C语言中之数学函数  
C语言提供了以下的数学函数，要使用这些函数时，在程序文件头必须加入：  
#include <math.h>  
编译时，必须加上参数「-lm」（表示连结至数学函式库），例如「gcc -lm test.c」。  
函数之自变量与传回之值型别见自变量或函数前之型别宣告。  
函数已经在「math.h」或其它标头档宣告过了，因此在使用时不必再加型别宣告，例如「y=sin(x);」，不用写成「y=double sin(double x);」。  
函数说明  
double sin(double x)  
x 的正弦函数值  
double cos(double x)  
x 的余弦函数值  
double tan(double x)  
x 的正切函数值  
double asin(double x)  
x 的反正弦函数值 sin-1x，x的值在 [-1,1] 之间，传回的值在 [-p/2,p/2] 之间  
double acos(double x)  
x 的反余弦函数值cos-1x，x的值在 [-1,1] 之间，传回的值在 [-p/2,p/2] 之间  
double atan(double x)  
x 的反正切函数值tan-1x，传回的值在 [-p/2,p/2] 之间  
double atan2(double y, double x)  
y/x 的反正切函数值tan-1(y/x)，传回的值在 [-p, p] 之间  
double sinh(double x)  
x 的双曲正弦函数值  
double cosh(double x)  
x 的双曲余弦函数值  
double tanh(double x)  
x 的双曲正切函数值  
double exp(double x)  
x 的指数函数 ex  
double log(double x)  
x 的自然对数 ln(x)，x > 0  
double log10(double x)  
x 底数为 10 的对数，log10x，x > 0  
double pow(double x, double y)  
x 的 y 次方 xy  
double sqrt(double x)  
x 的根号值 √x  
double ceil(double x)  
不小于 x 的最小整数（但其型别为 double）  
double floor(double x)  
不大于 x 的最大整数（但其型别为 double）  
int abs(int x)  
整数 x 的绝对值 |x|   
long labs(long x)  
长整数 x 的绝对值 |x|   
double fabs(double x)  
实数 x 的绝对值 |x|  
double ldexp(double x, int n)  
x?2n  
double fmod(double x, double y)  
x/y 的浮点数余数，符号与 x 相同  
范例：各个数学函式的使用方法  
#include <stdio.h>  
#include <math.h>  
#define PI 3.14159  
int main(void)  
{  
     double x,y,z;  
     int n;  
     x=4.0;  
     y=sqrt(x);  
     printf("x=%fty=%fn",x,y);  
     x=PI/4;  
     y=sin(x);  
     printf("x=%fty=%fn",x,y);  
     x=2.0;  
     y=3.0;  
     z=pow(x,y);  
     printf("x=%fty=%ftz=%fn",x,y,z);  
     x=1.5;  
     n=4;  
     y=ldexp(x,n);  
     printf("x=%ftn=%dty=%fn",x,n,y);  
     return 0;  
}

c++中 string与string.h 的作用和区别    
#include <string.h>     
  
void main()     
{     
  string aaa= "abcsd d";     
  printf("looking for abc from abcdecd %s\n", (strcmp(aaa,"abc")) ? "Found" : "Not Found");     
}     
不能正确执行，提示说是string类型没有定义     
而下面：     
#include <string>     
  
using namespace std;     
void main()     
{     
  string aaa= "abcsd d";     
  printf("looking for abc from abcdecd %s\n", (strcmp(aaa,"abc")) ? "Found" : "Not Found");     
}     
这里的string编译器就认识了，但是strcmp就不认识了呢？     
一般一个C++的老的带“.h”扩展名的库文件，比如iostream.h，在新标准后的标准库中都有一个不带“.h”扩展名的相对应，区别除了后者的好多改进之外，还有一点就是后者的东东都塞进了“std”名字空间中。     
  
但唯独string特别。     
问题在于C++要兼容C的标准库，而C的标准库里碰巧也已经有一个名字叫做“string.h”的头文件，包含一些常用的C字符串处理函数，比如楼主提到的strcmp。     
这个头文件跟C++的string类半点关系也没有，所以 <string>并非 <string.h>的“升级版本”，他们是毫无关系的两个头文件。     
要达到楼主的目的，比如同时：     
#include <string.h>     
#include <string>     
using namespace std;     
或者     
#include <cstring>     
#include <string>     
  
其 中 <cstring>是与C标准库的 <string.h>相对应，但裹有std名字空间的版本。 C++标准库很大。非常大。难以置信的大。怎么个大法？这么说吧：在C++ 标准中，关于标准库的规格说明占了密密麻麻300 多页，这还不包括标准C 库，后者只是"作为参考"（老实说，原文就是用的这个词）包含在C++库中。    
当然，并非总是越大越好，但在现在的情况下，确实越大越好，因为 大的库会包含大量的功能。标准库中的功能越多，开发自己的应用程序时能借助的功能就越多。C++库并非提供了一切（很明显的是，没有提供并发和图形用户接 口的支持），但确实提供了很多。几乎任何事你都可以求助于它。    
在归纳标准库中有些什么之前，需要介绍一下它是如何组织的。因为标准库中东西如 此之多，你（或象你一样的其他什么人）所选择的类名或函数名就很有可能和标准库中的某个名字相同。为了避免这种情况所造成的名字冲突，实际上标准库中的一 切都被放在名字空间std 中（参见条款28）。但这带来了一个新问题。无数现有的C++代码都依赖于使用了多年的伪标准库中的功能，例如，声明在 <iostream.h>， <complex.h>， <limits.h>等头文件中的功能。现有软件没有针对使用名字空间而进行设计，如果用std 来包装标准库导致现有代码不能用，将是一种可耻行为。（这种釜底抽薪的做法会让现有代码的程序员    
说出比"可耻" 更难听的话)慑于被激怒的程序员会产生的破坏力，标准委员会决定为包装了std 的那部分标准库构件创建新的头文件名。生成新头文件的方法仅仅是将现有C++头文件名中的.h 去掉，方法本身不重要，正如最后产生的结果不一致也并不重要一样。所以<iostream.h>变成了 <iostream>， <complex.h>变成了 <complex>等等。对于C 头文件，采用同样的方法，但在每个名字前还要添加一个c。所以C 的 <string.h>变成了<cstring>， <stdio.h>变成了 <cstdio>，等等。最后一点是，旧的C++头文件是官方所反对使用的（即，明确列出不再支持），但旧的C 头文件则没有（以保持对C 的兼容性）。实际上，编译器制造商不会停止对客户现有软件提供支持，所以可以预计，旧的C++头文件在未来几年内还是会被支持。    
所以，实际来说，下面是C++头文件的现状：    
旧的C++头文件名如 <iostream.h>将会继续被支持，尽管它们不在官方标 准中。这些头文件的内容不在名字空间std 中。    
新的C++头文件如 <iostream>包含的基本功能和对应的旧头文件相同，但头文件的内容在名字空间std 中。（在标准化的过程中，库中有些部分的细节被修改了，所以旧头文件和新头文件中的实体不一定完全对应。）    
标准C 头文件如 <stdio.h>继续被支持。头文件的内容不在std 中。    
具有C 库功能的新C++头文件具有如 <cstdio>这样的名字。它们提供的内容和相应的旧C 头文件相同，只是内容在std 中。    
所 有这些初看有点怪，但不难习惯它。最大的挑战是把字符串头文件理清楚： <string.h>是旧的C 头文件，对应的是基于char\*的字符串处理函数； <string>是包装了std 的C++头文件，对应的是新的string 类（看下文）； <cstring>是对应于旧C 头文件的std 版本。如果能掌握这些（我相信你能），其余的也就容易了。