0x01 运算符简介

运算符是指程序运行时执行的数学或者逻辑运算的符号。Golang中一个表达式中可以有多个运算符

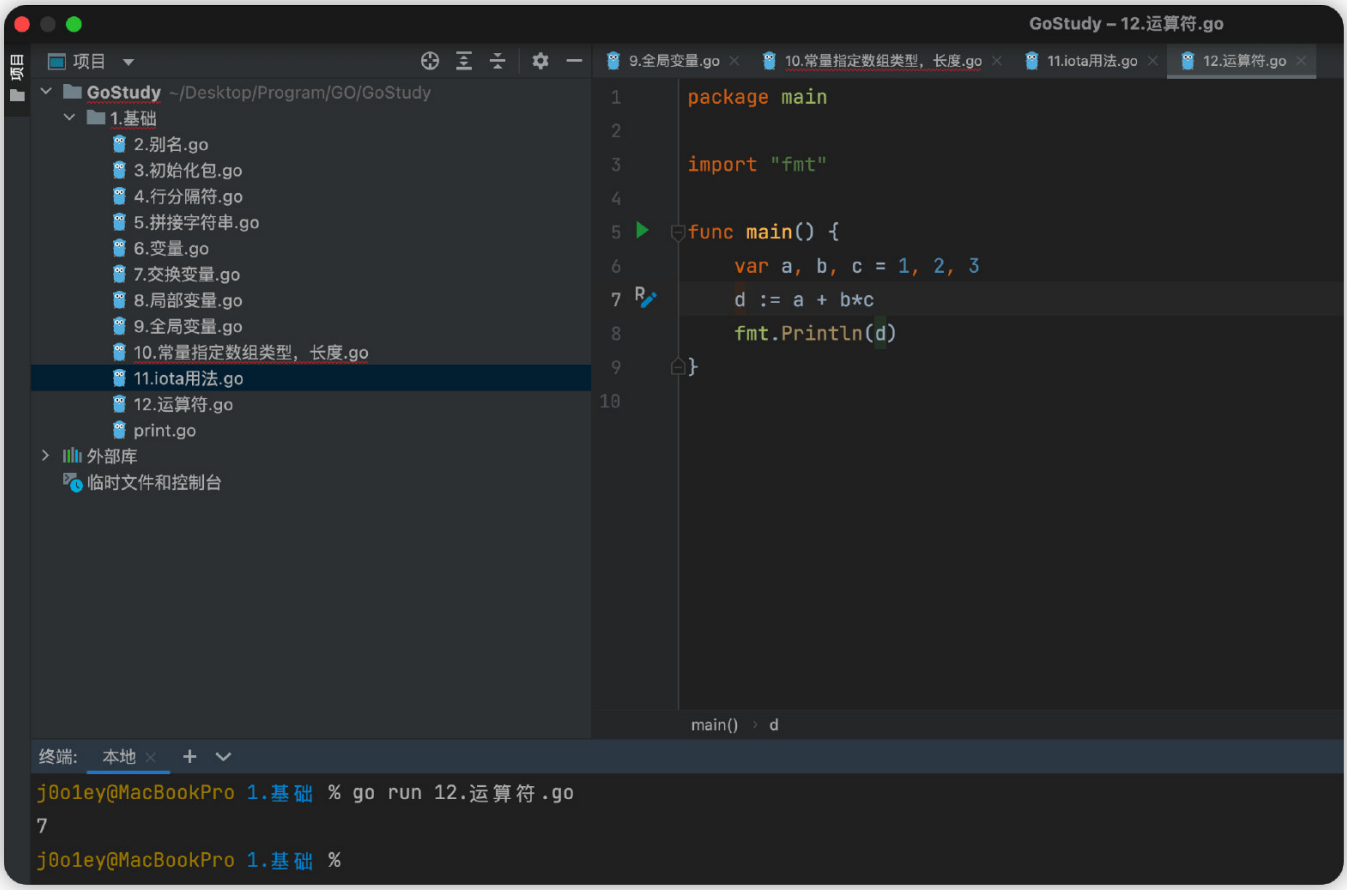
如果存在多个表达式，最会按照优先级来决定先执行哪一个表达式

示例

```
package main

import "fmt"

func main() {
  var a, b, c = 1, 2, 3
  d := a + b * c    // *优先级高于+
  fmt.Println(d)
}
```



0x02 Golang运算符及优先级

摘录自https://www.sojson.com/operation/go.html

Tips:加括号的()最优先，如果有多个括号，内层括号最优先

一、Go语言 算术运算符

下表显示了Go语言支持的所有算术运算符。假设变量 `A` 的值为 `10`，变量 `B` 的值为 `20`，则：

运算符	描述	示例
`+`	添加两个操作数	A+B=30
`-`	从第一个操作数中减去第二个操作数	A-B=10
`*`	将两个操作数相乘	A*B=200
`/`	将分子除以分母	B/A=2
`%`	模数运算符，以及整数除法的余数	B%A=0
`++`	增加(递增)运算符，将整数值加一	A++=11
`--`	相减(递减)运算符，将整数值减一	A—=9

二、Go语言 关系运算符

下表显示了Go语言支持的所有关系运算符。假设变量 `A` 的值为 `10`，变量 `B` 的值为 `20`，则：

运算符	描述	示例
`==`	检查两个操作数的值是否相等，如果相等，则条件为真。	(A==B)结果为假
`!=`	检查两个操作数的值是否相等，如果值不相等，则条件为真。	(A!=B)结果为真
`>`	检查左操作数的值是否大于右操作数的值，如果是，则条件为真。	(A>B)结果为假
`<`	检查左操作数的值是否小于右操作数的值，如果是，则条件为真。	(A<B)结果为真
`>=`	检查左操作数的值是否大于或等于右操作数的值，如果是，则条件为真。	(A>=B)结果为假
`<=`	检查左操作数的值是否小于或等于右操作数的值，如果是，则条件为真。	(A<=B)结果为真

三、Go语言 逻辑运算符

下表显示了Go语言支持的所有逻辑运算符。假设变量`A`的值为`1`，变量`B`的值为`0`，则：

运算符	描述	示例
`&&`	逻辑`AND`运算符。如果两个操作数都不为零，则条件为真。	(A&&B)结果为真
`||`	逻辑`OR`运算符。如果两个操作数中的任何一个非零，则条件变为真。	(A||B)结果为真
`!`	逻辑非运算符。用于反转其操作数的逻辑状态。如果条件为真，则逻辑非运算符将为假。	!(A&&B)结果为真

四、Go语言 位运算符

按位操作符对位进行操作，并执行逐位操作。 `&`， `|`和`^`的真值表如下：

p	q	p&q	p|q	p^q
0	0	0	0	0
0	1	0	1	1
1	1	1	1	0
1	0	0	1	1

假设`A = 60`, `B = 13`; 现在的二进制格式，如下：

```
A = 0011 1100

B = 0000 1101

-----------------

A&B = 0000 1100

A|B = 0011 1101

A^B = 0011 0001

~A  = 1100 0011
```

Go语言支持的位运算符,如在下表中所列。 假设变量`A=60`，并且变量`B=13`，则：

运算符	描述	示例
&	如果两个操作数中都存在二进制 `AND` 运算符，则将其复制到结果。	(`A&B`)结果为 `12` ,也就是 `0000 1100`
|	二进制 `OR` 运算符复制一个位，如果它存在于任一操作数。	(A|B)结果为61,也就是 `0011 1101`
^	二进制 `XOR` 运算符复制位，如果它在一个操作数中设置，但不是在两个操作数中设置。	(A^B)结果为 `49` ,也就是 `0011 0001`
<<	二进制左移位运算符。左操作数值向左移动由右操作数指定的位数。	`A<<2` 结果为 `240` ,也就是 `1111 0000`
>>	二进制右移运算符。左操作数值向右移动由右操作数指定位数。	`A>>2` 结果为15,也就是 `0000 1111`

五、Go语言 赋值运算符

Go语言支持以下赋值运算符：

运算符	描述	示例
=	简单赋值操作符，将值从右侧操作数分配给左侧操作数	`C=A+B`，就是将`A+B`的值赋给`C`
`+=`	相加和赋值运算符，向左操作数添加右操作数，并将结果赋给左操作数	`C+=A`相当于`C=C+A`
-=	减去和赋值运算符，从左操作数中减去右操作数，并将结果赋给左操作数	`C-=A`相当于`C=C-A`
*=	乘法和赋值运算符，它将右操作数与左操作数相乘，并将结果赋给左操作数	`C*=A`相当于`C=C*A`
/=	除法和赋值运算符，它用右操作数划分左操作数，并将结果分配给左操作数	`C/=A`相当于`C=C/A`
%=	模数和赋值运算符，它使用两个操作数来取模，并将结果分配给左操作数	`C%=A`相当于`C=C%A`
<<=	左移和赋值运算符	`C<<=2`相当于`C=C<<2`
>>=	右移和赋值运算符	`C>>=2`相当于`C=C>>2`
&=	按位和赋值运算符	`C&=2`相当于`C=C&2`
^=	按位异或和赋值运算符	`C^=2`相当于`C=C^2`
|=	按位包含`OR`和赋值运算符	C|=2相当于C=C|2

六、Go语言 其他运算符

还有一些其他重要的运算符包括`sizeof`和`? :`，在Go语言中也是支持的。

运算符	描述	示例
&	返回变量的地址	`&a`将给出变量`a`的实际地址。
*	指向变量的指针	`*a`是指向变量`a`的指针。

0x03 Go语言中的运算符优先级

运算符优先级确定表达式中的分组。这会影响表达式的计算方式。某些运算符比其他运算符具有更高的优先级; 例如，乘法运算符比加法运算符有更高的优先级。

当同级别的运算符出现在同一个表达式中，从左到右的顺序计算，比如乘除一起，不管是乘在前面还是除在前面都是从左到右计算乘、除运算符。加减亦是如此。

例如：`x = 7 + 3 * 2`; 这里，计算结果`x`被分配`13`，而不是`20`，因为运算符`*`具有比`+`有更的优先级，所以它首先乘以`3 * 2`，然后加上`7`。

这里，具有最高优先级的运算符放在表的顶部，具有最低优先级的运算符出现在底部。 在表达式中，将首先计算较高优先级运算符。

分类	描述	关联性
后缀	()[]->.++ --	左到右
一元	+ -!~++ --(type)*&sizeof	右到左
乘法	*/ %	左到右
加法	+ -	左到右
移位	<<>>	左到右
关系	<<=>>=	左到右
相等	==!=	左到右
按位AND	&	左到右
按位XOR	^	左到右
按位OR	|	左到右
逻辑AND	&&	左到右
逻辑OR	||	左到右
条件	?:	右到左
分配	=+=-=*=/= %=>>= <<= &= ^= |=	右到左
逗号	,	左到右