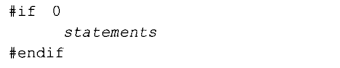
# C和指针

## C语言注释

注释时，符号/\*…..\*/之间不能在有其他的注释。且在C语言中，不能使用/\*…..\*/来注释代码，如果这段代码中原先就有注释存在，这样就会出问题。要从逻辑上删除这一段C代码，更好的方法就是使用#if指令。只要如下面使用：

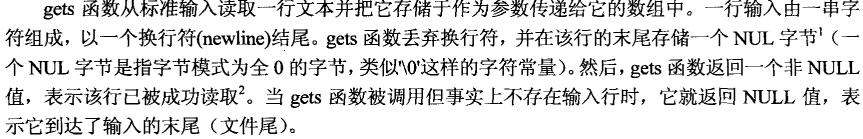


## 预处理指令

#include把包含的c文件导入到当前文件中，如果当前文件需要使用导入文件中的函数需要在当前文件前方引入函数原型，因此，可以在映入的文件中直接写上函数原型，那么在当前文件中也就不需要再写上函数院系了。

const表示这个函数不会修改函数调用者所传递的参数。

#define是另一种预处理，他将把名字替换为想要替换的东西。

C语言参数传值时，数组参数是传地址（引用）。而标量和常量则是传值。

C语言中并不存在字符串”String”,但在语言中，存在一项约定：字符串就是以NUL（全0字节）为结尾。字符串常量：“Hello”在内存中占据6个字节，按顺序分别是H、e、l、l、o和NUL。

&在C语言中被解释为取地址符号。&&是“逻辑与”操作。与C语言一样，&&操作连接的式子两端，如果左端为假，右边的表达式将不再进行求值，因此不管他是真是假，表达式总是假的。

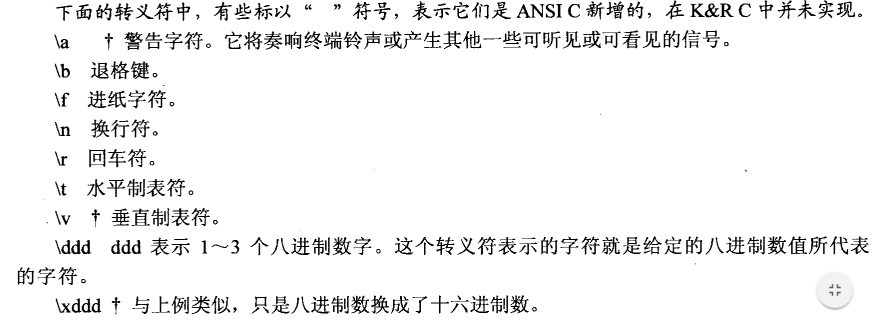
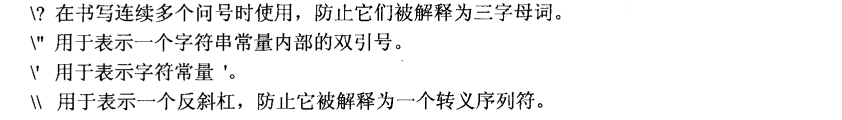
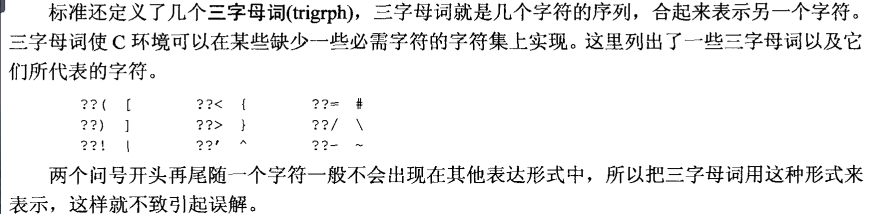
## 编译执行

编译过程也由几个阶段组成，首先是预处理器(preprocessor)处理。在这个阶段预处理器在源代码上执行一些文本操作。例如，用实际值代替#define指令定义的符号以及读入#include指令包含的文件内容。然后，源代码经过解析判断语句的意思。随后产生目标代码。之后优化器会对目标代码会进行进一步的处理，使他的效率更高。

## 文件名约定

C源代码以.c为扩展名保存。含有#include指令的C源代码称为头文件必须具有扩展名.h。

## 字符

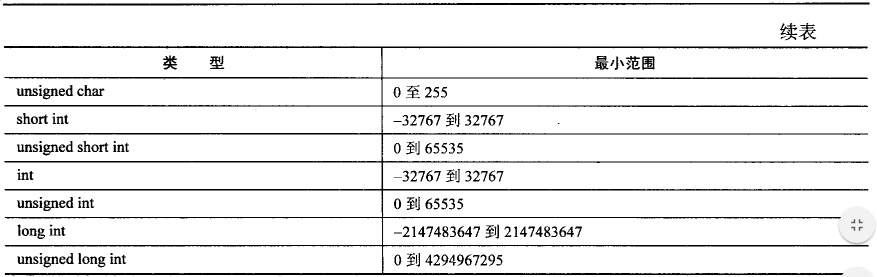
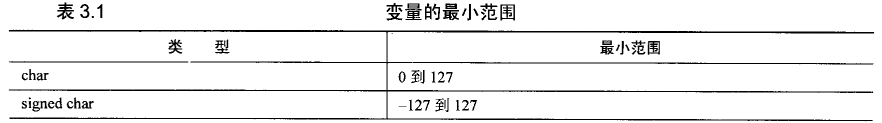


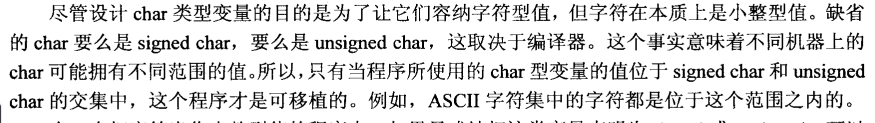
## 基本数据类型

四中基本数据累心：整型、浮点型、指针、和聚合类型(如数组和联合)

### 整形

整型家族包括字符、短整型、整型、和长整型，他们都有有符号和无符号数之分。而长整型至少应该和整型一样长，而整型至少应该和短整型一样长。

变量最小范围表。short int 至少16位(2 byte)，long int至少32位(4 byte)。Z至于缺省的int究竟是16位还是32位，或者是其他值则由编译器设计者决定。

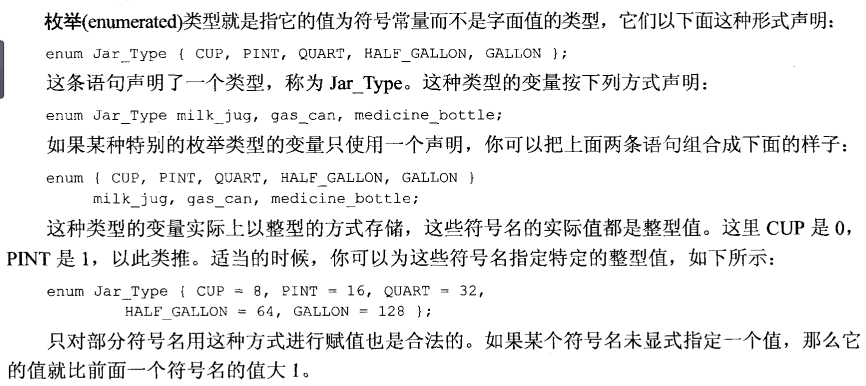
所以只有当程序所使用的char型变量位于Signed char和unsigned char的交集中，这个程序才能移植。例如，ASCII字符集中的字符都是位于这个范围之内的。

### 整型字面值

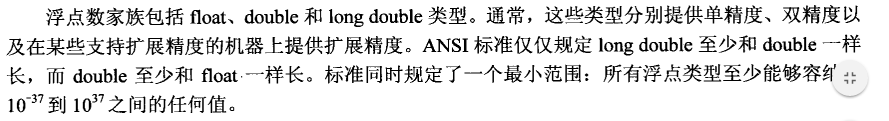
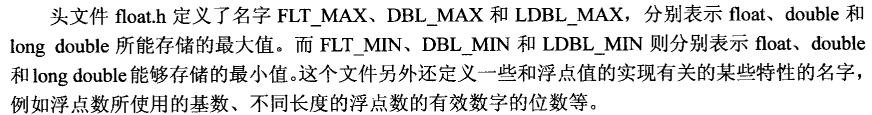
整型数字后面添加字符L或l，那么这个整数可以被解释为long整型值，字符U或u则用于把数值指定为Unsigned整型值。整型常量在缺省情况下，他是最短类型但能完整容纳这个值。整型也可以是八进制表示，只要数值前面以0开头。整数也可以是十六进制来表示，他以0x开头。

字符型常量也是允许的，例如’M’。最后，如果一个多字节字符常量前面有一个L，那么他就是宽字符常量，如L’x’、L’e^’。当运行时环境支持一个宽字节集时，就有可能使用他们。

## 枚举类型



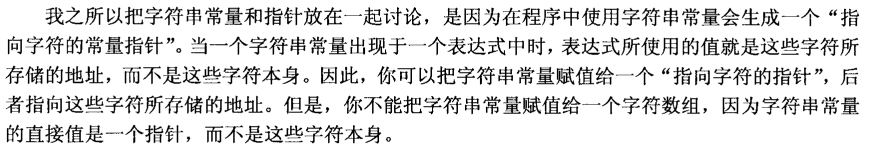
## 浮点类型

浮点数字字面值在缺省的情况下都是double类型的，除非他的后面跟着L或l表示他是一个long double类型的值，或者跟一个F或f表示他是一个float类型的。

## 指针

指针常量：

字符串常量：字符串常量通常被书写方式是一对引号保卫一串字符，如“”“Hello”、“\aWarning\a”。



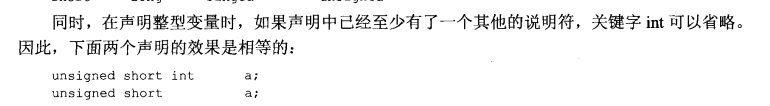
标准C函数库中包含一组函数，他们可以用于操纵字符串，包括字符串进行赋值、连接、比较、以及计算字符串长度和在字符串中查找某些特定的字符的函数。

### 数据类型声明

基本形式为：



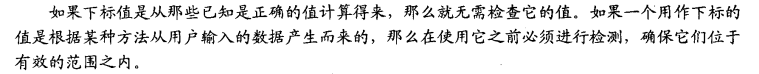
例如：unsigned long int i；



### 声明简单数组

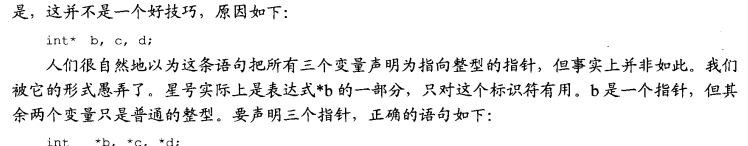
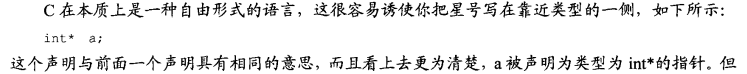
例如：int values[20];

对于数组的小标是否进行检验有如下的经验：

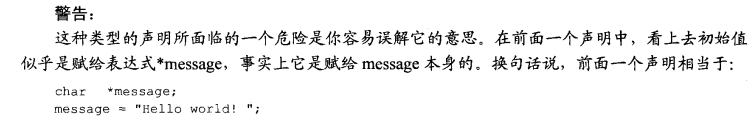


### 声明指针

例如:int \* a，这句表达式表示\*a产生的结果类型时int。知道了\*操作是间接访问操作后，我们可以断定a肯定是一个指向int的指针。

\*b是表达式的一部分，只对这个标识符有用。

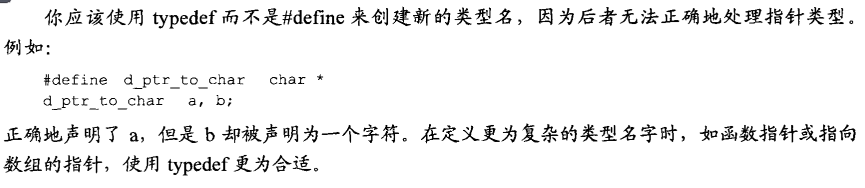
int \*a;\*可以理解为取地址符号，那么这里可以被理解为a指针所指的位置是一个int类型数据。



这里的意思是：message是一个指针，指向一个内存的地址，\*message表示的是取message所指示地址中存储的值，可以看出该地址中存储的是char类型的数据。Message = “Hello world”是表示将message这个指针指向“Hello world”这个字符串的地址。

## Typedef

允许你为各种数据类型定义新名字。例如：typedef char \*ptr\_to\_char这个声明是将ptr\_to\_char指针所指向的元素当做char的别名，这样你就可以使用ptr\_to\_char a 来声明一个指向字符的指针，这里是将char \*(即指向char的指针)生成另外的别名ptr\_to\_char。



这里#define是无条件的替换，这里就会替换成char\* a,b;这就变成上面声明指针时的一个例子。

## 常量

你可以使用const声明常量：const int a,int const a。你可以在初始化时对它进行初始化。另外当他被当做形参时，被调用是就会获得实参的值。

当涉及指针变量时，情况就会变得跟加又去，因为有两样东西就可能称为常量—指针变量和指针变量所指向的实体。下面有两个声明的例子：

int const \*pi : 可以看做const int \* pi，意思是你可以修改指针的值，但你不能修改他所指向的元素的值

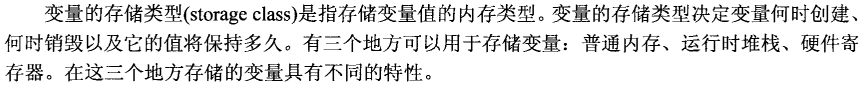
int \* const cpi :表示不可以修改指针指向的地址，但是可以修改指向的元素的值。

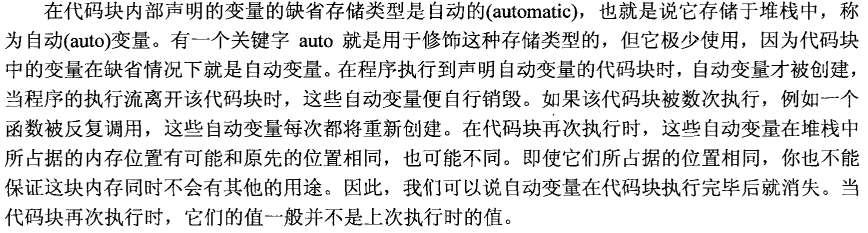
Int const \*const cpi :表示两者都不可进行修改。

## 作用域

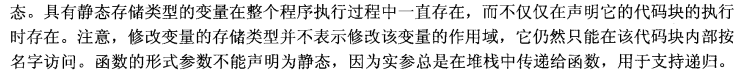
函数形参开始于形参的声明处，如果在函数体内声明了名字与形参相同的局部变量，他们就将隐藏形参。

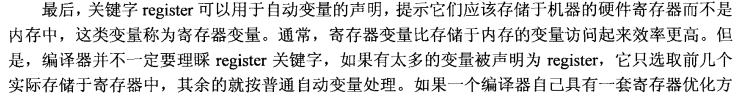
## 存储类型

变量的缺省值存储类型取决于他的声明位置。凡是任何代码块之外声明的变量总存储于静态内存中，也就是不属于堆栈的内存，这类变量称为静态变量。对于这类变量，你无法为他制定其他存储类型。

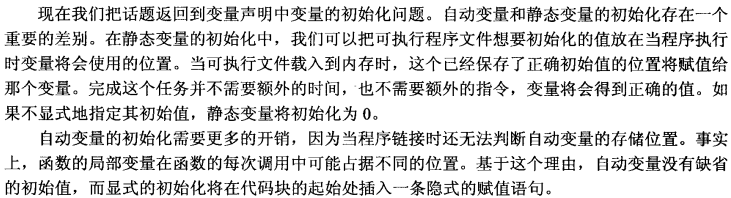


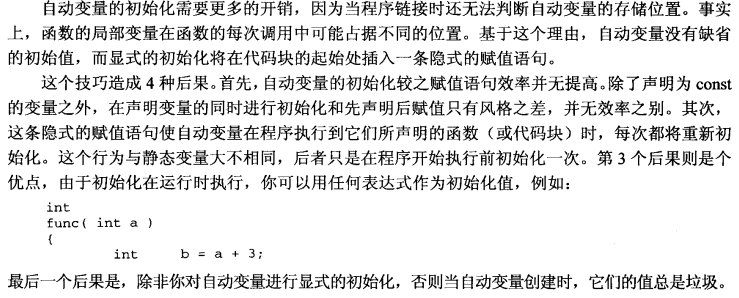
在代码块中内部声明的变量，如果给他加上关键字static，可以使他的存储类型从自动变为静态。

形参不能声明为静态变量，应为实参总是在堆栈中传递，用于支持递归。



### 初始化

这里需要注意：静态变量的初始化中，他的位置就已经确定了。当可执行载入内存时，这个已经保存了正确初始值的位置就赋值给这个变量。(这里也可以与Java对比，例如Java中A类中有一个static变量，那么当A类被类加载器加载之后就存在内存中---C中，static变量被提前装载到内存中)

这里的四个后果：1、即初始化效率与先声明后赋值效率差别不大2、隐式的访问语句每次被执行到时都会重新赋值3、这个是说自动变量不像静态变量在装载执行程序时就已经被初始化，他可以使用形参来赋值。

## Static关键字

