Go语言基本语法——变量及常量

目录

一、变量

（一）、变量的概念

- 变量是计算机语言中储存数据的抽象概念。变量的功能是存储数据。变量通过变量名访问；
- 变量的本质是计算机分配的一小块内存，专门用于存放指定数据，在程序运行过程中该数值可以发生改变；
- 变量的存储往往具有瞬时性，或者说是临时存储，当程序运行结束，存放该数据的内存就会释放，而该变量就会消失；
- Go 语言的变量名由字母、数字、下划线组成，首个字符不能为数字;
- Go语法规定，定义的局部变量若没有被调用则编译错误。

（二）、声明变量

- 变量声明有多种形式

1、未初始化的标准格式

- var 变量名 变量类型

2、未初始化的批量格式

- 不用每行都用var申明

```
var (
    a  int
    b string
    c []float32
    d func() bool
    e struct {
        x int
        y string
    }
)
```

- 未初始化变量的默认值：
  - 整形和浮点型变量默认值：0
  - 字符串默认值为空字符串
  - 布尔型默认值为false
  - 函数、指针变量初始值为nil

3、初始化变量的标准格式

- var 变量名 类型 = 表达式

4、初始化变量的编译器自动推断类型格式

- var 变量名 = 表达式

5、初始化变量的简短声明格式（短变量声明格式）

- 变量名 := 表达式
- 使用 := 赋值操作符，:= 可以高效地创建一个新的变量，称之为初始化声明。
- 声明语句省略了var 关键字
- 变量类型将由编译器自动推断
- 这是声明变量的首选形式，但是它只能被用在函数体内，而不可以用于全局变量的声明与赋值
- 该变量名必须是没有定义过的变量，若定义过，将发生编译错误
- 在多个短变量声明和赋值中，至少有一个新声明的变量出现在左侧中，那么即便有其它变量名可能是重复声明的，编译器也不会报错。

（三）、变量多重赋值（多个变量同时赋值）

- Go语法中，变量初始化和变量赋值是两个不同的概念。Go语言的变量赋值与其他语言一样，但是Go提供了其他程序员期待已久的多重赋值功能，可以实现变量交换。多重赋值让Go语言比其他语言减少了代码量。

（四）、匿名变量

- Go语言的函数可以返回多个值，而事实上我们并不是对所有的返回值都用得上。那么就可以使用匿名变量，用“_”下划线替换即可。
- 匿名变量不占用命名空间，不会分配内存。

二、数据类型

- 基本数据类型（原生数据类型）：整型、浮点型、布尔型、字符串、字符（byte、rune）
- 复合数据类型（派生数据类型）：指针（pointer）、数组（array）、切片（slice）、映射（map）、函数（function）、结构体（struct）、通道（channel）

（一）、整型

- 整型分两大类
  - 按长度分：int8、int16、int32、int64、int
  - 无符号整型：uint8、uint16、uint32、uint64、uint
  - 其中uint8就是byte型，int16对应C语言的short型，int64对应C语言的long型。

序号	类型和描述
1	uint8 无符号 8 位整型 (0 到 255) 【2的8次方】
2	uint16 无符号 16 位整型 (0 到 65535) 【2的16次方】
3	uint32 无符号 32 位整型 (0 到 4294967295) 【2的32次方】
4	uint64 无符号 64 位整型 (0 到 18446744073709551615) 【2的64次方】
5	int8 有符号 8 位整型 (-128 到 127)

还有其他数字类型

序号	类型和描述
1	byte 类似 uint8
2	rune 类似 int32
3	uint 32 或 64 位
4	int 与 uint 一样大小
5	uintptr 无符号整型，用于存放一个指针

（二）、浮点型

- Go语言支持4种浮点型数：float32、float64、complex64（32 位实数和虚数）、complex128（64 位实数和虚数）
- float32的最大范围是3.4e38，用常量定义是：math.MaxFloat32
- float64的最大范围是1.8e308，用常量定义是：math.MaxFloat64

（三）、布尔型

- 声明方式：var flag bool
- 布尔型无法参与数值运算，也无法与其他类型进行转换。

（四）、字符串

- 字符串在Go语言中是以基本数据类型出现的，使用字符串就像使用其他原生基本数据类型int、float32、float64、bool一样。

- 字符串中可以使用转移符
  - \r 回车符return，返回行首
  - \n 换行符new line，直接跳到下一行的同列位置
  - \t 制表符TAB
  - \' 单引号
  - \" 双引号
  - \\ 反斜杠
- 定义多行字符串
  - 双引号书写字符串被称为字符串字面量（string literal），这种字面量不能跨行；
  - 多行字符串需要使用“`”反引号，多用于内嵌源码和内嵌数据；
  - 在反引号中的所有代码不会被编译器识别，而只是作为字符串的一部分。

（五）、字符

字符串中的每一个元素叫做“字符”，定义字符时使用单引号。Go语言的字符有两种：

1、byte型：其实是uint8的别名。代表了一个ASCII码的一个字符

2、rune型：其实就是int32。代表一个UTF-8字符。当需要处理中文等unicode字符集时需要用到rune类型。

  - var a byte = 'a'
  - var b rune = '—'

三、打印格式化

（一）、通用

- %v　　值的默认格式表示　value
- %+v　类似%v，但输出结构体时会添加字段名
- %#v　值的Go语法表示
- %T　　值的类型的Go语法表示　type

（二）、布尔值

- %t　　单词true或false　true

（三）整数

- %b　表示为二进制　binary
- %c　该值对应的unicode码值　char
- %d　表示为十进制　digital
- %8d　表示该整型长度是8，不足8则在数值前补空格。如果超出8，则以实际为准。
- %08d　数字长度是8，不足8位的，在数字前补0。如果超出8，则以实际为准。
- %o　表示为八进制　octal
- %q　该值对应的单引号括起来的go语法字符字面值，必要时会采用安全的转义表示　quotation
- %x　表示为十六进制，使用a-f　hex
- %X　表示为十六进制，使用A-F
- %U　表示为Unicode格式：U+1234，等价于"U+%04X"　unicode

（四）、浮点数与复数的两个组分

- %b　无小数部分、二进制指数的科学计数法，如-123456p-78；参见strconv.FormatFloat
- %e　（=%.6e）有6位小数部分的科学计数法，如-1234.456e+78
- %E　科学计数法，如-1234.456E+78
- %f　（=%.6f）有6位小数部分，如123.456123　float
- %F　等价于%f
- %g　根据实际情况采用%e或%f格式（以获得更简洁、准确的输出）
- %G　根据实际情况采用%E或%F格式（以获得更简洁、准确的输出）

（五）、字符串和[]byte

- %s　直接输出字符串或者[]byte　string
- %q　该值对应的双引号括起来的go语法字符串字面值，必要时会采用安全的转义表示
- %x　每个字节用两字符十六进制数表示（使用a-f）
- %X　每个字节用两字符十六进制数表示（使用A-F）

（六）、指针

- %p　　表示为十六进制，并加上前导的0x　pointer
- 没有%u。整数如果是无符号类型自然输出也是无符号的。类似的，也没有必要指定操作数的尺寸（int8，int64）。
- 宽度通过一个紧跟在百分号后面的十进制数指定，如果未指定宽度，则表示值时除必需之外不作填充。精度通过（可选的）宽度后跟点号后跟的十进制数指定。如果未指定精度，会使用默认精度；如果点号后没有跟数字，表示精度为0。举例如下：
- %f:　默认宽度，默认精度
- %9f　宽度9，默认精度
- %.2f　默认宽度，精度2
- %9.2f　宽度9，精度2
- %9.f　宽度9，精度0

（七）、其它flag

- '+'　　总是输出数值的正负号；对%q（%+q）会生成全部是ASCII字符的输出（通过转义）；
- ' ' 对数值，正数前加空格而负数前加负号；
- '-' 在输出右边填充空白而不是默认的左边（即从默认的右对齐切换为左对齐）；
- '#'　　切换格式：
- 八进制数前加0（%#o），十六进制数前加0x（%#x）或0X（%#X），指针去掉前面的0x（%#p）；
- 对%q（%#q），如果strconv.CanBackquote返回真会输出反引号括起来的未转义字符串；
- 对%U（%#U），输出Unicode格式后，如字符可打印，还会输出空格和单引号括起来的go字面值；
- 对字符串采用%x或%X时（% x或% X）会给各打印的字节之间加空格；
- '0'　　使用0而不是空格填充，对于数值类型会把填充的0放在正负号后面；

四、数据类型转换

（一）、数据类型转换的格式

1、T(表达式)

- 采用数据类型前置加括号的方式进行类型转换。T表示要转换的类型；表达式包括变量、数值、函数返回值等。
- 类型转换时，需要考虑两种类型之间的关系和范围，是否会发生数值截断。
- 布尔型无法与其他类型进行转换。

2、float与int之间转换

- 需要注意float转int时精度的损失

3、int转string

- 其实相当于是byte或rune转string。
- 该int数值是ASCII码的编号或Unicode字符集的编号。转成string就是将根据字符集，将对应编号的字符查找出来。
- 当该数值超出Unicode编号范围，则转成的字符串显示为乱码。
- 例如19968转string，就是“一”。

【备注：】

- ASCII字符集中数字的10进制范围是[30 - 39]
- ASCII字符集中大写字母的10进制范围是[65 - 90]
- ASCII字符集中小写字母的10进制范围是[97 - 122]
- Unicode字符集中汉字的范围是[4e00-9fa5]，10进制范围是[19968 - 40869]

4、string转int

- 不允许字符串转int（cannot convert 变量 (type string) to type int）

二进制	十进制	十六进制	缩写	解释	二进制	十进制	十六进制	字符	二进制	十进制	十六进制	字符	二进制	十进制	十六进制	字符
0000 0000	0	0	NUL	空字符(Null)	0010 0000	32	20	空格	1	65	41	A	0110 0001	97	61	a
0000 0001	1	1	SOH	标题开始	0010 0001	33	21	!	0	66	42	B	0110 0010	98	62	b
0000 0010	2	2	STX	正文开始	0010 0010	34	22	"	1	67	43	C	0110 0011	99	63	c
0000 0011	3	3	ETX	正文结束	0010 0011	35	23	#	0	68	44	D	0110 0100	100	64	d
0000 0100	4	4	EOT	传输结束	0010 0100	36	24	$	1	69	45	E	0110 0101	101	65	e
	5	5	ENQ	请求	0010 0101	37	25	%	0	70	46	F	0110 0110	102	66	f
0000 0110	6	6	ACK	收到通知	0010 0110	38	26	&	1	71	47	G	0110 0111	103	67	g
0000 0111	7	7	BEL	响铃	0010 0111	39	27	'	0	72	48	H	0110 1000	104	68	h
0000 1000	8	8	BS	退格	0010 1000	40	28	(	1	73	49	I	0110 1001	105	69	i
0000 1001	9	9	HT	水平制表符	0010 1001	41	29	)	0	74	4A	J	0110 1010	106	6A	j
0000 1010	10	0A	LF	换行键	0010 1010	42	2A	*	1	75	4B	K	0110 1011	107	6B	k
0000 1011	11	0B	VT	垂直制表符	0010 1011	43	2B	+	0	76	4C	L	0110 1100	108	6C	l
0000 1100	12	0C	FF	换页键	0010 1100	44	2C	,	1	77	4D	M	0110 1101	109	6D	m
0000 1101	13	0D	CR	回车键	0010 1101	45	2D	-	0	78	4E	N	0110 1110	110	6E	n
0000 1110	14	0E	SO	不用切换	0010 1110	46	2E	.	1	79	4F	O	0110 1111	111	6F	o
0000 1111	15	0F	SI	启用切换	0010 1111	47	2F	/	0	80	50	P	0111 0000	112	70	p
0001 0000	16	10	DLE	数据链路转	0011 0000	48	30	0	1	81	51	Q	0111 0001	113	71	q
0001 0001	17	11	DC1	设备控制1	0011 0001	49	31	1	0	82	52	R	0111 0010	114	72	r
0001 0010	18	12	DC2	设备控制2	0011 0010	50	32	2	1	83	53	S	0111 0011	115	73	s
0001 0011	19	13	DC3	设备控制3	0011 0011	51	33	3	0	84	54	T	0111 0100	116	74	t
0001 0100	20	14	DC4	设备控制4	0011 0100	52	34	4	1	85	55	U	0111 0101	117	75	u
0001 0101	21	15	NAK	拒绝接收	0011 0101	53	35	5	0	86	56	V	0111 0110	118	76	v
0001 0110	22	16	SYN	同步空闲	0011 0110	54	36	6	1	87	57	W	0111 0111	119	77	w
0001 0111	23	17	ETB	传输块结束	0011 0111	55	37	7	0	88	58	X	0111 1000	120	78	x
0001 1000	24	18	CAN	取消	0011 1000	56	38	8	1	89	59	Y	0111 1001	121	79	y
0001 1001	25	19	EM	介质中断	0011 1001	57	39	9	0	90	5A	Z	0111 1010	122	7A	z
0001 1010	26	1A	SUB	替补	0011 1010	58	3A	:	1	91	5B	[	0111 1011	123	7B	{
0001 1011	27	1B	ESC	溢出	0011 1011	59	3B	;	0	92	5C	\	0111 1100	124	7C	|
0001 1100	28	1C	FS	文件分割符	0011 1100	60	3C	<	1	93	5D	]	0111 1101	125	7D	}
0001 1101	29	1D	GS	分组符	0011 1101	61	3D	=	0	94	5E	^	0111 1110	126	7E	~
0001 1110	30	1E	RS	记录分离符	0011 1110	62	3E	>	1	95	5F	_				
0001 1111	31	1F	US	单元分隔符	0011 1111	63	3F	?	0	96	60	`				
0111 1111	127	7F	DEL	删除	0100 0000	64	40	@								

五、常量

(一)、声明方式

1、相对于变量，常量是恒定不变的值，例如圆周率。

- 常量是一个简单值的标识符，在程序运行时，不会被修改。

2、常量中的数据类型只可以是布尔型、数字型（整数型、浮点型和复数）和字符串型。

3、常量的定义格式：

- const 标识符 [类型] = 值
- 可以省略类型说明符 [type]，因为编译器可以根据变量的值来自动推断其类型。
    - 显式类型定义： const B string = "Steven"
    - 隐式类型定义： const C = "Steven"

4、多个相同类型的声明可以简写为：

- const WIDTH , HEIGHT = value1, value2

5、常量定义后未被使用，不会在编译时出错。

(二)、常量用于枚举（常量组）

- 例如以下格式：

```
const (
    Unknown = 0
    Female = 1
    Male = 2
)
```

数字 0、1 和 2 分别代表未知性别、女性和男性。

- 常量组中如果不指定类型和初始值，则与上一行非空常量的值相同。

```
const (
    a = 10
    b
    c
)
```

打印a、b、c，输出：10 10 10

(三)、iota

1、iota，特殊常量值，是一个系统定义的可以被编译器修改的常量值。iota只能用在常量赋值中。

2、在每一个const关键字出现时，被重置为0，然后每出现一个常量，iota所代表的数值会自动增加1。iota可以理解成常量组中常量的计数器，不论该常量的值是什么，只要有一个常量，那么iota就加1。

3、iota 可以被用作枚举值：

```
const (
    a = iota
    b = iota
    c = iota
)
println(a, b, c)
```

- 打印输出：0 1 2
- 第一个 iota 等于 0，每当 iota 在新的一行被使用时，它的值都会自动加 1；所以 a=0, b=1, c=2

4、常量组中如果不指定类型和初始值，则与上一行非空常量的值相同。所以上述

的枚举可以简写为如下形式：

```
const (
    a = iota
    b
    c
)
println(a, b, c)
```

打印输出：0 1 2

5、示例一

```
const (
    i = 1<<iota
    j = 3<<iota
    k
    l
)

func main() {
    fmt.Println("i=",i)
    fmt.Println("j=",j)
    fmt.Println("k=",k)
    fmt.Println("l=",l)
}
```

- 打印输出结果：
  - i= 1
  - j= 6
  - k= 12
  - l= 24

6、示例二

```
const (
      a1 = '—'
      b1
      c1 = iota
```

```
        d1
)
func main() {
    fmt.Println(a1, b1, c1, d1)
}
```

- 打印输出结果：
  - 19968 19968 2 3

六、类型别名（Type Alias）

（一）、概要

类型别名是Go1.9版本添加的新功能。主要用于代码升级、迁移中类型的兼容性问题。

在Go1.9版本前内建类型定义的代码是：

- type byte uint8
- type rune int32

而在Go1.9版本之后变更为：

- type byte = uint8
- type rune = int32

（二）、类型别名与类型定义

1、类型别名的语法格式：

- type 类型别名 = 类型

2、定义类型的语法格式：

- type 新的类型名 类型

例如：

- type NewString string

该语句是将NewString定义为string类型。通过type关键字，NewString会形成一种新的类型。NewString本身依然具备string的特性。

- type StringAlias = string

该语句是将StringAlias定义为string的一个别名。使用StringAlias与string等

效。别名类型只会在代码中存在，编译完成时，不会有别名类型。

（三）、非本地类型不能定义方法
不能为不在同一个包中的类型定义方法。
package main
improt "time"

七、出于性能考虑的最佳实践和建议

1. 尽可能的使用 := 去初始化声明一个变量（在函数内部）；
2. 尽可能的使用字符代替字符串；
3. 尽可能的使用切片代替数组；
4. 尽可能的使用数组和切片代替map；
5. 如果只想获取切片中某项值，不需要值的索引，尽可能的使用for range去遍历切片，这比必须查询切片中的每个元素要快一些；
6. 当数组元素是稀疏的（例如有很多0值或者空值nil），使用map会降低内存消耗；
7. 初始化map时指定其容量；
8. 当定义一个方法时，使用指针类型作为方法的接收者；
9. 在代码中使用常量或者标志提取常量的值；
10. 尽可能在需要分配大量内存时使用缓存；
11. 使用缓存模板。