# Go 语法汇总

## 数据类型

---

var varName type，var var1,var2… type，var varName type = Value，var varName1,varName2 type = Value1,Value2，var varName1,varName2=Value1,Value2，varName1,varName2:=Value1,Value2，定义数据类型。

声明了没有被使用的变量将在编译时报错。

常量定义： const varName = Value , const varName type = Value

string 字符串类型值不可改变，但是可以切片，字符串可以使用+进行连接

iota 用来声明 enum，表示自加 1，初始为 0

var arrayName [N]type 用来声明数组，或者使用 arrayName := [N]type{ v1,v2…} 来声明

数组声明可以嵌套

slice 用来表示切片，声明方式 var sliceName []type 或者 sliceName := []type{v1,v2…}

slice 保存的是引用而非实体

在 slice 中有一些内置函数，len 获取长度，cap 获取最大容量，append 追加数据，copy 用来拷贝数据

map 声明方式为 var mapName map[keyType] valueType 或者 mapName := make(map[keyType]valueType)

map 可通过 key : value 初始化

make 用于内建类型的内存分配，new 用于各种类型的内存分配，new 返回指针而 make 返回非 0 的值

## 流程控制

if 语句不需要括号，在 if 语句中可以声明变量，用分好分割 if 语句的条件判断

```
if x:=function();x<10 {
    fmt.Printf("x < 10,%d\n",x);
}else{
    fmt.Print("x >= 10 ,%d\n",x);
}
```

goto 语句类似 C 语言，但是跳转到必须在当前函数内定义的标签

for 语句类似 C 语言，但是 break 和 continue 可以跟标号，跳出多重循环。

switch 语句不用 break，如果想强行执行下面的 case 可以使用 fallthrough

## 函数

声明方式：

```
func funcName(input1 type1, input2 type2) (output1 type1, output2 type2)
```

func 用来声明函数，函数名为 funcName，后面跟输入，输出的数据类型。

函数可以有多个返回值

函数的值操作和指针操作类似 C 语言，内置类型中的 string,slice,map 直接使用的是类似的指针传递，不用使用取地址符，但是，如果需要改变 slice 的长度，则需要取地址穿指针。

defer 语句用来表示在函数返回前执行的语句。

type typeName func(input1 inputType1 , input2 inputType2 [, ...]) (result1 resultType1 [, ...])用来声明一个函数类型，主要用于高阶函数中。

import 用来导入包，package 用来导出包，包操作使用.操作符

## Struct 类型

声明方式：

```
type Person struct {
        name string
        age int
}
```

匿名方式，匿名方式下 A 含有 B 的所有类型

```
type Student struct {
        Person  // 默认 Person 的所有字段
        Speciality string
}
```

如果匿名类型中有字段和本身有冲突，可以使用匿名类型+.访问

类型的方法声明：

```
func (r ReceiverType) funcName(parameters) (results)
```

可以使用：type typeName typeLiteral 来自定义类型，定义完以后可以使用方法来扩展类型的功能。

需要改变 struct 内部的值时，需要将 ReceiverType 定义为*指针类型，但是调用的时候不需要，go 语言自动帮你完成了。

方法可以继承，可以重载

## interface 接口

---

type InterfaceName interface 用来定义 inerface

interface 类型定义了一组方法，如果某个对象实现了某个接口的所有方法，则此对象就实现了此接口。

空 interface(interface{})不包含任何的 method，正因为如此，所有的类型都实现了空 interface

一个函数把 interface{}作为参数，那么他可以接受任意类型的值作为参数，如果一个函数返回 interface{},那么也就可以返回任意类型的值

value, ok = element.(T)，这里 value 就是变量的值，ok 是一个 bool 类型，element 是 interface 变量，T 是断言的类型，如果 ok 为 true 则表示，element 确实是 T 类型的。

interface 可以嵌套

## 并发

---

使用 go 关键字+函数名实现并发

使用 channel 实现线程间通讯，channel 通过 make 构造，使用<-来发送和接受数据。

chan 是 channel 的关键字，后面跟数据类型 ch <- v 发送数据，v:=<-ch 接收数据，ch 是 chan 类型。