C：表达式、语句、声明

“表达式、语句、声明之间的区别到底是什么？”

这个问题对很多人来说确实很模糊，甚至很多出版物中也有很多错误的讲述，故此本文力图对此做一详尽说明和澄清。

表达式(Expression)

根据C标准，表达式(Expression)是运算符(operator)和操作数(operand)所构成的序列，例如"3+2"。最简单的情形，也可能没有运算符，例如“3”。

可能需要特意提一下的是，函数调用也是表达式。例如 sqrt(1.0),这里的“()”是一个运算符，sqrt和1.0则是“()”这个运算符的操作数。

表达式所表达的可能是要求计算机进行值的计算(computation of a value)，例如“3+2”这个表达式，表示的就是要求计算机求3+2的值。当然，表达式同时也表示这个计算所得到的值。

表达式还可能用来指明一个数据对象(object)或一个函数(function)。例如，若有 int i; 那么在 i = 1 这个表达式中的 i 这个子表达式就是指 i 所代表的那个object——一块连续的内存。再比如, 表达式 & printf 中，printf这个子表达式表示的是相应的printf()库函数。

除此之外，表达式可能产生副效应(side effects)，例如printf("ABC") 这个表达式的值是3，其副效应是在标准输出设备上连续输出A、B、C这三个字符。再比如,对于 int i; 表达式 i = 3 的值是3，副效应是 i 所代表的数据对象被写入了3。

语句(Statement)

C语言语句(Statement)的定义从其功能上来说有些含糊：语句规定的是一种“动作”(action)。这种动作最显著的特点是规定了如何跳转或结束。例如，对于int i; "i = 1 ;"这条语句的意义是到“;”位置时 i = 1 这个表达式求值(Evaluate)， 即值的计算和副效应必须完成。再比如“return ;”语句规定的是跳转到另一位置继续执行。

尽管语句的功能有些含糊，但其语法形式确实极其清晰的。形式上较为简单的语句有:

表达式语句(expression-statement)，形式为:

expression ;

其中的表达式是可选的。这种语句以“;”作为结束标志。对于初学者来说最常见的表达式语句是调用printf()函数所形成的：printf("Hello World\n");

跳转语句(jump-statement)，包括：

goto 语句、continue 语句、break 语句和return 语句。这种语句也以“;”作为结束标志。

复合语句(compound-statement)，其一般形式为：

{block-item-list}

其中的block-item-list可以有也可以没有，可以是声明，也可以是语句。

有一点特别需要说明，复合语句并非以“;”作为结束标志。这表明语句中不一定有“;”。国内C语言书中有一种常见的陈词滥调：“一个语句必须在最后有一个分号，分号是语句中不可缺少的组成部分”(谭浩强,《C程序设计》第四版,p58)，那绝对在是瞪着眼睛说胡话。因为复合语句的最后就不需要有分号。最简单的复合语句只有一对“{}”，根本就没分号什么事儿。

其他的语句都是构造性的，即是在其他语句的基础上构造出来的。比如标号语句(labeled-statement)，就是在语句前加一冒号及其他内容：

identifier : statement

case constant-expression : statement

default : statement

选择语句也是依据类似的原则在语句的基础上构造的：

if (expression) statement
if (expression) statement else statement
switch (expression) statement

值得一提的是switch语句并不一定是在复合语句的基础上构造的，这和很多人的常识不同。例如，

switch (i) case 0:case 1:case 2:printf("ABC\n");

就是一条完全合法的switch语句。 其功能等价于

if (i==0||i==1||i==2) printf("ABC\n");

C语言的循环语句同样是在语句的基础上构造而成。

while (expression) statement
do statement while (expression) ;
for (expressionopt ; expression ; expression) statement
for (declaration expression; expression) statement

在这些语句中，只有do-while语句最后一定是以“;”作为结束标志。

总之，由于复合语句并不是以分号作为结束标志，而很多语句都是在语句的基础上进一步构造而成，因此除了表达式语句、跳转语句及do-while语句一定是以分号结束，其他语句则可能以分号结束，也可能不以分号结束，分号并不是语句必须的组成部分。

声明(Declarations)

除了static_assert declaration(C11)，声明的作用都是向编译器解释一个或多个标识符的含义及属性。

C标准给出的声明的一般形式为

declaration-specifiers init-declarator-list ;

因此，通常情况下声明都有“;”。

但实际上，函数定义同时也是声明(A definition of an identifier is a declaration)，在格式上却与此不符。

尽管功能明确，形式简单，但声明其实是C语言中最复杂的成分，至少是之一。这种复杂性被 declarator 漠然地掩盖起来了。

C语言要求每个声明至少要声明一个 declarator 、一个tag或一个枚举成员。也就是说

int i ;//没问题，i是declarator

int ; //违法，无declarator

struct t;//合法，声明了一个tag——t

enum { A };//是合法的，声明了枚举成员A

struct { int a ;}; //违背语言要求，可能招致警告。(在C语言早期，并不违反标准)

union { int a ;}; //违背语言要求，可能招致警告。(在C语言早期，并不违反标准)

声明不是语句(《C Primer Plus》一书把声明称为“声明语句”，明显是把C++的概念张冠李戴到C上面了)。一个简单的证明是，声明无法按照语句规则构成语句。譬如

if(1) int i ; //这在C语言中是不成立的

有些声明也叫定义(definition)，所有的定义都是声明。包括

导致保留存储区的对象声明。最典型的就是局部变量的声明。

含有函数体的函数声明，即函数定义。函数定义尽管在形式上不符合由“;”结尾的形式，但同样是一个声明，有时也被称之为外部声明(external declaration)或外部定义(external definition)。但习惯上，很多人所说的“函数声明”往往特指那些非定义式的函数声明。

除此之外，枚举常量(enumeration constant)和typedef name也是定义。

declaration-specifiers包括存储类别说明符(storage class specifier)，类型说明符(type specifier) ，类型限定符(type qualifier),函数说明符(function specifier),从C11起又增加了一个对齐说明符(alignment specifier)。

和多数人常识不符的是，typedef也属于存储类别说明符(storage class specifier)。

把typedef类型定义归为声明并把typedef关键字作为存储类别说明符(storage class specifier)只是为了语法描述上的方便。实际上typedef并没有像其他几个存储类别说明符(extern,static,auto,register,_Thread_local(C11))那样对存储类别做出任何说明。

在每个声明中，存储类别说明符只能出现一次。C11引人新的存储类别说明符_Thread_local之后，这个说法不再成立了。

类型说明符(type specifier)包括：void、char、short、int、long、float、double、signed、unsigned、_Bool、_Complex、atomic-type-specifier、struct-or-union-specifier、enum-specifier、typedef-name，使用时可能是类型说明符的某种组合，譬如 long double。其中_Bool、_Complex是C99新增的，最初还增加了一个_Imaginary，后来又删除了。atomic-type-specifier是C11新增的。此外，C11在struct-or-union声明中正式支持了匿名结构体或联合体(anonymous structures and unions)。

类型限定符(type qualifier)在C90中只有const和volatile两个，const借鉴的是C++，volatile则完全是C标准委员会的发明。C99增加了一个restrict，只用于指针类型，提出这个限定符的目的是更好地优化。C又增加了一个_Atomic类型限定符。这个限定符也是唯一的一个Atomic type specifiers，这种Atomic 类型说明符是C11新增的内容。

函数说明符(function specifier)是C99之后出现的。C99增加了一个inline这个函数说明符，C11又新增了一个_Noreturn函数说明符用来说明不返回调用者的函数,例如exit()函数。

C11增加的另一个新的声明说明符是对齐说明符(alignment specifier)。对齐问题不再像以前那样犹抱琵琶半遮面，而是直接摆到了桌面上。

声明的init-declarator-list部分由一个或多个用“,”分隔的init-declarator组成。init-declarator可以是一个单独的declarator或初始化形式declarator = initializer。

最简单的declarator是一个标识符。这是一种direct-declarator。在direct-declarator前面可以加* type-qualifier-list用以声明指针。声明指针时type-qualifier置于*的右侧。而在declaration-specifiers中则没有这样的次序要求。

direct-declarator还可以具有下面几种形式：

(declarator)
direct-declarator [type-qualifier-list assignment-expressionopt]
direct-declarator [static type-qualifier-list assignment-expression]
direct-declarator [type-qualifier-list static assignment-expression]
direct-declarator [type-qualifier-list *]
direct-declarator (parameter-type-list)
direct-declarator (identifier-list)

第一种中的“()”可能改变对declarator的类型解释(例如：int *p[1];int (*p)[1];)。

自C99起，不再要求声明数组时[]内是整数常量表达式，可以用变量描述数组尺寸。例如 int a[n]; 这就是所谓的VLA。VLA是C99的正式特性，在C11中则是一个可选特性。编译器可以支持，也可以不支持。至于[]内的* 、static及type-qualifier-list，只用于声明函数参数时。

函数声明“()”内虽然可以有parameter-type-list和 identifier-list两种形式，但后者其实是一种正在逐步废弃的形式，即

void f(a)

int a;

{/*……*/}

这种形式。除了在老代码中可以看到，现在已经很少有人这样写了。现代风格的C语言提倡函数原型风格的声明，即 参数为parameter-type-list这种形式。这种形式有两种：

parameter-list

或
parameter-list , ...

“...”用于声明参数个数不定的函数。

标签: C语言, 表达式, C99, C11, 声明, 语句

posted @ 2013-04-06 11:07 garbageMan 阅读(5947) 评论(6)

评论列表

#1楼 2013-04-07 16:28 黄博文 回复 引

函数式编程中这些概念显的及其重要。

支持(0) 反对(0)

#2楼 [楼主] 2013-04-07 21:38 garbageMan 回复 引

@ 黄博文

引用

函数式编程中这些概念显的及其重要。

所言极是

支持(0) 反对(0)