1. 函数声明要在使用函数之前进行。

1.1 void print\_name(); //一个ANSI C编译器会假设你没有使用函数原型声明函数，它就不会进行参数检查。

1.2 void print\_name(void);//ANSI C编译器会把这个解释为不接受任何参数。

1.3 可变参数：

va\_start宏涉及到C语言中可变参数的具体实现：

上面的宏一点一点说吧：

宏\_\_builtin\_saveregs ():

根据GCC文档中的描述，由于某些函数参数的传递是通过寄存器来的，为了使可变函数参数机制成功，该宏将寄存器中的参数复制到内存中。

宏\_\_va\_rounded\_size():

该宏定义为：#define \_\_va\_rounded\_size(TYPE) \

(((sizeof (TYPE) + sizeof (int) - 1) / sizeof (int)) \* sizeof

(int))va\_rounded\_size，好像有点复杂，实际上就是(X+K-1/K)\*K，这里实现了很多机器上的对齐机制，举个例子，假如X是6个字节，而K(int)是4个字节，上面的计算结果就是8个字节，也就是向4个字节宽度对齐。

那么，回到va\_start宏，这个宏就是计算了可变参数中的第一个可变参数的开始地址，&LASTARG求出最后一个固定参数的开始地址，再加上对齐后的LASTARG本身的长度便是紧接着LASTARG的第一个可变参数的地址了，再将该参数赋值给AP。

关于可变参数的更详细的解释见此：

http://hi.baidu.com/garymb/blog/item/0c8732125ae8c90e5aaf5396.html

关于GCC对\_\_builtin\_saveregs ()宏的描述：

http://www.delorie.com/gnu/docs/gcc/gccint\_135.html

2. C语言中参数传递方式：

2.1 值传递，

2.2 引用传递。

3. const 关键字，声明的时候初始化，因为声明了之后就不能修改 。

4. 数组和数组的初始化：

4.1

int test[12] ; //未初始化的数组，数组元素的值不会初始化，编译器使用的数值是存储单元中已有的数值。

int test[12] ={1,2,3,4,5} ; //部分初始化，其他的会被初始化为0 ；

int test[] ={1,2,3,4,5}; //未指定数组的大小、

int test[6]= {1,2,3,[4]=2121}; //C99 支持的新特性。

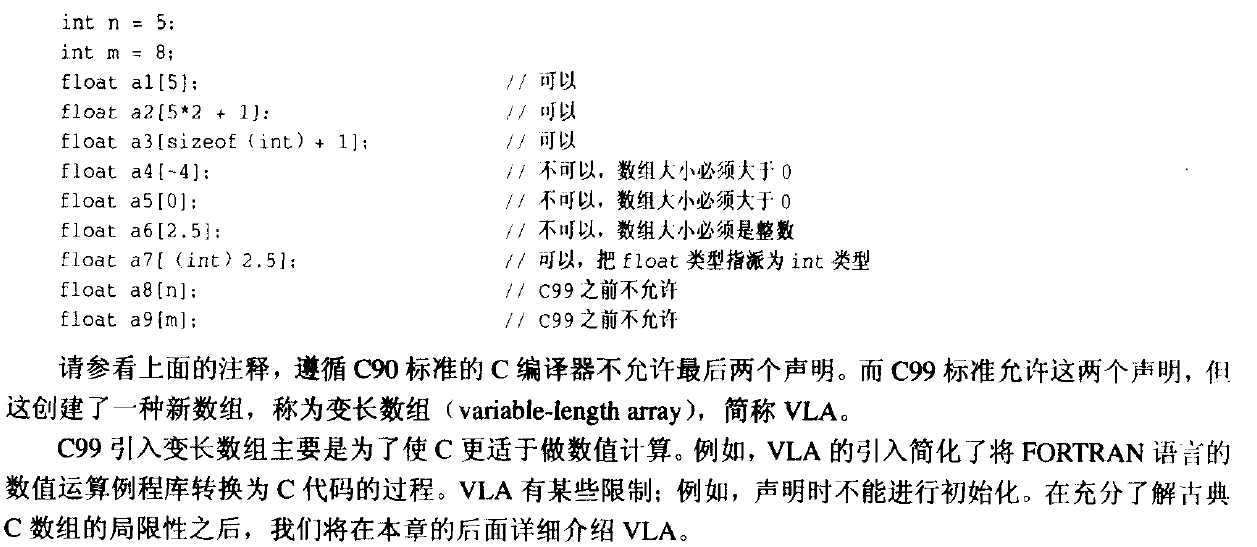
int test[6] ={1,2,[3]=12,13,14,[3]=33}; //C99 支持的新特性。

//注意：

C语言不支持把数组作为一个整体来赋值 。

C语言不支持用花括号括起来的列表形式进行赋值。(初始化的时候除外)。

C编译器不检查索引的合法性，

4.2 数组声明方式:   
  


**缩放比例 159%**

**缩放比例 159%**