一、从“hello world!”开始

```
package main

import "fmt"

func main() {
    fmt.Println("Hello world!")
}
```

1.Golang 执行流程分析

第一种：对源码编译后，再执行；

第二种：对源码直接执行 go run 源码。

两种执行流程的方式区别

> 1. 如果我们先编译生成了可执行文件，那么我们可以将该可执行文件拷贝到没有 go 开发环境的机器上，仍然可以运行。**
> 2. 如果我们是直接 go run go 源代码，那么如果要在另外一个机器上这么运行，也需要 go 开发环境，否则无法执行。
> 3. 在编译时，编译器会将程序运行依赖的库文件包含在可执行文件中，所以，可执行文件变大了很多。**

2.Go 程序开发的注意事项

- Go 源文件以 "go" 为扩展名。
- Go 应用程序的执行入口是 main()函数。

- Go 语言严格区分大小写。
- Go 方法由一条条语句构成，每个语句后不需要分号(Go 语言会在每行后自动加分号)，这也体现出 Golang 的简洁性。
- Go 编译器是一行行进行编译的，因此我们一行就写一条语句，不能把多条语句写在同一个，否则报错。
- go 语言定义的变量或者 import 的包如果没有使用到，代码不能编译通过。

3.自问自答的小小知识点

> **Go 语言的 SDK 是什么?**
> SDK 就是软件开发工具包。我们做 Go 开发，首先需要先安装并配置好 sdk.
> **Golang 程序的编写、编译、运行步骤是什么? 能否一步执行?**
> 编写：就是写源码
> 编译：go build 源码 =》 生成一个二进制的可执行文件
> **运行：**
>
> \1. 对可执行文件运行 xx.exe ./可执行文件
> \2. go run 源码

二、变量（变量是程序的基本组成单位）

1、变量的概念

变量相当于内存中一个数据存储空间的表示，有趣的说：可以将变量做是一个房间的门牌号，通过门牌号我们可以找到房间，同样的道理，通过变量名可以访问到变量(值)。

2、变量的使用步骤

1. 声明变量(也叫:定义变量)
2. 非变量赋值
3. 使用变量

3.小小demo

```
package main

import "fmt"

func main() {
    var i int                 //定义变量/声明变量
    i = 10                    //给变量赋值
    fmt.Println("i = ", i) //使用变量
}
```

4、Golang 变量使用的三种方式

1. 第一种：指定变量类型，声明后若不赋值，使用默认值。
2. 第二种：根据值自行判定变量类型(类型推导)
3. 第三种：省略 var, 注意 :=左侧的变量不应该是已经声明过的，否则会导致编译错误

```
package main

import "fmt"

func main() {
    //第一种：指定变量类型，声明后若不赋值，使用默认值。
    var i int
    fmt.Println("i=", i)

    // 第二种：根据值自行判定变量类型(类型推导)
```

```
    var num = 10.11
    fmt.Println("num=", num)

    //第三种：省略 var, 注意 :=左侧的变量不应该是已经声明过的，否则会导致编译错误
    name := "tom"
    fmt.Println("name=", name)

}

//输出:
//i= 0
//num= 10.11
//name= tom
```

5.多变量声明

在编程中，有时我们需要一次性声明多个变量，Golang 也提供这样的语法

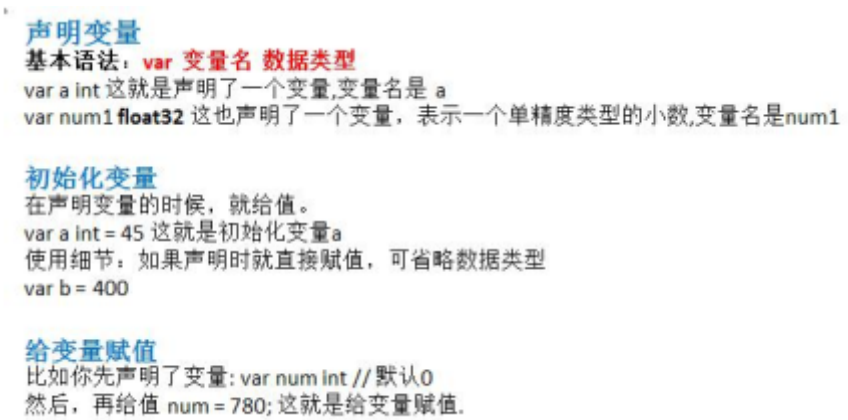
```
package main

import "fmt"

// 全局变量
// 在go中函数外部定义变量就是全局变量
var n1 = 100
var n2 = 200
var name = "jack"

// 可以一次性声明
var (
    n3    = 300
```

```
    n4    = 900
    name2 = "mary"
)

func main() {
    var n1, n2, n3 int
    fmt.Println("n1=", n1, "n2=", n2, "n3=", n3)

    var na1, name, na3 = 100, "tom", 688
    fmt.Println("na1=", na1, "name=", name, "na3=",
na3)

    ns1, name2, ns3 := 100, "tom", 678
    fmt.Println("ns1=", ns1, "name2=", name2,
"ns3=", ns3)

}
```

- 变量=变量名+值+数据类型
- Golang 的变量如果没有赋初值，编译器会使用默认值, 比如 int 默认值 0

  string 默认值为空串，小数默认为 0
- 变量在同一个作用域(在一个函数或者在代码块)内不能重名

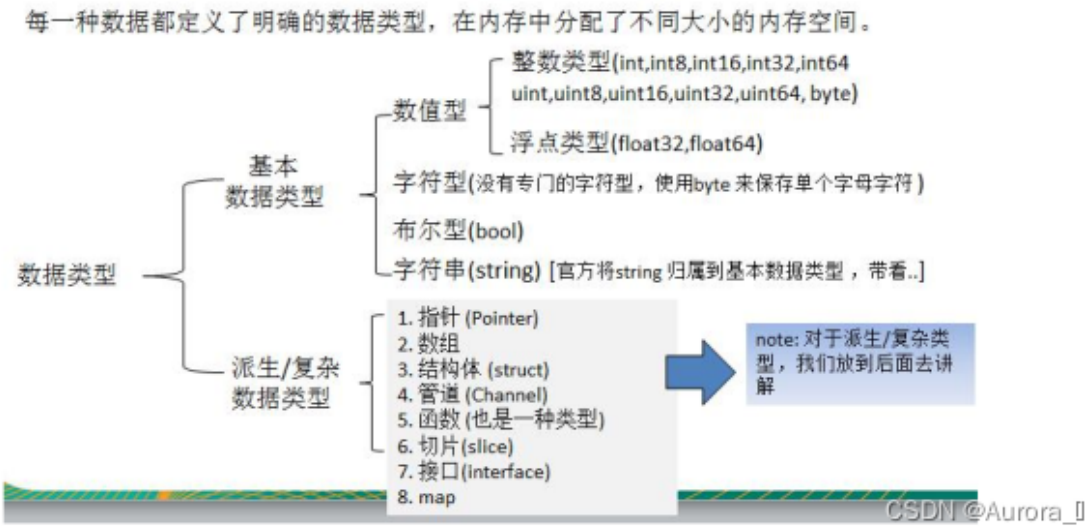
6、变量的声明，初始化和赋值



6、程序中 +号的使用

1. 当左右两边都是数值型时，则做加法运算
2. 当左右两边都是字符串，则做字符串拼接

三、数据类型的基本介绍



编辑

> **字符类型**
>
> Golang 中没有专门的字符类型，如果要存储单个字符(字母)，一般使用 byte 来保存。
> 字符串就是一串固定长度的字符连接起来的字符序列。
>
> Go 的字符串是由单个字节连接起来的。也就是说对于传统的字符串是由字符组成的，而 Go 的字符串不同，它是由字节组成的。
>
> **rune类型和int32一样哦！！**

1、基本数据类型的相互转换

Golang 和 java / c 不同，Go 在不同类型的变量之间赋值时需要显式转换。也就是说 Golang 中数
据类型不能自动转换。

2、值类型和引用类型

1、 值类型：基本数据类型 int 系列, float 系列, bool, string 、数组和结构体 struct

> 值类型：变量直接存储值，内存通常在栈中分配

2、引用类型：指针、slice 切片、map、管道 chan、interface 等都是引用类型

> 引用类型：变量存储的是一个地址，这个地址对应的空间才真正存储数据(值)，内存通常在堆上分配，当没有任何变量引用这个地址时，该地址对应的数据空间就成为一个垃圾，由 GC 来回收。

3.整型

1.整数的各个类型

1.int的有符号型

整型的类型

类　型	有无符号	占用存储空间	表数范围	备注
int8	有	1字节	-128 ~ 127	
int16	有	2字节	$-2^{15} \sim 2^{15}-1$	
int32	有	4字节	$-2^{31} \sim 2^{31}-1$	
int64	有	8字节	$-2^{63} \sim 2^{63}-1$	

2.int的无符号型

整型的类型

类 型	有无符号	占用存储空间	表数范围	备注
uint8	无	1字节	0～255	
uint16	无	2字节	$0 \sim 2^{16}-1$	
uint32	无	4字节	$0 \sim 2^{32}-1$	
uint64	无	8字节	$0 \sim 2^{64}-1$	

3.int的其他类型

类 型	有无符号	占用存储空间	表数范围	备注
int	有	32位系统4个字节 64位系统8个字节	$-2^{31} \sim 2^{31}-1$ $-2^{63} \sim 2^{63}-1$	
uint	无	32位系统4个字节 64位系统8个字节	$0 \sim 2^{32}-1$ $0 \sim 2^{64}-1$	
rune	有	与int32一样	$-2^{31} \sim 2^{31}-1$	等价 int32，表示一个 Unicode 码
byte	无	与 uint8 等价	0～255	当要存储字符时选用byte

2.整型的使用细节

1. Golang 各整数类型分：有符号和无符号，int uint 的大小和系统有关。
2. Golang 的整型默认声明为 int 型
3. 如何在程序查看某个变量的字节大小和数据类型 （使用较多）

```
var n2 int64 = 10
fmt.Printf("n2的类型是：%T ,n2占用的字节数是%d ",
n2, unsafe.Sizeof(n2))
```

4. Golang 程序中整型变量在使用时，遵守保小不保大的原则，即：在保证程序正确运行下，尽量使用占用空间小的数据类型。

3.动动小手，练习一下

```
package main

import (
```

```
    "fmt"
    "unsafe"
)

func main() {
    var i int = 1
    fmt.Println("i=", i)

    //测试int8的范围 -128~127
    var j int8 = -128
    fmt.Println("j=", j)

    var c byte = 'a'
    fmt.Println(c)

    //如何让查看一个变量的数据类型
    //fmt.Printf() 可以用作格式化输出
    var n1 = 100
    fmt.Printf("n1的类型是: %T \n", n1)

    //如何查看一个变量的占用的字节大小和数据类型
    var n2 int64 = 10
    fmt.Printf("n2的类型是: %T ,n2占用的字节数是%d ",
n2, unsafe.Sizeof(n2))

    //保小不保大

}
/*
输出:
i= 1
j= -128
97
n1的类型是: int
n2的类型是: int64 ,n2占用的字节数是8
*/
```

4.小数类型/浮点型

1.小数类型分类

类 型	占用存储空间	表数范围
单精度float32	4字节	-3.403E38 ~ 3.403E38
双精度float64	8字节	-1.798E308 ~ 1.798E308

> - 关于浮点数在机器中存放形式的简单说明,浮点数=符号位+指数位+尾数位
>   说明：浮点数都是有符号的.
> - 尾数部分可能丢失，造成精度损失。
> - float64 的精度比 float32 的要准确.
> - 浮点型的存储分为三部分：符号位+指数位+尾数位 在存储过程中，精度会有丢失

2.浮点型使用细节

1.Golang 浮点类型有固定的范围和字段长度，不受具体 OS(操作系统)的影响。

2. Golang 的浮点型默认声明为 float64 类型。
3. 浮点型常量有两种表示形式
   十进制数形式：如：5.12 .512(必须有小数点)
   科学计数法形式:如：5.1234e2 = 5.12 * 10 的 2 次方
   5.12E-2= 5.12/10 的 2 次方。

3.动动小手，练习一下

```
package main

import "fmt"

func main() {
```

```
    var price float32 = 89.12
    fmt.Println("price=", price)

    var num1 float32 = -0.00089
    var num2 float64 = -7809556.08
    fmt.Println("num1=", num1, "num2=", num2)

    var num3 float32 = -1233.3563454543
    var num4 float64 = -1233.7832532653
    fmt.Println("num3=", num3, "num4 =", num4)

    //golang的浮点型默认声明为float64
    var num5 = 1.1
    fmt.Printf("num5的数据类型是%T \n", num5)

    num6 := 5.12
    num7 := .123 //=>0.123
    fmt.Println("num6=", num6, "num7=", num7)
}
/*
输出:
price= 89.12
num1= -0.00089 num2= -7.80955608e+06
num3= -1233.3563 num4 = -1233.7832532653
num5的数据类型是float64
num6= 5.12 num7= 0.123
*/
```

5.字符类型

1.基本介绍

- Golang 中没有专门的字符类型，如果要存储单个字符(字母)，一般使用 byte 来保存。

- 字符串就是一串固定长度的字符连接起来的字符序列。Go 的字符串是由单个字节连接起来的。也就是说对于传统的字符串是由字符组成的，而 Go 的字符串不同，它是由字节组成的。

2.看代码详细了解

```
package main

import "fmt"

// golang中字符类型的使用
func main() {

    var c1 byte = 'a'
    var c2 byte = '0'

    fmt.Println("c1=", c1)
    fmt.Println("c2=", c2)
    //输出对应的字符
    fmt.Printf("c1=%c  c2=%c \n ", c1, c2)

    var c3 int = '北'
    fmt.Printf("c3=%c ", c3)
}


/*
输出:
c1= 97
c2= 48
c1=a  c2=0
c3=北
*/
```

> 对上面代码说明

> 1. 如果我们保存的字符在 ASCII 表的,比如[0-1, a-z,A-Z..]直接可以保存到 byte
> 2. 如果我们保存的字符对应码值大于 255,这时我们可以考虑使用 int 类型保存
> 3. 如果我们需要安装字符的方式输出，这时我们需要格式化输出，即 fmt.Printf(“%c”, c1)..

3.字符类型使用细节

1. 字符常量是用单引号('')括起来的单个字符。

   例如：var c1 byte = 'a'
   var c2 int = '中'
   var c3 byte = '9'

2. Go 中允许使用转义字符 '\'来将其后的字符转变为特殊字符型常量。

   例如： var c3 char= ‘\n’
   // '\n'表示换行符

3. Go 语言的字符使用 UTF-8 编码，如果想查询字符对应的 utf8 码值
   http://www.mytju.com/classcode/tools/encode_utf8.asp英文字母-1 个字节
   汉字-3 个字节

4. 在 Go 中，字符的本质是一个整数，直接输出时，是该字符对应的 UTF-8 编码的码值。

5. 可以直接给某个变量赋一个数字，然后按格式化输出时%c，会输出该数字对应的 unicode 字符。

6. 字符类型是可以进行运算的，相当于一个整数，因为它都对应有 Unicode 码.

4.字符类型本质探讨

1. 字符型 存储到 计算机中，需要将字符对应的码值（整数）找出来
   存储：字符--->对应码值---->二进制-->存储
   读取：二进制----> 码值 ----> 字符 --> 读取
2. 字符和码值的对应关系是通过字符编码表决定的(是规定好)
3. Go 语言的编码都统一成了 utf-8。非常的方便，很统一，再也没有编码乱码的困扰了

6.布尔类型

1. 布尔类型也叫 bool 类型，bool 类型数据只允许取值 true 和 false
2. bool 类型占 1 个字节。
3. bool 类型适于逻辑运算，一般用于程序流程控制[注：这个后面会详细介绍]：
   if 条件控制语句；
   for 循环控制语句