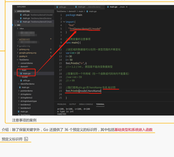
Golang 入门教程

- Golang 的开发前期的准
  - 应用领域
    - 区块链
    - 服务端/微代理
    - 分布式/云计算
      - Docker
      - K8s
  - Go 语言的编译和跨平台
    - 1. 编译可以直接跑个程
    - 2. 支持跨平台
    - 3. 每一个公式文件需要分开
    - 4. 它的 defer（结果倒叙，延迟执行）defer
- 跟流基础
  - Hello World
  - 语法基础
    - 变量
      - 关键字
      - 标识符
      - 变量
      - 字符串
      - 常量
    - 分隔符
      - Go 语言一行代表一个语句结束，不像 C 语言这一样以分号结束。
      - 如果一个语句太长需要换一行分隔，不推荐。
    - 字符串连接
      - Go 语言字符串连接通过 + 实现
      - 例如：fmt.Println("Google" + "Runoob")
    - 关键字
      - 关键字和保留字
      - 预定义标识符
    - Go 语言的空格
      - 1. 变量声明后没有空格报错，例如：var age int
      - 2. 函数中的左括号不可以换行否则会产生错误，推荐换行。
    - 1. 布尔类型bool
      - 基本介绍
        - 1. bool 类型占 1 个字节
        - 1. bool 类型默认值为false
      - 应用场景
        - 用于逻辑判断
        - for 循环控制条件
      - 测试案例
    - 2. 数字类型
      - int, float32, float64 ... Go 语言支持整型和浮点数字，并且支持复数，其中位运算采用补码。
      - 整数类型
        - uint8 无符号 8 位整型 (0 到 255)
        - uint16 无符号 16 位整型 (0 到 65535)
        - uint32 无符号 32 位整型 (0 到 4294967295)
        - uint64 无符号 64 位整型 (0 到 18446744073709551615)
        - int8 有符号 8 位整型 (-128 到 127)
        - int16 有符号 16 位整型 (-32768 到 32767)
        - int32 有符号 32 位整型 (-2147483648 到 2147483647)
        - int64 有符号 64 位整型 (-9223372036854775808 到 9223372036854775807)
      - 浮点类型
        - float32 IEEE-754 32位浮点数
        - float64 IEEE-754 64位浮点数
        - complex64 32 位实数虚数
        - complex128 64 位实数虚数
    - 3. 字符类型（string）
      - 字符串是一串固定长度的字符连接起来的字符序列。Go 的字符串是多个字节连接起来的。Go 语言的字符串按字节来看，它不是 Unicode 文本，Go 的字符串不等，它是由字节组成的。
      - 字符串
        - 例子
        - 如果要在字符串中转义字符，一般使用 Go 标准的，Unicode 或 Hex 转义序列。
        - 例子
        - 如果要输出包括回车符，需要使用反引号
        - 例子
        - 测试案例
      - 定义字符串
        - 如果要对字符串中的字符等，一般使用 byte 来操作。
        - 例子
        - 字符串一旦定义就不能改变
        - var str1 string = "123"
        - str1[1] = '2'
      - 注意事项（常用操作）
        - 1. Golang 字符串的字节是以 UT-8 编码的进行表示 Unicode 文本，这种golang 统一选用了 utf 8 变长，能能避免了中文乱码的问题
        - 1. 字符串的两种形式
          - 例如案例：
          - var str1 string = "123"
          - str1[1] = '2'
        - 2. 字符串一旦定义就不能改变
        - 1. 字符串的两种形式
        - 2. 字符串的拼接方式
        - 3. 字符串的拼接方式
        - 4. 字符串的拼接方式
        - 5. 字符串的拼接方式
        - 1. 单双引号
        - 2. 反引号
        - 3. 字符串的拼接方式
        - 4. 字符串的拼接方式
        - 1. 通过 + 加号拼接
        - 2. 通过 += 加号拼接
        - 3. 字符串的拼接方式 ，拼接方式有 + 加号和要拼接的内容换行
        - 测试案例
    - 4. 字符类型（char）
      - 1. 字符串中的字符可以单独的，例如 var c1 char = 'a', var c2 char = '中' var c3 char = 'c'
      - 2. Go 中允许使用转义字符 '\' 来将其后的字符转义为特殊字符类型常量。例如：var c3 = '\n' // '\n' 表示换行符
      - 3. Go 语言的字符串使用 UTF-8编码文本字符，这种字符可以
      - 4. 在 Go 中，字符的本质是一个整数，直接输出时，这是该字符对应的 UTF-8 编码。
      - 5. 可以通过赋值给字符变量或一个整数，然后按照格式化输出时，会输出该字符串对应的 unicode 字符
      - 6. 字符类型可以进行运算，相当于一个整数，因为他们都对应的 Unicode 码。
        - 1. 字符型 存储到计算机中，需要将字符对应的码值（整数）转成二进制
        - 例子
        - 字符 - 字符码 -> 对应的码值 -> 二进制 -> 存储
        - 例子
        - 读取 二进制 -> 码值 -> 字符 -> 使用
        - 2. 字符型存储的方式是基于字符的编码表来进行（最常用字符）
        - 2.Go 语言的编码采用一种扩展的 UTF-8 编码方案，兼容有效的 ASCII 字符
    - 5. 数据类型
      - byte 别名 uint8
      - rune 别名 int32
      - uint 32 或 64 位
      - int 与 uint 一样
      - uintptr 足够保存指针的无符号整型
    - 6. 基本数据类型的运算
      - 除法 +++ 0
      - 除法 +++ 0
      - 除数不能为0
      - 测试案例
    - 7.基本数据类型的转换
      - 例子 +++ 0
      - 例子 +++ 0
      - 例子 +++ 0
      - 布尔类型 false
      - 测试案例
    - 8. 基本数据类型的初始化
      - 介绍：Golang 和 Java/C 不同，Go 在不同类型的变量上初始化时有更**灵活的规则**，就是说Golang不初始化时的
      - 默认的工作机制，变量/对象一旦被定义，这个初始化的过程时的过程 是，比如：var pi= 3.14 + 8mum
      - 4. 获取指针的类型时指向的值，使用 *，比如：var *pint int, 使用 *pint 来获取 p 指向的变量。
        - var num int = 1
        - var i = 999
        - var p= *int = &i
        - func main() {
        - var a int = 100
        - var pb= *float32 = &a // 错误，正确方式 fa
        - }
        - func main() {
        - var a int = 100
        - var b int = 200
        - var c int = 300
        - var fa int = 400
        - var fb int = 500
        - *pa = 100 // a = 100
        - *pb = 200 // b = 200
        - fmt.Println("a=%d b=%d ",*pa,*pb) // a=100 b=200
        - }
        - // 输出结果
        - a=100, b=200, *pa = 100
        - 值类型，都有对应的指针类型，形式为 *数据类型，比如 int 的指针类型是 *int, float32 的指针的类型是 *float ，以此类推。
        - 值类型比较，基本数据类型 int 系列，bool, string, 数组和结构体 struct
      - 引用数据（指针，slice 切片，map ，管道 chan ，interface 等都是引用类型
      - 值类型
        - 1. 值类型，变量直接存储值，通常在栈中分配
        - 1. 引用类型，变量存储的是一个地址，这个地址存的才是最正存储数据（值）。内存通常在堆上分配，当没有任何变量引用这个地址时，该地址的数据才回被成为一个垃圾，由 GC 来回收
      - Golang 程序中变量，可分，这都是有在程序的不同作用域和生命周期。
        - 几个重要的可以或者说的内部作用域的说明
        - 1. 函 内 个其它字母大小写，小 为，组成
        - 2. 函数可以公开
        - 3. Golang 中严格区分大小写
        - 4. 标识符不能包含空格
        - 5. 下划线 "_" 本是匿 Go 中是一个特殊的标识符，或为空标识符，可以代表任何其他的标识符，但是它对包括声明在内的任何用法，其行为都是惟一的。例如，如果函数返回一个整数和一个布尔值，但是，程序中只需要整数值，那么可以把布尔值赋给 "_"。
        - 6. 不能以数字开头，比如：var a 2 是
        - 在Golang 中为了简化编译速度以提高可读性，规定了这种强制要求**规定**的语法 ++
        - 表示程序结束的语句
      - 标识符的命名规则
        - 1. 包括，保留package的文件名称及的文件夹的第一条，并要求用有意义的名称，例如，有意义，不要和标准库冲突
        - 1. 1. 变量名，函数名，常量名，采用驼峰命名。
        - 1. 如果变量名，函数名，名两个字母大写，则可以被其他的包访问，如果要导出小写，则只能在本包内使用（注意：可以使用结构体，或字母大写的公用的，或字母小写的私有的）。
        - 
        - 注意事项的说明
      - 预定义标识符
        - 介绍：除了保留关键字外，Go 还提供了 36 个预定义的标识符，其中包括基础类型和内置函数。
        - 预定义标识符
  - 变量
    - 变量的定义
      - 声明变量
        - 第一种，指定变量类型，如果没有初始值，则变量默认为零值。 var
        - 第二种，根据值的声明变量类型。 var
        - 第三种，**简短** var 注意：在简短声明左侧声明的变量，跟声明新变量，格式：
      - 多个变量声明
      - 并行赋
    - 常量
      - 常量是用来表示固定的，在运行时，不会修改的数据
      - 常量声明语句的格式同样也是用 const 关键字，常量的类型是 (标识符和表达式
      - const 关键字
        - 例如：const b string = "abc" 或者 const b string = "abc"
        - iota，特殊常量，可以认为是一个可以被编译器修改的常量。
        - iota 在 const 关键字出现时将重置为 0(const 内部的第一行之前)，const 中每新增一行常量声明将使 iota 计数一次(iota 可理解为 const 语句块中的行索引)。
        - 代码示例
      - iota 关键字
    - 函数
      - 函数是基本的代码块，用于执行一个任务。
      - Go 语言至少有个 main() 函数。
      - 你可以通过函数来划分不同功能，逻辑上每个函数执行的是指定的任务。
      - 函数声明 (函数签名) 告诉了编译器函数的名称，返回类型，和参数。
      - Go 语言标准库提供了多种可动用的内置的函数。 例如，len() 函数可以接受不同类型参数并返回该类型的长度。如果给的是字符串则返回字符串的长度，如果传入是数组，则返回数组中包含的元素个数。
      - 函数定义如下：
      - func function_name (parameter list) [return_types] {
      - 函数体
      - }
      - 简单函数调用
      - 函数调用多个值
      - 函数参数
        - 值传递是指在调用函数时将实际参数复制一份传递到函数中，这样在函数中如果对参数进行修改，将不会影响到实际参数。
        - 例子
        - 引用传递是指在调用函数时将实际参数的地址传递到函数中，那么在函数中对参数所进行的修改，将影响到实际参数。
        - 例子
      - 函数作为一种另一个函数的实参
        - 定义：函数定义好后作为另一个函数的实参数传入
        - 例子
      - 函数闭包
        - 闭包：闭包是匿名函数，可以动态编程中使用。
        - 例子
        - 定义：方法就是一个包括了接受者的函数。
        - 例子
      - 方法
      - 函数型：基本数据类型 int 系列，bool, string，数组和结构体 struct
    - 程序控制结构和语句
      - 引用数据（指针，slice 切片，map ，管道 chan ，interface 等都是引用类型
      - 值类型
        - 1. 值类型，变量直接存储值，通常在栈中分配
        - 1. 引用类型，变量存储的是一个地址，这个地址存的才是最正存储数据（值）。内存通常在堆上分配，当没有任何变量引用这个地址时，该地址的数据才回被成为一个垃圾，由 GC 来回收
      - Golang 程序中变量，可分，这都是有在程序的不同作用域和生命周期。
        - 几个重要的可以或者说的内部作用域的说明
        - 1. 函 内 个其它字母大小写，小 为，组成
        - 2. 函数可以公开
        - 3. Golang 中严格区分大小写
        - 4. 标识符不能包含空格
        - 5. 下划线 "_" 本是匿 Go 中是一个特殊的标识符，或为空标识符，可以代表任何其他的标识符，但是它对包括声明在内的任何用法，其行为都是惟一的。例如，如果函数返回一个整数和一个布尔值，但是，程序中只需要整数值，那么可以把布尔值赋给 "_"。
        - 6. 不能以数字开头，比如：var a 2 是
        - 在Golang 中为了简化编译速度以提高可读性，规定了这种强制要求**规定**的语法 ++
        - 表示程序结束的语句
      - 标识符的命名规则
        - 1. 包括，保留package的文件名称及的文件夹的第一条，并要求用有意义的名称，例如，有意义，不要和标准库冲突
        - 1. 1. 变量名，函数名，常量名，采用驼峰命名。
        - 1. 如果变量名，函数名，名两个字母大写，则可以被其他的包访问，如果要导出小写，则只能在本包内使用（注意：可以使用结构体，或字母大写的公用的，或字母小写的私有的）。
        -
        - 注意事项的说明
      - 预定义标识符
        - 介绍：除了保留关键字外，Go 还提供了 36 个预定义的标识符，其中包括基础类型和内置函数。
        - 预定义标识符
  - 变量
    - 数据运算符类型**数值类型的变量**的计运算符，比如，加减乘除，取Go中的运算符类型
    - 算术运算符
      - 数据运算符

运算符	说明	实例	结果
+	正号	+5	5
	负号	-4	-4
+	加	5 + 5	10
	减	8 - 6	2
*	乘	3 * 4	12
/	除	5 / 5	1
%	取模(余数)	7 % 5	2
++	自增	a=2 a++	a=3
--	自减	a=3 a--	a=2
+	字符串相加	"he" + "llo"	"hello"

      - 数据运算符
        - 介绍：数据运算符是对**数值类型的变量**进行赋值的一种运算符，例如：将某个值赋给某个变量，或者把一个表达式的结果赋给一个变量。
        - 赋值表达式
        - var n1 float32 = 10 / 4
        - fmt.Printf(n1) //2
        - //如果我们希望保留小数，那么需要将其中的数字改成浮点数
        - var n1 float32 = 10.0 / 4
        - fmt.Printf(n1) // 2.5
        - 二元运 为 赋值表
        - 例子 +++ 0
        - 例子 +++ 0
        - 例子 +++ 0

运算符

1. 算术运算符

运算符	描述	实例
+	加	两个对象相加	a + b 输出结果 30
-	减	得到负数或是一个数减去另一个数	a - b 输出结果 -10
*	乘	两个数相乘或是返回一个被重复若干次后的字符串	a * b 输出结果 200
/	除	b / a 输出结果 2
%	模除	b % a 输出结果 0
**	幂	返回x的y次幂	a**b 为 10 的 20 次方
//	取整除	向下取接近除数的整数	9//2 输出结果 4 , -9//2 输出结果 -5

2. 比较运算符

运算符	描述	实例
==	等于 - 比较对象是否相等	(a == b) 返回 False。
!=	不等于 - 比较两个对象是否不相等	(a != b) 返回 True。
<>	不等于 - 比较两个对象是否不相等。在 3.x 中被弃用	(a <> b) 返回 True。这个运算符类似 != 。
>	大于 - 返回 x 是否大于 y	(a > b) 返回 False。
<	小于 - 返回 x 是否小于 y。所有比较运算符返回 1 表示真，返回 0 表示假。这分别对应于特殊的内建变量 True 和 False。	(a < b) 返回 True。
>=	大于等于 - 返回 x 是否大于等于 y。	(a >= b) 返回 False。
<=	小于等于 - 返回 x 是否小于等于 y。	(a <= b) 返回 True。

3. 赋值运算符

运算符	描述	实例
=	简单的赋值运算符，将一个表达式的赋值给一个左侧的变量	c = a + b 将 a + b 表达式计算结果赋值 c
+=	加法赋值	c += a 等于 c = c + a
-=	减法赋值	c -= a 等于 c = c - a
*=	乘法赋值	c *= a 等于 c = c * a
/=	除法赋值	c /= a 等于 c = c / a
%=	模赋值	c %= a 等于 c = c % a
**=	幂赋值	c **= a 等于 c = c ** a
//=	取整除赋值	c //= a 等于 c = c // a
&=	按位与赋值	c &= a 等于 c = c & a
|=	按位或赋值	c |= a 等于 c = c | a
^=	按位异或赋值	c ^= a 等于 c = c ^ a
>>=	右移赋值	c >>= a 等于 c = c >> a
<<=	左移赋值	c <<= a 等于 c = c << a

4. 逻辑运算符

运算符	描述
and	逻辑与运算符，语法格式为：a and b 。如果 a 为 False，则表达式结果为 False，否则表达式结果为 b 的计算值。
or	逻辑或运算符，语法格式为：a or b 。如果 a 是非零数值，则表达式结果为 a 的值，否则表达式结果为 b 的计算值。
not	逻辑非运算符，语法格式为：not a 。如果 a 为 True，则表达式结果为 False。如果 a 为 False，则表达式结果为 True。

5. 成员运算符

运算符	描述
in	在指定的序列中找到值时返回 True，否则返回 False。
not in	在指定的序列中没有找到值时返回 True，否则返回 False。

6. 位运算符

运算符	描述	实例
&	按位与运算符"&"是双目运算符。其功能是参与运算的两数各对应的二进位相与。	(a & b) 输出结果 12，二进制为 0000 1100
|	按位或运算符"|"是双目运算符。其功能是参与运算的两数各对应的二进位相或。	(a | b) 输出结果 61，二进制为 0011 1101
^	按位异或运算符"^"是双目运算。其功能是参与运算的两数各对应的二进位相异或，当两对应的二进位相异时，结果为1。	(a ^ b) 输出结果 49，二进制为 0011 0001
~	按位取反运算符"~"是双目运算符。按照位就是对位的反方，其功能是对参与运算的数的各二进位按照1变0，0变1的原则进行取反。	(~a) 输出结果 -61，二进制为 1111 0000
<<	左移动运算符"<<"是双目运算。左移动运算是将<<左边的运算数的各二进位全部左移若干位，由<<右边的数目决定。高位丢弃，低位补0。	a << 2 输出结果 240，二进制为 1111 0000
>>	右移动算符">>"是双目运算。右移动运算是将>>左边运算数的各二进位全部右移若干位，由>>右边的数目决定。数目是一个正数值。丢弃最低位的位数，剩下的数字向右移动。	a >> 2 输出结果 15，二进制为 0000 1111

位运算符一览表

7. 身份运算符

运算符	描述	实例
is	是对变量内存地址的比较	a is b，判断两个变量的实际地址。
is not	是对变量内存地址的比较	a is not b，是一个否定变量。

身份运算符列表

Go 语言特殊不支持三元运算符

8. 字符串

字符串 ⇒

9. 数组

字符串 ⇒

10. 切片

字符串 ⇒

11. 位运算

字符串 ⇒

12. 程序流程控制

13. 函数

14. 指针

15. 结构体

16. 接口

17. 并发

18. 包的基本概念

19. 数据库操作

20. Golang 数据操作

- MySQL 数据库操作
- Redis 操作