函数详解2

作者：少林之巅

目录

变量作用域

1. 全局变量，在程序整个生命周期有效。

```
var a int = 100
```

变量作用域

2. 局部变量，分为两种：1）函数内定义，2）语句块内定义。

```
func add(a int, b int) int {
    var sum int = 0
    //sum是局部变量
    if a > 0 {
       var c int = 100
       //c是布局变量，尽在if语句块有效
    }
}
```

变量作用域

3. 可见性，包内任何变量或函数都是能访问的。包外的话，首字母大写是可导出的能够被其他包访问或调用。小写表示是私有的，不能被外部的包访问。

```
func add(a int, b int) int {

}
//add这个函数只能在包内部调用，是私有的，不能被外部的包调用
```

匿名函数

1. 函数也是一种类型，因此可以定义一个函数类型的变量

匿名函数

2. 匿名函数，即没有名字的函数

匿名函数

3. defer中使用匿名函数

匿名函数

4. 函数作为一个参数

闭包

1. 闭包：一个函数和与其相关的引用环境组合而成的实体

```
package main

import “fmt”

func main() {
        var f = Adder()
        fmt.Print(f(1),” - “)
        fmt.Print(f(20),” - “)
        fmt.Print(f(300))
}

func Adder() func(int) int {
        var x int
        return func(d int) int {
                x += d
                return x
        }
}
```

闭包

2. 闭包的例子1

```
package main

func add(base int) func(int) int {
    return func(i int) int {
        base += i
        return base
    }
}

func main() {
    tmp1 := add(10)
    fmt.Println(tmp(1), tmp(2))
    tmp2 := add(100)
    fmt.Println(tmp2(1), tmp2(2))
}
```

闭包

3. 闭包的例子2

```
package main

import (
	"fmt"
	"strings"
)
func makeSuffixFunc(suffix string) func(string) string {
	return func(name string) string {
		if !strings.HasSuffix(name, suffix) {
			return name + suffix
		}
		return name
	}
}
func main() {
	func1 := makeSuffixFunc(".bmp")
	func2 := makeSuffixFunc(".jpg")
	fmt.Println(func1("test"))
	fmt.Println(func2("test"))
}
```

闭包

4. 闭包的例子3

```
func calc(base int) (func(int) int, func(int) int) {

    add := func(i int) int {
        base += i
        return base
    }

    sub := func(i int) int {
        base -= i
        return base
    }

    return add, sub
}

func main() {
    f1, f2 := calc(10)
    fmt.Println(f1(1), f2(2))
    fmt.Println(f1(3), f2(4))
    fmt.Println(f1(5), f2(6))
    fmt.Println(f1(7), f2(8))
}
```

闭包

5. 闭包的例子4

```
func calc(base int) (func(int) int, func(int) int) {

    add := func(i int) int {
        base += i
        return base
    }

    sub := func(i int) int {
        base -= i
        return base
    }

    return add, sub
}

func main() {
    f1, f2 := calc(10)
    fmt.Println(f1(1), f2(2))
    fmt.Println(f1(3), f2(4))
    fmt.Println(f1(5), f2(6))
    fmt.Println(f1(7), f2(8))
}
```

闭包

6. 闭包的例子5

```
package main

import (
    "fmt"
    "time"
)

func main() {
    for i:=0; i<5; i++ {
        go func(){
            fmt.Println(i)
        }()
    }
    time.Sleep(time.Second)
}
```

练习

1. 实现一个插入排序

2.实现一个选择排序

3. 实现一个冒泡排序