1.函数

2.不定参数的函数

```
//定义一个有参数没有返回值的函数
func testFunc(str string) {
    fmt.Println("str is ", str)
}

func testFunc2(args ...int) {
    fmt.Println("这里是不定参数", "元素个数总共有", len(args))
    //打印里面的每一个元素
    for i := 0; i < len(args); i++ {
        fmt.Printf("arg[%d] is %d\n", i, args[i])
    }

    for i, data := range args {
        fmt.Printf("arg[%d] is also %d\n", i, data)
    }
}
```

3.不定参数一定只能放在参数列表的最后一个

4.不定参数的传递

```
func test(args ...int) {
    //全部元素传递给myfunc
    //myfunc(args...)

    //只想把后2个参数传递给另外一个函数使用
    myfunc2(args[:2]...) //0~2（不包括数字2），      传递过去
    myfunc2(args[2:]...) //从args[2]开始(包括本身)，把后面所有元素传递过去
}
```

test(args[2:])表示省略前两个

test(args[:2])只保留前两个

5.有一个返回值的函数,常用写法如下,即给返回值命名

```go
//给返回值起一个变量名，go推荐写法
func myfunc02() (result int) {
    return 666
}

//给返回值起一个变量名，go推荐写法
func myfunc03() (result int) {

    result = 666
    return
}
```

6.返回值可以有多个

7.函数调用流程,先进后出的特点

8.递归函数

```go
//用递归来实现数的累加
func testAdd(num int) int {
    //先定义一个返回值
    var result int
    //给出结束条件,防止栈溢出
    if num == 1 {
        return 1
    }
    result = num + testAdd(num-1)
    num--
    return result
}
```

9.函数类型:go语言中函数也是一种类型,体现了面向对象编程的多态的特性

```go
//定义函数类型,函数本身也可以当成一个类型来使用
type FuncType func(int, int) int

func add(a int, b int) (result int) {
    return a + b
}

func main() {
```

```
    //定义一个函数变量
    var testFunctype FuncType
    testFunctype = add
    result := testFunctype(3, 7)
    fmt.Println("result is ", result)

}
```

10.回调函数,函数有一个参数是函数类型,这个函数就是回调函数

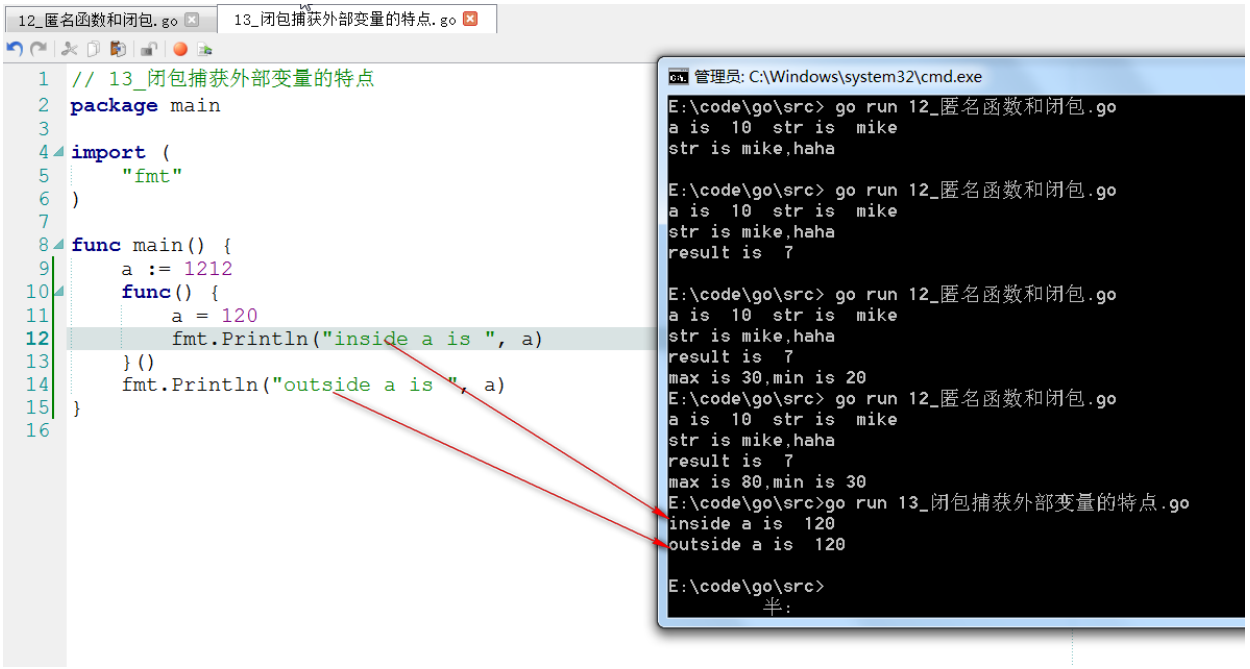
```
//定义一个函数类型
type FuncType func(int, int) int

//定义一个函数
func add(a int, b int) int {
    return a + b
}

//定义一个回调函数
func Calc(a int, b int, testFunc FuncType) (result int) {
    value := testFunc(a, b)
    return value
}

func main() {
    result := Calc(3, 8, add)
    fmt.Println("result is ", result)
}
```

11.定义匿名函数和调用的流程

```
func main() {
    a := 10
    str := "mike"
    //定义匿名函数,然后调用
    testFunc := func() {
        fmt.Println("a is ", a, " str is ", str)
```

```
    }
    testFunc()

    //定义匿名函数,同时调用
    func() {
        fmt.Printf("str is %s,haha\n", str)
    }()

    //带参数的匿名函数
    func(x int, y int) {
        fmt.Println("result is ", x+y)
    }(3, 4)

    //匿名函数,有参有返回值
    max, min := func(i int, j int) (max int, min int) {
        if i > j {
            return i, j
        } else {
            return j, i
        }
    }(80, 30)

    fmt.Printf("max is %d,min is %d", max, min)
}
```

12.闭包内部修改了外部的变量的话,外部变量的值会改变




13.闭包的特点:闭包不关心捕获的变量和常量是否已经超出了作用域,所以只有闭包还在使用它,这些变量就还会存在

```
//定义一个函数返回一个闭包函数
//闭包不关心捕获的变量和常量是否已经超出了作用域,所以只有闭包还在使用它,这些变量就还会存在
func testFunc() func() int {
    var a int
    return func() int {
        a++
        return a * a
    }
}

func main() {
    //    a := 1212
    //    func() {
    //        a = 120
    //        fmt.Println("inside a is ", a)
    //    }()
    //    fmt.Println("outside a is ", a)
    test := testFunc()
    fmt.Println("result is ", test())
    fmt.Println("result is ", test())
```

```
    fmt.Println("result is ", test())
}
```

14.defer和闭包结合使用

```
func main() {
    var a int
    var b int

    a = 1
    b = 1

    //这里面闭包捕获的是形参里面的a和b,第一次赋值的时候已经有了
    defer func(a int, b int) {
        fmt.Println("defer a + b is ", a+b)
    }(a, b)

    a = 12
    b = 23
    fmt.Println("a + b is ", a+b)

}
```

15.获取命令行参数

```
func main() {
    //    获取参数切片
    list := os.Args
    fmt.Println("list 长度是 ", len(list))
    //对切片进行遍历
    for i, data := range list {
        fmt.Printf("list[%d] is %s\n", i, data)
    }

}
```

16.局部变量的特点

```go
func main() {
    //定义在{}里面的变量就是局部变量，只能在{}里面有效
    //执行到定义变量那句话，才开始分配空间，离开作用域自动释放
    //作用域，变量其作用的范围

    //a = 111
    {
        i := 10
        fmt.Println("i = ", i)
    }
    //i = 111

    if flag := 3; flag == 3 {
        fmt.Println("flag = ", flag)
    }
```

## 17.全局变量的特点

```go
//定义在函数外部的变量是全局变量
//全局变量在任何地方都能使用
var a int

func main() {
    a = 10
    fmt.Println("a = ", a)

    test()
}
```

## 18.工程管理的一些注意事项

1、分文件编程（多个源文件），必须放在src目录
2、设置GOPATH环境变量
3、同一个目录，包名必须一样
4、go env查看go相关的环境路径
5、同一个目录，调用别的文件的函数，直接调用即可，无需包名引用

src

main.go

```go
package main //必须

func main() {

}
```

test.go

```go
package main

import "fmt"

func test() {
    fmt.Println("this is a test func")
}
```

## 19.函数名首字母大写的话才可以被别的包识别到

## 20.导入一个包的时候,会首先调用这个包的init函数

## 21.工程管理的相关总结

工程管理总结.go

```
src: 放源代码

如果有多个文件或多个包

1) 配置GOPATH环境变量， 配置src同级目录的绝对路径
    如：C:\Users\superman\Desktop\code\go\src\25_工程管理：不同目录

2) 自动生成bin或pkg目录，需要使用go install命令(了解)
    除了要配置GOPATH环境变量，还要配置GOBIN环境变量

src: 放源代码
bin: 放可执行程序
pkg: 放平台相关的库
```