C语言运算符优先级和结合性一览表

所谓优先级就是当一个表达式中有多个运算符时，先计算谁，后计算谁。这个其实我们在小学学算术的时候就学过，如1+4÷2。

但是C语言中的运算符已经远不止四则运算中的加减乘除了，还有其他很多运算符。当它们出现在同一个表达式中时先计算谁后计算谁呢？所以本节还是有必要讲一下的。最后我还会将所有运算符展示出来，然后告诉你哪个优先级高、哪个优先级低。

首先不需要专门记忆，也没有必要。因为作为初学者，哪个优先级高、哪个优先级低我们很难记住。就算死记硬背记住了，时间长不用也会忘记。所以当一个表达式中有多个运算符时，如果不知道哪个优先级高哪个优先级低就查一下优先级表，附录E有一个运算符优先级表。此外用的时间长了自然而然就记住了，这样记才会记得深刻。

而且事实上在编程的时候也不需要考虑优先级的问题。因为如果不知道优先级高低的话，加一个括号就可以了，因为括号( )的优先级是最高的。比如前面的程序中：
k = (j>i) && (8==i);
根据运算符的优先级，这条语句完全可以写成：
k = j>i && 8==i;
但是第一种写法别人一看就知道先计算谁后计算谁。

而且加圆括号也是一种编程规范，因为程序不只是写给自己看。

此外运算符还有“目”和“结合性”的概念，这个很简单。“目”就是“眼睛”的意思，一个运算符需要几个数就叫“几目”。比如加法运算符+，要使用这个运算符需要两个数，如3+2。对+而言，3 和 2 就像它的两只眼睛，所以这个运算符是双目的。

C语言中大多数的运算符都是双目的，也有单目和三目的。单目运算符比如逻辑非，如!1，它就只有一只眼睛，所以是单目的。整个C语言中只有一个三目运算符，即条件运算符? :。这个稍后讲到条件语句的时候再介绍。关于“目”大家了解一下就行了。

那么“结合性”是什么呢？上面讲的优先级都是关于优先级不同的运算符参与运算时先计算谁后计算谁。但是如果运算符的优先级相同，那么先计算谁后计算谁呢？这个就是由“结合性”决定的。

比如1+2×3÷4，乘和除的优先级相同，但是计算的时候是从左往右，即先计算乘再计算除，所以乘和除的结合性就是从左往右。就是这么简单！

C语言中大多数运算符的结合性都是从左往右，只有三个运算符是从右往左的。一个是单目运算符，另一个是三目运算符，还有一个就是双目运算符中的赋值运算符=。双目运算符中只有赋值运算符的结合性是从右往左的，其他的都是从左往右。运算符的“结合性”也不要死记，在不断使用中就记住了。

运算符优先级和结合性一览表

优先级	运算符	名称或含义	使用形式	结合方向	说明
1	[]	数组下标	数组名[常量表达式]	左到右	
	()	圆括号	(表达式) 函数名(形参表)		
	.	成员选择（对象）	对象.成员名		
	->	成员选择（指针）	对象指针->成员名		
2	-	负号运算符	-表达式	右到左	单目运算符
	(类型)	强制类型转换	(数据类型)表达式		
	++	自增运算符	++变量名 变量名++		单目运算符
	--	自减运算符	--变量名 变量名--		单目运算符
	*	取值运算符	*指针变量		单目运算符
	&	取地址运算符	&变量名		单目运算符
	!	逻辑非运算符	!表达式		单目运算符
	~	按位取反运算符	~表达式		单目运算符
	sizeof	长度运算符	sizeof(表达式)		
3	/	除	表达式 / 表达式	左到右	双目运算符
	*	乘	表达式*表达式		双目运算符
	%	余数（取模）	整型表达式%整型表达式		双目运算符

4	+	加	表达式+表达式	左到右	双目运算符
	-	减	表达式-表达式		双目运算符
5	<<	左移	变量<<表达式	左到右	双目运算符
	>>	右移	变量>>表达式		双目运算符
6	>	大于	表达式>表达式	左到右	双目运算符
	>=	大于等于	表达式>=表达式		双目运算符
	<	小于	表达式<表达式		双目运算符
	<=	小于等于	表达式<=表达式		双目运算符
7	==	等于	表达式==表达式	左到右	双目运算符
	!=	不等于	表达式!= 表达式		双目运算符
8	&	按位与	表达式&表达式	左到右	双目运算符
9	^	按位异或	表达式^表达式	左到右	双目运算符
10	|	按位或	表达式|表达式	左到右	双目运算符
11	&&	逻辑与	表达式&&表达式	左到右	双目运算符
12	||	逻辑或	表达式||表达式	左到右	双目运算符
13	?:	条件运算符	表达式1? 表达式2: 表达式3	右到左	三目运算符
14	=	赋值运算符	变量=表达式	右到左	
	/=	除后赋值	变量/=表达式		
	*=	乘后赋值	变量*=表达式		
	%=	取模后赋值	变量%=表达式		
	+=	加后赋值	变量+=表达式		
	-=	减后赋值	变量-=表达式		
	<<=	左移后赋值	变量<<=表达式		
	>>=	右移后赋值	变量>>=表达式		
	&=	按位与后赋值	变量&=表达式		

	^=	按位异或后赋值	变量^=表达式		
	|=	按位或后赋值	变量|=表达式		
15	,	逗号运算符	表达式,表达式,...	左到右	

上表中可以总结出如下规律：

1. 结合方向只有三个是从右往左，其余都是从左往右。
2. 所有双目运算符中只有赋值运算符的结合方向是从右往左。
3. 另外两个从右往左结合的运算符也很好记，因为它们很特殊：一个是单目运算符，一个是三目运算符。
4. C语言中有且只有一个三目运算符。
5. 逗号运算符的优先级最低，要记住。
6. 此外要记住，对于优先级：算术运算符 > 关系运算符 > 逻辑运算符 > 赋值运算符。逻辑运算符中“逻辑非 !”除外。

一些容易出错的优先级问题

上表中，优先级同为1 的几种运算符如果同时出现，那怎么确定表达式的优先级呢？这是很多初学者迷糊的地方。下表就整理了这些容易出错的情况：

优先级问题	表达式	经常误认为的结果	实际结果
. 的优先级高于 *（-> 操作符用于消除这个问题）	*p.f	p 所指对象的字段 f，等价于: (*p).f	对 p 取 f 偏移，作为指针，然后进行解除引用操作，等价于: *(p.f)
[] 高于 *	int *ap[]	ap 是个指向 int 数组的指针，等价于: int (*ap)[]	ap 是个元素为 int 指针的数组，等价于: int *(ap [])
函数 () 高于 *	int *fp()	fp 是个函数指针，所指函数返回 int，等价于: int (*fp)()	fp 是个函数，返回 int*，等价于: int* (fp())
== 和 != 高于位操作	(val & mask != 0)	(val &mask) != 0	val & (mask != 0)
== 和 != 高于赋值符	c = getchar() != EOF	(c = getchar()) != EOF	c = (getchar() != EOF)
算术运算符高于位移 运算符	msb << 4 + lsb	(msb << 4) + lsb	msb << (4 + lsb)
逗号运算符在所有运 算符中优先级最低	i = 1, 2	i = (1,2)	(i = 1), 2

这些容易出错的情况，希望读者好好在编译器上调试调试，这样印象会深一些。一定要多调试，光靠看代码，水平是很难提上来的。调试代码才是最长水平的。