32 单片机采用的 C99 标准在参考书籍中！！！

C 语言由 Dennis M. Ritchie 在 1973 年设计和实现。从那以后使用者逐渐增加。到 1978 年 Ritchie 和 Bell 实验室的另一位程序专家 Kernighan 合写了著名的《The C Programming Language》，将 C 语言推向全世界，许多国家都出了译本，国内有一些 C 语言书就是这本书的翻译或者编译。由这本书定义的 C 语言后来被人们称作 K&R C。

随着 C 语言使用得越来越广泛，出现了许多新问题，人们日益强烈地要求对 C 语言进行标准化。1983 年，美国国家标准协会（ANSI）组成了一个委员会，X3J11，为了创立 C 的一套标准。经过漫长而艰苦的过程，该标准于 1989 年完成，这个版本的语言经常被称作 ANSI C，或有时称为 C89（为了区别 C99）。在 1990 年，ANSI C 标准（带有一些小改动）被美国国家标准协会（ANSI）采纳为 ISO/IEC 9899:1990。这个版本有时候称为 C90 或者 ISO C。综上，ANSI C、ISO C、C89、C90 其实是同一种标准。

2000 年 3 月，ANSI 采纳了 ISO/IEC 9899:1999 标准。这个标准通常指 C99。 C99 新增了一些特性，如：支持不定长的数组，即数组长度可以在运行时决定。变量声明不必放在语句块的开头，for 语句提倡写成 for(int i=0;i<100;++i) 的形式，即 i 只在 for 语句块内部有效。

参考 c 专家编程 1.8 节 19 页

K&R C 和 ANSI C 之间的区别

阅读本节内容时，我假定你已经完全明白 K&R C，对 ANSI C 也已知道了 90%。ANSI C 和 K&R C 的区别分成四大类，按其重要性分列于下：

1. 第一类区别是指一些新的、非常不同的、并且很重要的东西。惟一属于这类区别的特性是原型——把形参的类型作为函数声明的一部分。原型使得编译器很容易根据函数的定义检查函数的用法。

2. 第二类区别是一些新的关键字。ANSIC 正式增加了一些关键字：enum 代表枚举类型（最初出现于 pcc 的后期版本），const、volatile、signed、void 也有各自相关的语义。另外，原先可能由于疏忽而加入到 C 中的关键字 entry 则弃之不用。

3. 第三类区别被称作“安静的改变”——原先的有些语言特性仍然合法，但它的意思有了一些轻微的改变。这方面的例子很多，但都不是很重要，几乎可以被忽略。在你偶尔漫步于它们之上时，可能由于不注意而被其中一个绊了个趔趄。例如，现在的预处理规则定义得更加严格，有一条新规则，就是相邻的字符串字面值会被自动连接在一起。

4. 最后一类区别就是除上面 3 类之外的所有区别，包括那些在语言的标准化过程中长期争论的东西，这些区别在现实中几乎不可能碰到，如符号粘贴(token-pasting)和三字母词(trigraph)（三字母词就是用 3 个字符表示一个单独的字符，如果该字符不存在于某种计算机的字符集中，就可以用这 3 个字符来表示。比如两字母词(digraph)\t 表示“tab”，而三字母词??< 则表示“开放的花括号”）。

1.10 “安静的改变”究竟有多少安静

标准所作的修改并非都如原型那样引人注目。ANSI C 作了其他一些修改，目的是使 C 语言更加可靠。例如，“寻常算术转换(usual arithmetic conversion)”在旧式的 K&R C 和 ANSI C 中的意思就有所不同。Kernighan 和 Ritchie 当初是这样写的：

第 6.6 节：算术转换

许多运算符都会引发转换，以类似的方式产生结果类型。这个模式称为“寻常算术转换”。

首先，任何类型为 char 或 short 的操作数被转换为 int，任何类型为 float 的操作数被转换为 double。其次，如果其中一个操作数的类型是 double，那么另一个操作数被转换成 double，计算结果的类型也是 double。再次，如果其中一个操作数的类型是 long，那么另一个操作数被转换成 long，计算结果的类型也是 long。或者，如果其中一个操作数的类型是 unsigned，那么另一个操作数被转换成 unsigned，计算结果的类型也是 unsigned。如果不符合上面几种情况，那么两个操作数的类型都作为 int，计算结果的类型也是 int。

ANSI C 手册重新编写了有关内容，填补了其中的漏洞：

第 6.2.1.1 节 字符和整型（整型升级）

char, short int 或者 int 型位段(bit-field)，包括它们的有符号或无符号变型，以及枚举类型，可以使用在需要 int 或 unsigned int 的表达式中。如果 int 可以完整表示源类型的所有值[1]，那么该源类型的值就转换为 int，否则转换为 unsigned int。这称为整型升级。

第 6.2.1.5 节 寻常算术转换

许多操作数类型为算术类型的双目运算符会引发转换，并以类似的方式产生结果类型。它的目的是产生一个普通类型，同时也是运算结果的类型。这个模式称为“寻常算术转换”。

首先，如果其中一个操作数的类型是 long double，那么另一个操作数也被转换为 long double。其次，如果其中一个操作数的类型是 double，那么另一个操作数也被转换为 double。再次，如果其中一个操作数的类型是 float，那么另一个操作数也被转换为 float。否则，两个操作数进行整型升级（第 6.2.1.1 节描述整型升级），执行下面的规则：

如果其中一个操作数的类型是 unsigned long int，那么另一个操作数也被转换为 unsigned long int。其次，如果其中一个操作数的类型是 long int，而另一个操作数的类型是 unsigned int，

如果 long int 能够完整表示 unsigned int 的所有值[1]，那么 unsigned int 类型操作数被转换为 long int，如果 long int 不能完整表示 unsigned int 的所有值[2]，那么两个操作数都被转换为 unsigned long int。再次，如果其中一个操作数的类型是 long int，那么另一个操作数被转换为 long int。再再次，如果其中一个操作数的类型是 unsigned int，那么另一个操作数被转换为 unsigned int。如果所上以上情况都不属于，那么两个操作数都为 int。

浮点操作数和浮点表达式的值可以用比类型本身所要求的更大的精度和更广的范围来表示，而它的类型并不因此改变。

表示，而它的类型并不因此改变。

采用通俗语言（当然存有漏洞，而且不够精确），ANSI C 标准所表示的意思大致如下：

当执行算术运算时，操作数的类型如果不同，就会发生转换。数据类型一般朝着浮点精度更高、长度更长的方向转换，整型数如果转换为 signed 不会丢失信息，就转换为 signed，否则转换为 unsigned。

K&R C 所采用无符号保留(unsigned preserving)原则，就是当一个无符号类型与 int 或更小的整型混合使用时，结果类型是无符号类型。这是个简单的规则，与硬件无关。但是，正如下面的例子所展示的那样，它有时会使一个负数丢失符号位。

ANSI C 标准则采用值保留(value preserving)原则，就是当把几个整型操作数像下面这样混合使用时，结果类型有可能是有符号数，也可能是无符号数，取决于操作数的类型的相对大小。

下面的程序段分别在 ANSI C 和 K&R C 编译器中运行时，将打印出不同的信息：

```
main(){
  if(-1 < (unsigned char)1
    printf("-1 is less than (unsigned char)1: ANSI semantics ");
  else
    printf("-1 NOT less than (unsigned char)1: K&R semantics");
}
```

程序中的表达式在两种编译器下编译的结果不同。−1 的位模式是一样的，但一个编译器(ANSI C)将它解释为负数，另一个编译器(K&R C)却将它解释为无符号数，也就是变成了正数。