编程语言的构成

一周学会Go+

@xushiwei

常量 (const)

- 数值
  - 整数：-1, 0, 100
  - 实数 (浮点数)：1.2, 2.3e-6, -1.2e3
  - 复数：2+5i, 1.2i
- 布尔
  - true, false
- 文本 (字符串)
  - "Hello", "你好", "123", "-1.2e3", "true"

运算 (operator)

- 数值运算 (整数/浮点数/复数)：+ - * / %
  - 1 + 2, 10 % 3, 32.5 / 3
- 文本运算 (字符串)：+(连接)
  - "-1" + "23"
- 比较运算 (数值/文本)：> >= < <= == !=
  - 数值比较 (值大小)：1 > 2, 32 >= 0, 100 != 1e2
  - 文本比较 (字典顺序)："Hello" > "-123"
- 布尔运算 (布尔)：|| && !
  - false || true, 32 >= 0 && !false
- 位运算 (整数)：| & ^(异或/取反)
  - 3 | 51, ^32, 32 ^ 0

函数 (func)

- sin(-1.2), pow(2, 3)
- 运算即函数
  - 1 + 2 等价于 add(1, 2)
  - 32 >= 0 等价于 greatThan(32, 0)
  - !false 等价于 not(false)
  - true && false 等价于 and(true, false)
- 运算优先级
  - 32 >= 0 && !false 等价于 and(greatThan(32, 0), not(false))

- 不同语言运算有细微差异
  - 求余运算：大部分语言有求余运算 10 % 3，少量语言用 mod(10, 3)
  - 指数运算：有些语言用 2 ** 3 或 2 ^ 3，大部分语言用 pow(2, 3)

命令 (command)

- 有些语言有命令的概念，它其实也是函数 (调用时省略括号)
  - step 10 等价于 step(10)
  - say "Hello", 3 等价于 say("Hello", 3)

包 (package)

- 可以简单看作函数集合

```
import "math"
echo math.sin(math.Pi / 3)
```

变量 (var)

- 变量：可以修改的值 var x = 123
- 常量：不可修改的值 const x = 123

- x := 123 (第一次设置的值)
  - 等价于 var x = 123
- x = 10 (后续再改)
- x += 10 (边算边改)
  - 即 x = x + 10

- 语言差异
  - 有些语言第一次和后续修改都用 =
  - 变量和数学中的未知数 (值未知但不可修改) 并不是一个概念 (数学中并不存在变量)

流程控制 (control)

- 条件 (if/else, switch/case)
  - if cond { ... }
  - if cond { ... } else { ... }
  - switch expr { case v1: ...; case v2: ...; default: ... }
- 循环 (for)
  - 条件循环：for cond { ... } 或 while cond { ... }
  - 循环N次：for :N { ... } 或 repeat N { ... }
  - 死循环：for { ... }
  - 结束循环 (break)
- 语言差异
  - 大部分语言条件需要加 ()，例如:
    - if (cond) { ... }
    - while (cond) { ... }
  - 循环不同语言差异较大，有些语言是 until cond { ... }，和 while 条件相反
    - 等价于：while !cond { ... }

求 1+2+...+100

```
sum := 0
i := 1
for :100 {
    sum += i
    i += 1
}
echo sum
```

自定义函数 (UDF)

```
func gauss(n int) int {
    sum := 0
    i := 1
    for :n {
        sum += i
        i += 1
    }
    return sum
}

echo gauss(100)
```

类型 (type)

- 数值
  - 整数： int, uint
  - 实数 (浮点数)： float32, float64
  - 复数： complex64, complex128
- 布尔
  - bool
- 文本 (字符串)
  - string

闭包 (closure)

- 闭包：没有名字 (且省略类型) 的函数

- 无参数闭包 => { ... }
- 单参数闭包 x => { ... }
- 多参数闭包 (x, y, z) => { ... }

事件 (如何使用闭包)

- 事件是普通函数，特别之处只是它有参数是一个函数 (callback)

- 消息：onMsg "game over", => { ... }
- 点击：onClick => { ... }
- 按键：onKey key => { ... }

数据结构：列表与字典

- 列表 []V
  - [1, 3, 5]
  - ["Hello", "123", "-1.2e3"]
  - 列表操作
    - 取值/子列表：a[i], a[i:j], a[i:], a[:j]
    - 添加值：a = append(a, v)
- 字典 map[K]V
  - {"Mon": 1, "Tue": 2, "Sat": 6}
  - 字典操作
    - 取值：v = a[k] 或 v, ok := a[k]
    - 添加值：a[k] = v
    - 删除值：delete(a, k)
- 语言差异
  - 列表在有的语言中叫动态数组

类文件 (classfile)

- 自定义类型 UDT (数据结构 = 变量 + 函数)
- Rect.gox

```
var Width, Height float64

func Area() float64 {
    return Width * Height
}

rect := Rect{20, 30}
echo rect.Area()
```

类文件 vs. DSL

- 类文件 (classfile) 不只是自定义类型
- 它简化了领域编程 (取代 DSL)

Go+ 编程哲学

- 尽可能少的概念
  - const、var、operator/func/command/closure、package
  - control: if/else, switch/case, for/break
  - type、list []V、map[K]V
  - classfile (var + func as UDT)
- 丰富的函数
  - 函数多不会带来理解负担 (就像这个世界上人名很多一样)
  - operator 也是函数，Go+ 支持了尽可能多的 operator (例如它提供了 ->, <> 这样的操作符)