一、strlen与sizeof的区别：

    1、strlen计算字符串的具体长度（只能是字符串），不包括字符串结束符。返回的是字符个数。

    2、sizeof计算声明后所占的内存数（空间数）（字节大小），不是实际长度。

    3、sizeof是一个取字节运算符，而strlen是个函数。

4、sizeof的返回值=字符个数\*字符所占的字节数，字符实际长度小于定义的长度，此时字符个数就等于定义的长度。若未给出定义的大小，分类讨论，对于字符串数组，字符大 小等于实际的字符个数+1；对于整型数组，字符个数为实际的字符个数。字符串每个字符占1个字节，整型数据每个字符占的字节数需根据系统的位数类确定，32位占4个字节。

1）sizeof操作符的结果类型是size\_t，它在头文件中typedef为unsigned int类型。  
该类型保证能容纳实现所建立的最大对象的字节大小。

2）sizeof是算符，strlen是函数。

3）sizeof可以用类型做参数，strlen只能用char\*做参数，且必须是以''\0''结尾的。  
    sizeof还可以用函数做参数，比如：   
    short f();  
    printf("%d\n", sizeof(f()));  
   输出的结果是sizeof(short)，即2。

4）数组做sizeof的参数不退化，传递给strlen就退化为指针了。

5）大部分编译程序在编译的时候就把sizeof计算过了,是类型或是变量的长度,这就是sizeof(x)可以用来定义数组维数的原因。   
   char str[20]="0123456789";  
   int a=strlen(str); //a=10;  
   int b=sizeof(str); //而b=20;

6）strlen的结果要在运行的时候才能计算出来，是用来计算字符串的长度，不是类型占内存的大小。

7）sizeof后如果是类型必须加括弧，如果是变量名可以不加括弧。这是因为sizeof是个操作符不是个函数。

8）当适用一个结构类型或变量时， sizeof 返回实际的大小；当适用一静态地空间数组， sizeof 归还全部数组的尺寸；sizeof 操作符不能返回动态地被分派了的数组或外部的数组的尺寸。

9）数组作为参数传给函数时传的是指针而不是数组，传递的是数组的首地址，  
    如：   
    fun(char [8])  
    fun(char [])  
    都等价于 fun(char \*)   
    在C++里参数传递数组永远都是传递指向数组首元素的指针，编译器不知道数组的大小。  
    如果想在函数内知道数组的大小， 需要这样做：  
    进入函数后用memcpy拷贝出来，长度由另一个形参传进去   
   fun(unsiged char \*p1, int len)  
  {  
       unsigned char\* buf = new unsigned char[len+1]  
       memcpy(buf, p1, len);  
   }  
  
看了上面的详细解释，发现两者的使用还是有区别的，从这个例子可以看得很清楚：  
char str[20]="0123456789";  
int a=strlen(str); //a=10; >>>> strlen 计算字符串的长度，以结束符 0x00 为字符串结束。  
int b=sizeof(str); //而b=20; >>>> sizeof 计算的则是分配的数组 str[20] 所占的内存空间的大小，不受里面存储的内容改变。    
  
上面是对静态数组处理的结果，如果是对指针，结果就不一样了  
char\* ss = "0123456789";  
sizeof(ss) 结果 4 ＝＝＝》ss是指向字符串常量的字符指针，sizeof 获得的是一个指针的之所占的空间,应该是长整型的，所以是4。  
sizeof(\*ss) 结果 1 ＝＝＝》\*ss是第一个字符 其实就是获得了字符串的第一位'0' 所占的内存空间，是char类型的，占了 1 位  
strlen(ss)= 10      ＝＝＝》 如果要获得这个字符串的长度，则一定要使用 strlen。strlen用来求字符串的长度；而sizeof是用来求指定变量或者变量类型等所占内存大小