**[iBoard 电子学堂][第二卷 C程序设计语言 ]第二篇 数据类型与运算符**

**一、数据类型**

**1、变量及命名规则**

变量是存储在硬件存储器某一区间的数值。C语言中，变量的命名有一定的规则。变量名字是由字母和数字组成的序列（a~z、A~Z、0~9和 \_），但第一个字符必须为字母（“\_”下划线也可以）。C语言是区分大小写的，所以abc和ABC 是两个不同的名字。变量不能与C语言关键字重复（如 while、if、else、int等等）。

在函数内部的变量，我们称之为局部变量，其作用域为函数内部。函数体外声明的变量，我们称之为全局变量；其作用域为此文件内的所有函数。一般地，全局变量尽量能使其表达出自身含义，局部变量则需要使用较短的命名，以便于阅读。

**2、变量的声明**

所有的变量都必须先声明后使用。声明过程中，变量类型后可以包含一个或者多个该类型的变量他们之间用逗号分开。且变量声明过程中可以用紧跟的一个表达式进行初始化。如下所示：

int counter;

float frequency，amplitude;

int i = 6;

float pi\_2 = 3.1415926\*2;

C89规范中规定，在函数中，我们需首先声明变量，然后再紧跟着函数实体。虽然C99规范允许在函数过程中声明变量，但仍不推荐这样做。

变量可以被 const 修饰（把 const 关键字放到变量类型前，如 const int），该关键字表示变量值为一个常数，所以不能被修改。如：

const int i = 6；

**3、基本数据类型**

C语言提供了表一中的数据类型；这里需要指出的是，依编译器、硬件平台不一样，其数据类型占用字节数也会有差异，而且并不是所有的编译器都支持以下数据类型。

表一 C语言数据类型

|  |  |  |  |
| --- | --- | --- | --- |
| **类 型** | **名 称** | **字 节** | **数据范围** |
| （signed）char | （有符号）字符型 | 1 | -128~127 |
| unsigned char | 无符号字符型 | 1 | 0~255 |
| （signed）int | （有符号）整型 | 2或者  4 | -32768～32767或  -2147483648～2147483647 |
| unsigned int | 无符号整型 | 2或者  4 | 0～32767或  0～4294967295 |
| （signed）short int | （有符号）短整型 | 2 | -32768～32767或 |
| unsigned short int | 无符号短整型 | 2 | 0～32767或 |
| unsigned long int | 无符号长整型 | 4 | 0～4294967295 |
| （signed）long long int | （有符号）长长整形 | 8 | -263～263-1 |
| unsigned long long int | 无符号长长整型 | 8 | 0～264-1 |
| float | 浮点型 | 4 | 3.4x10−38～3.4x10+38  （7位有效数字） |
| double | 双精度浮点型 或  浮点型 | 4或者  8 | 3.4x10−38～3.4x10+38 或  1.7x10−308～1.7x10+308  （15位有效数字） |

**二、运算符**

**1、算术运算符**

一元算术运算符包括+、-，这里的“元“代表操作数的个数，一元算术运算符+、- 也就是数字的正、负号。

二元算术运算符包含+、-、\*、/、%，分别为加、减、乘、除、模。整数中的相除，会略去余数。如：

int i = 5，j；

若进行 j = i / 2; 这样的运算，则 j = 2。% 为求模运算，亦可以说求余运算。例如 ：

int x = 5 % 3；

则x = 2。模运算不能用于float 及 double类型。

一元算术运算符的优先级最高，然后是二元算术运算符中的 \*、/ 和 %。二元算术运算符 + 和 – 优先级较低。算术运算符采用从左到右的结合规则。

**2、关系运算符、相等性运算符**

关系运算符包括以下几个运算符：

> >= < <=

分别代表大于、大于等于、小于和小于等于。

相等性运算符包含了：

== ！=

分别代表相等、不等两种情况。关系运算符的优先级比相等性运算符的优先级高。但是算术运算符的优先级较关系运算符高。所以表达式 i < j + 1 我们需要先计算 j+1的值后，然后与 i相比较。

**3、逻辑运算符**

逻辑运算符包括：

|| && ！

分别表示逻辑或、逻辑与与求非运算。a || b含义为a与b有一个为真时，表达式结果为真，否则为假。 a && b 含义为a 与b 都为真时，结果为真，否则为假。！符号表示对操作数取非，表示将非 0 操作数转换为零，讲操作数0转换为1。若判断一个操作数是否为零，我们可以通过以下方式：

if（！a）{

}

运算符 && 的优先级比 || 运算符的优先级高；但是两者都比相等性运算符优先级低。

**4、自增、自减运算符**

C语言提供两个非常简便的用于变量递增与递减的运算符，即：++ 和 --。++和-- 他们既可以作为前缀放于操作数前（如 ++i），也可以作为后缀放于操作数后（如 i++）。但是他们表达的意思不同， ++i 表示变量 i首先加1，然后在使用 i 的值进行别的运算，而 i++则是首先使用 i 的值进行运算后，在把 i 加1。我们举个例子，令 i = 5；则：

x = i ++ ；//这里 x=5

x = ++ i； //这里 x=6

而且这两条语句执行完毕后，i的值均为6，这就是前缀和后缀不同的地方。同理，--运算符为递减1操作，原理相似。若不存在赋值，一条语句只有递增或者递减的情况下，前缀和后缀结果是一致的。如：

x = i； 和 x = i；

i ++； ++ i；

这两句执行完毕后，x 和 i 的结果都是一样的。

**5、按位运算符**

C语言中提供了六个按位运算符，分别为：

& | ^ << >> ~

分别代表位与（AND）、或（OR）、异或（XOR）、左移、右移和按位取反。位运算符只能用于有符号和无符号的char、short、int或者long类型，浮点型不能按位运算。

位运算是嵌入式系统程序设计中较常用的运算操作，它常常伴随硬件端口操作。我们声明一个 16 位的无符号整数，若我们需要让最低位变为零，其他位不变的情况下，我们可以按下面的代码来实现。

unsigned short int i = 3；

i = i & 0xfffe；

若我们需要让一个整数的的最低位变1，其余位不变，则可以按以下代码来实现。

unsigned short int i = 3；

i = i | 1；

移位运算符（左移 << 和 右移 >>）分别用于将运算的操作数左移和右移，移位后，空出的位填充零。移位运算也常常用于芯片驱动编写，如串行总线数据收、发等操作。为了避免歧义，我们最好对无符号数进行移位操作。

按位取反符号~用于对整数求反码，即对操作数的二进制为1变为0，0变为1。按位取反也常用于硬件操作，如我们对一整排的LED实现量、暗间歇性的闪烁操作。

**6、赋值运算符**

在赋值表达式中，若表达式左边的变量重复出现在表达式的右边，如：

i = i + 1；

我们就可以采用一种简写的方式：

i += 1；

这里 += 称之为赋值运算符。多数二元运算符都有一个相应的赋值运算符 op=，其中，op 可以使下面这些运算符之一：

+ - \* / ％ << >> & | ^

如果表达式 expr1 和 expr2 ，进行 op= 运算，即：

expr1 op= expr2，则此运算等价于： expr1 = （expr1） op （expr2）。

**7、条件运算符 ? :**

C语言中有一个三元运算符，它用于表达以下逻辑关系。我们首先看下面代码。

if (a > b) x = a;

else x=b

这两条语句用于求出a 和b 中的最大值，并把结果保存在 x 中。则我们可以采用如下表达式等价：

x = （a > b）?a ： b；

我们总结一下，可利用三元运算符？：写出表达式：

expr1？expr2：expr3

它的含义为：若expr1 为真，则表达式的值为expr2，否则表达式的值为 expr3。

**8、运算符的优先级:**

运算符的优先级的含义为同时出现多个运算时，我们首先计算哪一个，优先级高的优先计算。我们把运算符优先级大小由上向下排列，同一行中优先级相同。如表二所示。

表二 运算符优先级

|  |  |  |
| --- | --- | --- |
| **运 算 符** | **含 义** | **结合方式** |
| （） [ ] -> . | 括号、数组、结构体成员访问 | 由左向右 |
| ！ ~ ++ -- + - \* ＆ （类型） sizeof | 非、按位取反、累加、累减、正号、负号、取值、取地址、类型转换、求存储区大小。本行均为一元运算符 | 由右向左 |
| \* / % | 乘、除、模 | 由左向右 |
| + - | 加、减 | 由左向右 |
| << >> | 左移、右移 | 由左向右 |
| < <= >= > | 小于、小于等于、大于等于、大于 | 由左向右 |
| == ！= | 等于、不等于 | 由左向右 |
| & | 位与 | 由左向右 |
| ^ | 位异或 | 由左向右 |
| | | 位或 | 由左向右 |
| && | 逻辑与 | 由左向右 |
| || | 逻辑或 | 由左向右 |
| ? : | 条件运算符 | 由右向左 |
| = += -= \*= /= &= ^= |= <<= >>= | 赋值 | 由右向左 |
| , | 逗号 | 由左向右 |