C语言基本概念之表达式

原创 astrotycoon 2016-03-11 16:47:36

什么是表达式（表达式的定义）？

表达式是C语言的重要语法成分，不过对于**表达式的定义**，好像从来没有人关注过。今天就孔乙己一把吧，哈哈。先贴标准对expression的定义（ISO/IEC 9899:2011 6.5）：

An expression is a sequence of operators and operands that specifies computation of a value, or that designates an object or a function, or that generates side effects, or that performs a combination thereof. The value computations of the operands of an operator are sequenced before the value computation of the result of the operator.

标准就是标准，不搞的晦涩难懂都不好意思拿出来，哈哈。我们来解读一下吧。

“An expression is a sequence of operators and operands that specifies computation of a value”，这句明确给出了表达式的定义，即表达式是由一系列**运算符（operators）**和**操作数（operands）**组成的。这既是表达式的定义，同时也指明了**表达式的组成**成分。运算符指明了要进行何种运算和操作，而操作数则是运算符操作的对象。

我们来看一些合法的表达式，如下：

4

-6

4 + 21

a * (b + c/d) / 20

q=5*2

x=++q % 3

q > 3

"hello world"

可以看到一个表达式也可以没有操作符，例如“4”这种形式就是最简单的表达式形式，即最简单的表达式只有一个常量或一个变量名称而没有操作符。

还可以看出，一些表达式是多个较小的表达式的组合，这些小的表达式被称为子表达式（subexpression）。例如表达式c/d是表达式a * (b + c/d) / 20的子表达式，而表达式c和d又是表达式c/d的子表达式。

表达式有什么作用（表达式的目的）？

继续解读标准，可以看到**表达式的目的**有如下几个：

（1）**计算数值**（computation of a value）。

哈哈，这个目的其实是表达式的主要目的。

很好理解，例如表达式“3+2”的目的就是计算数值3和2的和。于此同时，表达式同时也表示这个计算所得到的值。具体的可以参阅下面的“表达式属性”小节。

（2）**指明数据对象或者函数**（designates an object or a function）

例如程序中有int i;声明语句，那么表达式i=3中子表达式i就指代i所代表的那个对象（object），即一块连续的内存空间。

而在表达式&printf中printf指代的是标准C库中的printf函数。

（3）**产生副作用**（generate side effects）

首先要明白什么是副作用，还是看标准（ISO/IEC 9899:2011 5.1.2.3）

Accessing a volatile object, modifying an object, modifying a file, or calling a function that does any of those operations are all side effects, which are changes in the state of the execution environment. Evaluation of an expression in general includes both value computations and initiation of side effects. Value computation for an lvalue expression includes determining the identity of the designated object.

OK，**副作用（side effects）就是运行时对数据对象或文件的修改。**来看几个例子：

①表达式 i = 50的副作用是将变量i的值设置为50，这样说是不是让你感到惊讶呢？ 这怎么可能是副作用呢，看起来更像是主要目的啊！ 然而，从C的角度来看，主要的目的却是对表达式求值。

②表达式printf("ABC")的值为3（实际打印的字符数，不包括字符'\0'），副作用就是在标准输出设备上连续打印字符A、B和C。

注意，并不是所有的表达式都有副作用，表达式2+3的值为5，但是没有任何副作用。

（4）**以上目的的组合**（combination）

最后来看一个完整的程序吧：

```c
#include <stdio.h>

static int print(void)
{
    printf("hello world\n");    /* Genearates side effect */

    return (100);
}

int main(int argc, const char *argv[])
{
    int a, b;

    a + b;          /* A computation */
    a;              /* An object */
    print();        /* A function */
    a = b;          /* Genearates side effect */
    a = b, a + print(); /* A combination of all above */

    return (0);
}
```

需要注意的是，函数调用也是表达式，print()这个表达式中()是操作符，而print是操作符()的操作数。

表达式的属性有哪些（表达式的属性）？

好了，在阅读了上面两小节之后，想必对什么是表达式以及表达式的诸多作用应该有个清晰的认识了吧。那么，在C语言中一个表达式到底具有哪些属性呢？ 首先给出结论如下：

任何表达式都有值和类型两个基本属性。

简单说明一下：既然我们知道一个表达式最终会计算出一个值，那么任何值在C中肯定是具有类型的。对吧，毋庸置疑啊。

但是，你可能会质疑这样一种情形，在C中如果一个函数的返回值类型为void，例如对函数void aa(int)的调用aa()，这样的表达式也会有值吗？ 关于这点我也不是很清楚，在wikipedia中有这样一段：In C and most C-derived languages, a call to a function with a void return type is a valid expression, of type void. Values of type void cannot be used, so the value of such an expression is always thrown away.

所以，记住表达式有两个属性就OK，没必要搞的太咬文嚼字！

（2016-3-23 16:55:14）补充：其实标准对void表达式是有说明的，如下（ISO/IEC 9899:2011 6.3.2.2）：

The (nonexistent) value of a void expression (an expression that has type void) shall not be used in any way, and implicit or explicit conversions (except to void) shall not be applied to such an expression. If an expression of any other type is evaluated as a void expression, its value or designator is discarded. (A void expression is evaluated for its side effects.)

可以看到，void expression的类型是void类型，值为void类型的值， 只是不存在这种值罢了。此外，对void expression的任何显式或者隐式的转换都是不允许的，而将其它类型的表达式的值转化为void类型的值是允许的，编译器会将这个值丢弃（可以说绝大多数情况下，void expression的目的就是产生副作用）。

我们可以借助gdb提供的print和ptype命令来实际地查看表达式的值和类型两大属性。

```
(gdb) ptype 1
type = int
(gdb) ptype 1.0
type = double
(gdb) ptype "hello world"
type = char [12]
(gdb) print printf("hello world\n")
hello world
```

```
$2 = 12
(gdb) ptype printf("hello world\n")
type = int
(gdb) ptype setbuf(stdout, 0)
type = void
(gdb) print setbuf(stdout, 0)
$3 = void
```

可以看到在C中字符串常量的类型是字符型数组。setbuf的返回类型为void，我们在使用print打印其值是没有意义的，因此gdb索性给了个void。

有哪些种类的表达式（表达的分类）？

在C中，表达式的种类几乎是与操作符相对应的。可是我们在第一小节也看到了，最简单的表达式形式也可以没有操作符，对于这种没有操作符的表达式在C标准中又分为两类，分别是基本表达式（primary expression）和常量表达式（constant expression）。现在我们把C中的表达式全部列出来，加上简略的说明即可。

- **基本表达式（primary expression）**
- **常量表达式（constant expression）**
- **后缀表达式（postfix expression）**
- **一元表达式（unary expression）**
- **强制转换表达式（cast expression）**
- **乘法表达式（multiplicative expression）**
- **加法表达式（additive expression）**
- **移位表达式（shift expression）**
- **关系表达式（relational expression）**
- **相等表达式（equality expression）**
- **AND表达式（AND expression）**
- **异或表达式（exclusive OR expression）**
- **或表达式（inclusive OR expression）**
- **逻辑与表达式（logical AND expression）**
- **逻辑或表达式（logical OR expression）**
- **条件表达式（conditional expression）**
- **赋值表达式（assignment expression）**

表达式与语句的关系

表达式求值的规则

参考链接：

《标准C语言指南》

《Expressions》（cppreference.com C）

《Expressions》（cppreference.com C++）

《C：表达式、语句、声明》

《C Programming – Expressions》

《Semantics of Expressions》

《Expressions (C++)》

《Expression Statements》

《expression》

《Introduction to Operators and Operands》

《Side effects in C》

《C语言里的side effect是什么意思？》

《Linux C编程一站式学习 -- 表达式》

《C/C++ 语言中的表达式求值》

《C的优先级、求值顺序和表达式副作用》

《Understanding lvalues and rvalues in C and C++》

《Why array type object is not modifiable?》

《C Programming Operators》

《C - Operators》

《Operands and Expressions》

《ISO/IEC 9899 C语言标准（非官方翻译）》