</td>
    <td>1</td>
    <td>0</td>
    <td>uint8</td>
  </tr>
```

```
<tr>
  <td>rune</td>
  <td>4</td>
  <td>0</td>
  <td>Unicode Code Point, int32</td>
</tr>
<tr>
  <td>int, uint</td>
  <td>4 或 8</td>
  <td>0</td>
  <td>32 或 64 位</td>
</tr>
<tr>
  <td>int8, uint8</td>
  <td>1</td>
  <td>0</td>
  <td>-128 ~ 127, 0 ~ 255</td>
</tr>
<tr>
  <td>int16, uint16</td>
  <td>2</td>
  <td>0</td>
  <td>-32768 ~ 32767, 0 ~ 65535</td>
</tr>
<tr>
  <td>int32, uint32</td>
  <td>4</td>
  <td>0</td>
  <td>-21亿 ~ 21 亿, 0 ~ 42 亿</td>
</tr>
<tr>
  <td>int64, uint64</td>
  <td>8</td>
  <td>0</td>
  <td></td>
</tr>
<tr>
  <td>float32</td>
  <td>4</td>
  <td>0.0</td>
  <td></td>
</tr>
<tr>
  <td>float64</td>
  <td>8</td>
  <td>0.0</td>
  <td></td>
</tr>
<tr>
  <td>complex64</td>
  <td>8</td>
  <td></td>
  <td></td>
</tr>
```

```
<tr>
  <td>complex128</td>
  <td>16</td>
  <td></td>
  <td></td>
</tr>
<tr>
  <td>uintptr</td>
  <td>4 或 8</td>
  <td></td>
  <td>足以存储指针的 uint32 或 uint64 整数</td>
</tr>
<tr>
  <td>array</td>
  <td></td>
  <td></td>
  <td>值类型</td>
</tr>
<tr>
  <td>struct</td>
  <td></td>
  <td></td>
  <td>值类型</td>
</tr>
<tr>
  <td>string</td>
  <td></td>
  <td>""</td>
  <td>UTF-8 字符串</td>
</tr>
<tr>
  <td>slice</td>
  <td></td>
  <td>nil</td>
  <td>引用类型</td>
</tr>
<tr>
  <td>map</td>
  <td></td>
  <td>nil</td>
  <td>引用类型</td>
</tr>
<tr>
  <td>channel</td>
  <td></td>
  <td>nil</td>
  <td>引用类型</td>
</tr>
<tr>
  <td>interface</td>
  <td></td>
  <td>nil</td>
  <td>接口</td>
</tr>
```

```
  <tr>
    <td>function</td>
    <td></td>
    <td>nil</td>
    <td>函数</td>
  </tr>
</table>
```

支持八进制、十六进制，以及科学记数法。标准库 math 定义了各数字类型取值范围。

```
a, b, c, d := 071, 0x1F, 1e9, math.MinInt16
```

空指针值 nil，而非 C/C++ NULL。

1.4 引用类型

引用类型包括 slice、map 和 channel。它们有复杂的内部结构，除了申请内存外，还需要初始化相属性。

内置函数 new 计算类型大小，为其分配零值内存，返回指针。而 make 会被编译器翻译成具体的创函数，由其分配内存和初始化成员结构，返回对象而非指针。

```
a := []int{0, 0, 0} // 提供初始化表达式。
a[1] = 10

b := make([]int, 3) // makeslice
b[1] = 10

c := new([]int)
c[1] = 10 // Error: invalid operation: c[1] (index of type *[]int)
```

有关引用类型具体的内存布局，可参考后续章节。

1.5 类型转换

不支持隐式类型转换，即便是从窄向宽转换也不行。

```
var b byte = 100
// var n int = b // Error: cannot use b (type byte) as type int in assignment
var n int = int(b) // 显式转换
```

使用括号避免优先级错误。

```
*Point(p) // 相当于 *(Point(p))
(*Point)(p)
<-chan int(c) // 相当于 <-(chan int(c))
(<-chan int)(c)
```

同样不能将其他类型当 bool 值使用。

```
a := 100
if a { // Error: non-bool a (type int) used as if condition
    println("true")
}
```

1.6 字符串

字符串是不可变值类型，内部用指针指向 UTF-8 字节数组。

- 默认值是空字符串 ""。
- 用索引号访问某字节，如 s[i]。
- 不能用序号获取字节元素指针， &s[i] 非法。
- 不可变类型，无法修改字节数组。
- 字节数组尾部不包含 NULL。

runtime.h

```
struct String
{
    byte* str;
    intgo len;
};
```

使用索引号访问字符 (byte)。

```
s := "abc"
println(s[0] == '\x61', s[1] == 'b', s[2] == 0x63)
```

输出：

```
true true true
```

使用 "`" 定义不做转义处理的原始字符串，支持跨行。

```
s := `a
b\r\n\x00
c`

println(s)
```

输出：

```
a
b\r\n\x00
c
```

连接跨行字符串时，"+" 必须在上一行末尾，否则导致编译错误。

```
s := "Hello, " +
"World!"

s2 := "Hello, "
+ "World!" // Error: invalid operation: + untyped string
```

支持用两个索引号返回子串。子串依然指向原字节数组，仅修改了指针和长度属性。

```
s := "Hello, World!"
s1 := s[:5] // Hello
s2 := s[7:] // World!
s3 := s[1:5] // ello
```

单引号字符常量表示 Unicode Code Point，支持 \uFFFF、\U7FFFFFFF、\xFF 格式。对应 rune 类，UCS-4。

<iframe style="border:1px solid" src="https://wide.b3log.org/playground/2d0bb32225a7d73052e06450efaae80.go?embed=true" width="99%" height="320"></iframe>

要修改字符串，可先将其转换成 []rune 或 []byte，完成后再转换为 string。无论哪种转换，都会重分配内存，并复制字节数组。

<iframe style="border:1px solid" src="https://wide.b3log.org/playground/6cb92a0c330f54a48b9a13d908775f5.go?embed=true" width="99%" height="450"></iframe>

用 for 循环遍历字符串时，也有 byte 和 rune 两种方式。

<iframe style="border:1px solid" src="https://wide.b3log.org/playground/26ab3e14ef786b3301acfc564e99348.go?embed=true" width="99%" height="420"></iframe>

下一篇： https://hacpai.com/article/1437719712835
