链滴

Go 边看边练 -《Go 学习笔记》系列（六）

作者：88250

原文链接：https://ld246.com/article/1438164538421

来源网站：链滴

上一篇： https://hacpai.com/article/1438070631857

ToC

3.1 函数定义

不支持 嵌套 (nested)、重载 (overload) 和 默认参数 (default parameter)。

- 无需声明原型。
- 支持不定长长变参。
- 支持多返回值。
- 支持命名返回参数。
- 支持匿名函数和闭包。

使用关键字 func 定义函数，左大括号依旧不能另起一行。

```
func test(x, y int, s string) (int, string) { // 类型相同的相邻参数可合并。
    n := x + y // 多返回值必须用括号。
    return n, fmt.Sprintf(s, n)
}
```

函数是第一类对象，可作为参数传递。建议将复杂签名定义为函数类型，以便于阅读。

<iframe style="border:1px solid" src="https://wide.b3log.org/playground/038adfea736d9c60

<iframe style="border:1px solid" src="https://wide.b3log.org/playground/ea1abd9c5be61a326090d12594cc9f.go?embed=true" width="99%" height="600"></iframe>

有返回值的函数，必须有明确的终止语句，否则会引发编译错误。

3.2 变参

变参本质上就是 slice。只能有一个，且必须是最后一个。

<iframe style="border:1px solid" src="https://wide.b3log.org/playground/ea1abd9c5be61a302529a8ded4861b9.go?embed=true" width="99%" height="500"></iframe>

使用 slice 对象做变参时，必须展开。

```
func main() {
    s := []int{1, 2, 3}
    println(test("sum: %d", s...))
}
```

3.3 返回值

不能用容器对象接收多返回值。只能用多个变量，或 "_
 忽略。

```
func test() (int, int) {
    return 1, 2
}

func main() {
    // s := make([]int, 2)
      // s = test() // Error: multiple-value test() in single-value context

      x, _ := test()
    println(x)
}
```

多返回值可直接作为其他函数调用实参。

```
func test() (int, int) {
    return 1, 2
}

func add(x, y int) int {
    return x + y
}

func sum(n ...int) int {
    var x int
    for _, i := range n {
        x += i
    }
```

```
    return x
}

func main() {
    println(add(test()))
    println(sum(test()))
}
```

命名返回参数可看做与形参类似的局部变量，最后由 return 隐式返回。

```
func add(x, y int) (z int) {
    z = x + y

    return
}

func main() {
    println(add(1, 2))
}
```

命名返回参数可被同名局部变量遮蔽，此时需要显式返回。

```
func add(x, y int) (z int) {
    { // 不能在一个级别，引发 "z redeclared in this block" 错误。
        var z = x + y
        // return // Error: z is shadowed during return
        return z // 必须显式返回。
    }
}
```

命名返回参数允许 defer 延迟调用通过闭包读取和修改。

```
func add(x, y int) (z int) {
    defer func() {
        z += 100
    }()

    z = x + y
    return
}

func main() {
    println(add(1, 2)) // 输出: 103
}
```

显式 return 返回前，会先修改命名返回参数。

```
func add(x, y int) (z int) {
    defer func() {
        println(z) // 输出: 203
```

```
    }()

    z = x + y
    return z + 200 // 执行行顺序: (z = z + 200) -> (call defer) -> (ret)
}

func main() {
    println(add(1, 2)) // 输出: 203
}
```

3.4 匿名函数

匿名函数可赋值给变量，做为结构字段，或者在 channel 里传送。

```
// --- function variable ---

fn := func() { println("Hello, World!") }
fn()

// --- function collection ---

fns := [](func(x int) int){
    func(x int) int { return x + 1 },
    func(x int) int { return x + 2 },
}

println(fns[0](100))

// --- function as field ---

d := struct {
    fn func() string
}{
    fn: func() string { return "Hello, World!" },
}

println(d.fn())

// --- channel of function ---

fc := make(chan func() string, 2)
fc <- func() string { return "Hello, World!" }
println((<-fc)())
```

闭包复制的是原对象指针，这就很容易解释延迟引用现象。

<iframe style="border:1px solid" src="https://wide.b3log.org/playground/25247ddce7ba5bf6ca2c8dad951d5bb.go?embed=true" width="99%" height="500"></iframe>

在汇编层面，test 实际返回的是 FuncVal 对象，其中包含了匿名函数地址、闭包对象指针。当调用名函数时，只需以某个寄存器传递该对象即可。

FuncVal { func_address, closure_var_pointer ... }

下一篇： https://hacpai.com/article/1438260619759

- **本系列是基于雨痕的《Go 学习笔记》（第四版）整理汇编而成，非常感谢雨痕的辛勤付出与分享！**
- 转载请注明：文章转载自： **黑客与画家的社区** [https://hacpai.com]
- 如果你觉得本章节做得不错，请在下面打赏一下吧~

社区小贴士

- 关注标签 [golang] 可以方便查看 Go 相关帖子
- 关注作者后如有新帖将会收到通知