Go 语言闭包

所谓“闭包”，指的是一个拥有许多变量和绑定了这些变量的环境的表达式（通常是一个函数），因而这些变量也是该表达式的一部分。

闭包可以理解成**定义在一个函数内部的函数。**在本质上，闭包是将函数内部和函数外部连接起来的桥梁。或者说是函数和其引用环境的组合体。

闭包是由函数及其相关引用环境组合而成的实体(即：闭包=函数+引用环境)。

示例：

```
package main

import "fmt"

// 外部引用函数参数局部变量
func add(base int) func(int) int {
    return func(i int) int {
        base += i
        return base
    }
}

func main() {
    tmp1 := add(10)
    fmt.Println(tmp1(1), tmp1(2))

    tmp2 := add(100)
    fmt.Println(tmp2(1), tmp2(2))
}
#结果
11 13
101 103
```

> 变量tmp1是一个函数并且引用了外部作用域中的base变量，此时tmp1就是闭包，在tmp1生命周期内，变量base也一直有效。

```
package main

import "fmt"

// 返回2个函数类型的返回值
func test01(base int) (func(int) int, func(int) int) {
    // 定义2个函数，并返回
    // 相加
    add := func(i int) int {
        base += i
        return base
    }
    // 相减
    sub := func(i int) int {
```

```
        base -= i
        return base
    }
    // 返回
    return add, sub
}

func main() {
    f1, f2 := test01(10)
    // base一直是没有消
    fmt.Println(f1(1), f2(2))
    // 此时base是9
    fmt.Println(f1(3), f2(4))
}
#结果
11 9
12 8
```