函数名: scanf   
功 能: 执行格式化输入   
用 法: int scanf(char \*format[,argument,...]);

scanf()函数是通用终端格式化输入函数，它从标准输入设备(键盘) 读取输入的信息。可以读入任何固有类型的数据并自动把数值变换成适当的机内格式。

其调用格式为:      scanf("<格式化字符串>"，<地址表>);

scanf()函数返回成功赋值的数据项数，出错时则返回EOF。

其控制串由三类字符构成：

1。格式化说明符；  
2。空白符；  
3。非空白符；

（A）                格式化说明符

格式字符           说明

%a                 读入一个浮点值(仅C99有效)   
%A                 同上  
%c                 读入一个字符  
%d                 读入十进制整数  
%i                 读入十进制，八进制，十六进制整数  
%o                 读入八进制整数  
%x                 读入十六进制整数  
%X                 同上  
%c                 读入一个字符  
%s                 读入一个字符串  
%f                 读入一个浮点数  
%F                 同上  
%e                 同上  
%E                 同上  
%g                 同上  
%G                 同上  
%p                 读入一个指针  
%u                 读入一个无符号十进制整数  
%n                 至此已读入值的等价字符数  
%[]                扫描字符集合  
%%                 读%符号  
                  
附加格式说明字符表

修饰符                       说明

L/l 长度修饰符               输入"长"数据  
h 长度修饰符                 输入"短"数据  
W 整型常数                   指定输入数据所占宽度  
\* 星号                       空读一个数据   
hh,ll同上h,l但仅对C99有效。

（B）         空白字符

空白字符会使scanf()函数在读操作中略去输入中的一个或多个空白字符，空白符可以是space,tab,newline等等，直到第一个非空白符出现为止。

（C）        非空白字符

一个非空白字符会使scanf()函数在读入时剔除掉与这个非空白字符相同的字符。

注：scanf()控制串知识就介绍到这里（应该比较齐全了^\_^)，如有遗漏下次补上。下面将结合实际例程，一一阐述.

三、      scanf()函数的控制串的使用

例1.

#include "stdio.h"   
int main(void)   
{   
int a,b,c;   
  
scanf("%d%d%d",&a,&b,&c);   
printf("%d,%d,%d/n",a,b,c);

return 0;  
}

运行时按如下方式输入三个值：

3□4□5　↙（输入a,b,c的值）

3，4，5 （printf输出的a，b，c的值）

   （1） &a、&b、&c中的&是地址运算符，分别获得这三个变量的内存地址。  
   （2） "%d%d%d"是按十进值格式输入三个数值。输入时，在两个数据之间可以用一个或多个空格、tab键、回车键分隔。  
      以下是合法输入方式：   
      ① 3□□4□□□□5↙  
      ② 3↙  
         4□5↙  
      ③ 3（tab键）4↙  
         5↙

例2.

#include "stdio.h"   
int main(void)   
{   
int a,b,c;

scanf("%d,%d,%d",&a,&b,&c);  
printf("%d,%d,%d/n",a,b,c);

return 0;  
}

运行时按如下方式输入三个值：

3,4,5　↙（输入a,b,c的值）

或者

3,□4,□5　↙（输入a,b,c的值）

3,□□□4,□5　↙（输入a,b,c的值）  
......  
都是合法的，但是","一定要跟在数字后面，如：  
3□，4,□5　↙就非法了，程序出错。（解决方法与原因后面讲）

再如：

1、sacnf()中的变量必须使用地址。

　　       int a, b;  
scanf("%d%d",a,b); //错误  
scanf("%d%d",&a,&b);

2、scanf()的格式控制串可以使用其它非空白字符，但在输入时必须输入这些字符。

例：  
scanf("%d,%d",&a,&b);   
输入： 3，4 ↙（逗号与"%d,%d"中的逗号对应）   
scanf("a=%d,b=%d",&a,&b);   
输入： a=3，b=4 ↙（"a=","b=",逗号与"%d,%d"中的"a=","b="及逗号对应）

3、在用"%c"输入时，空格和“转义字符”均作为有效字符。

例：  
scanf("%c%c%c",&c1,&c2,&c3);   
输入：a□b□c↙   
结果：a→c1，□→c2，b→c3 (其余被丢弃)

scanf()函数接收输入数据时，遇以下情况结束一个数据的输入：（不是结束该scanf函数，scanf函数仅在每一个数据域均有数据，并按回车后结束）。  
        ① 遇空格、“回车”、“跳格”键。  
        ② 遇宽度结束。  
        ③ 遇非法输入。

问题二：scanf()函数不能正确接受有空格的字符串？如: I love you!

#include <stdio.h>  
int main()  
{  
    char str[80];  
      
    scanf("%s",str);  
    printf("%s",str);

    return 0;  
}

输入：I live you!  
输出：I

       scanf()函数接收输入数据时，遇以下情况结束一个数据的输入：（不是结束该scanf函数，scanf函数仅在每一个数据域均有数据，并按回车后结束）。  
        ① 遇空格、“回车”、“跳格”键。  
        ② 遇宽度结束。  
        ③ 遇非法输入。

所以，上述程序并不能达到预期目的，scanf()扫描到"I"后面的空格就认为对str的赋值结束，并忽略后面的"love you!".这里要注意是"love you!"还在键盘缓冲区（关于这个问题，网上我所见的说法都是如此，但是，我经过调试发现，其实这时缓冲区字符串首尾指针已经相等了，也就是说缓冲区清空了，scanf()函数应该只是扫描stdin流，这个残存信息是在stdin中)。我们改动一下上面的程序来验证一下：

#include <stdio.h>  
int main()  
{  
    char str[80];  
    char str1[80];  
    char str2[80];  
      
    scanf("%s",str);/\*此处输入:I love you! \*/  
    printf("%s",str);  
    sleep(5);/\*这里等待5秒,告诉你程序运行到什么地方\*/  
    scanf("%s",str1);/\*这两句无需你再输入,是对键盘盘缓冲区再扫描   \*/  
    scanf("%s",str2);/\*这两句无需你再输入,是对键盘盘缓冲区再扫描    \*/  
    printf("/n%s",str1);  
    printf("/n%s",str2);  
    return 0;  
}

输入：I love you!  
输出：I  
      love  
      you!

好了，原因知道了，那么scanf()函数能不能完成这个任务？回答是：能！别忘了scanf()函数还有一个 %[] 格式控制符（如果对%[]不了解的请查看本文的上篇）,请看下面的程序：

#include "stdio.h"  
int main()  
{  
    char string[50];  
      
     /\*scanf("%s",string);不能接收空格符\*/  
     scanf("%[^/n]",string);  
     printf("%s/n",string);  
     return 0;  
}

问题三：键盘缓冲区残余信息问题

#include <stdio.h>  
int main()  
{  
    int a;  
    char c;

    do  
    {  
        scanf("%d",&a);  
        scanf("%c",&c);  
        printf("a=%d     c=%c/n",a,c);  
        /\*printf("c=%d/n",c);\*/  
    }while(c!='N');  
}

   scanf("%c",&c);这句不能正常接收字符,什么原因呢？我们用printf("c=%d/n",c);将C用int表示出来，启用printf("c=%d/n",c);这一句，看看scanf()函数赋给C到底是什么，结果是 c=10 ,ASCII值为10是什么？换行即/n.对了，我们每击打一下"Enter"键，向键盘缓冲区发去一个“回车”(/r),一个“换行"(/n),在这里/r被scanf()函数处理掉了（姑且这么认为吧^\_^），而/n被scanf()函数“错误”地赋给了c.

解决办法：可以在两个scanf()函数之后加个fflush(stdin);，还有加getch(); getchar();也可以，但是要视具体scanf()语句加那个，这里就不分析了，读者自己去摸索吧。但是加fflush(stdin);不管什么情况都可行。

函数名: fflush   
功 能: 清除一个流   
用 法: int fflush(FILE \*stream);

#include <stdio.h>  
int main()  
{  
    int a;  
    char c;

    do  
    {  
        scanf("%d",&a);  
        fflush(stdin);  
        scanf("%c",&c);  
        fflush(stdin);  
        printf("a=%d     c=%c/n",a,c);

    }while(c!='N');  
}

这里再给一个用“空格符”来处理缓冲区残余信息的示例：

运行出错的程序：

#include <stdio.h>  
int main()  
{  
    int i;  
    char j;  
    for(i = 0;i < 10;i++)  
    {  
        scanf("%c",&j);/\*这里%前没有空格\*/  
    }  
}

使用了空格控制符后：

#include <stdio.h>  
int main()  
{  
    int i;  
    char j;  
    for(i = 0;i < 10;i++)  
    {  
        scanf(" %c",&j);/\*注意这里%前有个空格\*/  
    }  
}

    可以运行看看两个程序有什么不同。

问题四   如何处理scanf()函数误输入造成程序死锁或出错？

#include <stdio.h>  
int main()  
{  
int a,b,c; /\*计算a+b\*/

scanf("%d,%d",&a,&b);  
c=a+b;  
printf("%d+%d=%d",a,b,c);  
}

如上程序，如果正确输入a,b的值，那么没什么问题，但是，你不能保证使用者每一次都能正确输入，一旦输入了错误的类型，你的程序不是死锁，就是得到一个错误的结果,呵呵，这可能所有人都遇到过的问题吧？

解决方法：scanf()函数执行成功时的返回值是成功读取的变量数,也就是说，你这个scanf()函数有几个变量，如果scanf()函数全部正常读取，它就返回几。但这里还要注意另一个问题，如果输入了非法数据，键盘缓冲区就可能还个有残余信息问题。

正确的例程：

#include <stdio.h>  
int main()  
{  
int a,b,c; /\*计算a+b\*/

while(scanf("%d,%d",&a,&b)!=2)fflush(stdin);  
c=a+b;  
printf("%d+%d=%d",a,b,c);  
}

scanf函数探讨   
  
1.空白符问题   
#include <stdio.h>   
main()   
{   
int a;   
printf( "input the data/n ");   
scanf( "%d/n ",&a);//这里多了一个回车符/n   
printf( "%d ",a);   
return 0;   
}   
结果要输入两个数程序才结束，而不是预期的一个。why？   
原因：用空白符结尾时，scanf会跳过空白符去读下一个字符，所以你必须再输入一个数。这里的空白符包括   
  
空格，制表符，换行符，回车符和换页符。所以如果你用scanf( "%d ",&a)也会出现同样的问题。   
解决方法：这种错误大多是输入的时候不小心，多注意一点就好了。这种问题也不好检查，编译没有问题，   
  
一个空格也不容易看出来。当你的程序出现上面的问题时，自己对照检查一下就可以了。   
  
  
2.缓冲区问题   
这是一个非常容易错的地方，我就错过多次。   
#include <stdio.h>   
main()   
{   
int n = 5;   
char c[n];   
for(int i = 0; i < n; i++)   
c[i] = scanf( "%c ",&c[i]);   
printf(c);   
return 0;   
}   
如果输入:   
a   
b   
c   
那么循环就会“提前”结束了.   
原因：输入a和第一个回车后，a和这个回车符都留在缓冲区中。第一个scanf读取了a，但是输入缓冲区里面   
  
还留有一个/n，第二个scanf读取这个/n。然后输入b和第二个回车，同样的，第三个scanf读取了b，第四个   
  
scanf读取了第二个回车符。第五个读取了c。所以五个scanf都执行了，并没有提前结束。只不过有的scanf   
  
读取到了回车符而已。   
解决方法：把程序改成这样就可以了：   
for( i = 0; i < n; i++){   
scanf( "%c ",&c[i]);   
fflush(stdin);//刷新缓冲区   
}   
或者不用scanf，而用gets（）函数，如：   
#include <stdio.h>   
main()   
{   
char c[5];   
gets(c);   
printf(c);   
return 0;   
}   
但要注意：这个函数自动把你最后敲的回车转换为字符 '/0 '。如果你的输入超过了数组的大小，那么就会产   
  
生错误。   
  
  
3.scanf()函数的参数输入类型不匹配问题   
这是我在csdn论坛上见到的问题，这个错误有时候会让人莫名其妙。   
#include <stdio.h>   
main()   
{   
int a=123;   
char c= 't ';   
printf( "input/n ");   
scanf( "%d%c ",&a,&c);   
scanf( "%d%c ",&a,&c);   
scanf( "%d%c ",&a,&c);   
printf( "%d/n%c/n ",a,c);   
return 0;   
}   
当输入a 回车 后，会直接跳过下面２个scanf语句，直接输出为   
123   
t   
原因：对于scanf( "%d%c ",&a,&c)，scanf语句执行时，首先试图从缓冲区中读入一个%d类型的数据，如果和   
  
第一个参数匹配，则继续从缓冲区中读取数据和第二个参数进行匹配，依次进行下去，直到匹配完所有的参   
  
数；如果其中有一个参数不匹配，那就从这个地方跳出，忽略这个scanf后面所有的参数，而去执行下一条语   
  
句。   
可以用下面的程序验证一下：   
#include <stdio.h>   
int main()   
{   
int a=123,b=1;   
char c= 't ';   
scanf( "%d%d ",&a,&b);   
scanf( "%c ",&c);   
printf( "%d/n%d/n%c/n ",a,b,c);   
return 0;   
}输入：2 回车a 回车   
结果是：   
2   
1   
a   
解决方法：scanf()函数执行成功时的返回值是成功读取的变量数,也就是说，你这个scanf()函数有几个变量   
  
，如果scanf()函数全部正常读取，它就返回几。但这里还要注意另一个问题，如果输入了非法数据，键盘缓   
  
冲区就可能还个有残余信息问题。   
比如：   
#include <stdio.h>   
main()   
{   
int a=123,b;   
while(scanf( "%d%d ",&a,&b)!=2)   
fflush(stdin);   
printf( "%d/n%d/n ",a,b);   
return 0;   
}   
你可以试一下，如果输入不是数字时，会有什么反应。   
  
补充：scanf中一种很少见但很有用的转换字符：[...]和[ ^...]。   
#include <stdio.h>   
main()   
{   
char strings[100];   
scanf( "%[1234567890] ",strings);   
printf( "%s ",strings);   
return 0;   
}   
运行，输入：1234werew后，结果是：1234。   
通过运行可以发现它的作用是：如果输入的字符属于方括号内字符串中某个字符，那么就提取该字符；如果   
  
一经发现不属于就结束提取。该方法会自动加上一个字符串结束符到已经提取的字符后面。   
scanf( "%[^1234567890] ",strings); 它的作用是：如果一经发现输入的字符属于方括号内字符串中某个字符   
  
，那么就结束提取；如果不属于就提取该字符。该方法会自动加上一个字符串结束符到已经提取的字符后面   
  
。   
注意：方括号两边不能空格，如：scanf( "%[ 1234567890 ] ",strings); scanf( "%[ ^1234567890   
  
] ",strings); 不让空格也会算在里面的。   
用这种方法还可以解决scanf的输入中不能有空格的问题。只要用   
scanf( "%[^/n] ",strings); 就可以了。很神奇吧。   
  
  
scanf原型:参见《C语言大全》和K&C   
# include <stdio.h> ;   
int scanf( const char \*format, ... );   
函数 scanf() 是从标准输入流 stdin 中读内容的通用子程序，可以读入全部固有类型的数据并自动转换成机内形式。

在 C99 中，format 用 restrict 修饰。   
format 指向的控制串由以下三类字符组成：   
● 格式说明符   
● 空白符   
● 非空白符   
  
转换字符(就是%后跟的部分）   
a 读浮点值(仅适用于 C99)   
A 读浮点值(仅适用于 C99)   
c 读单字符   
d 读十进制整数   
i 读十进制、八进制、十六进制整数   
e 读浮点数   
E 读浮点数   
f 读浮点数   
F 读浮点数(仅适用于 C99)   
g 读浮点数   
G 读浮点数   
o 读八进制数   
s 读字符串   
x 读十六进制数   
X 读十六进制数   
p 读指针值   
n 至此已读入值的等价字符数   
u 读无符号十进制整数   
[ ] 扫描字符集合   
% 读 % 符号(百分号)   
  
例如： %s 表示读串而 %d 表示读整数。格式串的处理顺序为从左到右，格式说明符逐一与变元表中的   
  
变元匹配。为了读取长整数，可以将 l(ell) 放在格式说明符的前面；为了读取短整数，可以将 h 放在格式   
  
说明符的前面。这些修饰符可以与 d、i、o、u 和 x 格式代码一起使用。   
  
默认情况下，a、f、e 和 g 告诉 scanf() 为 float 分配数据。 如果将 l(ell) 放在这些修饰符的前   
  
面，则 scanf() 为 double 分配数据。使用 L 就是告诉 scanf()，接收数据的变量是 long double 型变量。   
  
如果使用的现代编译器程序支持 1995 年增加的宽字符特性， 则可以与 c 格式代码一起，用 l 修饰符   
  
说明类型 wchar\_t 的宽字符指针；也可以与 s 格式代码一起，用 l 修饰符说明宽字符串的指针。l 修饰符   
  
也可以用于修饰扫描集，以说明宽字符。   
  
控制串中的空白符使 scanf() 在输入流中跳过一个或多个空白行。空白符可以是空格(space)、制表符   
  
(tab)和新行符(newline)。 本质上，控制串中的空白符使 scanf() 在输入流中读，但不保存结果，直到发   
  
现非空白字符为止。   
  
非空白符使 scanf() 在流中读一个匹配的字符并忽略之。例如， "%d,%d " 使 scanf() 先读入一个整数   
  
，读入中放弃逗号，然后读另一个整数。如未发现匹配，scanf() 返回。   
  
scanf() 中用于保存读入值的变元必须都是变量指针，即相应变量的地址。   
  
在输入流中，数据项必须由空格、制表符和新行符分割。逗号和分号等不是分隔符，比如以下代码：   
scanf( "%d %d ", &r, &c );   
将接受输入 10 20，但遇到 10,20 则失败。   
  
百分号(%)与格式符之间的星号(\*)表示读指定类型的数据但不保存。因此，   
scanf( "%d %\*c %d ", &x, &y );   
对 10/20 的读入操作中，10 放入变量 x，20 放入 y。   
  
格式命令可以说明最大域宽。 在百分号(%)与格式码之间的整数用于限制从对应域读入的最大字符数。   
  
例如，希望向 address 读入不多于 20 个字符时，可以书写成如下形式：   
scanf( "%20s ", address );   
  
如果输入流的内容多于 20 个字符，则下次 scanf() 从此次停止处开始读入。 若达到最大域宽前已遇   
  
到空白符，则对该域的读立即停止；此时，scanf() 跳到下一个域。   
  
虽然空格、制表符和新行符都用做域分割符号，但读单字符操作中却按一般字符处理。例如，对输入流   
  
"x y " 调用：   
scanf( "%c%c%c ", &a, &b, &c );   
返回后，x 在变量 a 中，空格在变量 b 中，y 在变量 c 中。   
  
注意，控制串中的其它字符，包括空格、制表符和新行符，都用于从输入流中匹配并放弃字符，被匹配   
  
的字符都放弃。例如，给定输入流 "10t20 "，调用：   
scanf( "%dt%d ", &x, &y );   
将把 10 和 20 分别放到 x 和 y 中，t 被放弃，因为 t 在控制串中。   
  
ANSI C 标准向 scanf() 增加了一种新特性，称为扫描集(scanset)。 扫描集定义一个字符集合，可由   
  
scanf() 读入其中允许的字符并赋给对应字符数组。 扫描集合由一对方括号中的一串字符定义，左方括号前   
  
必须缀以百分号。 例如，以下的扫描集使 scanf() 读入字符 A、B 和 C：   
%[ABC]   
  
使用扫描集时，scanf() 连续吃进集合中的字符并放入对应的字符数组，直到发现不在集合中的字符为   
  
止(即扫描集仅读匹配的字符)。返回时，数组中放置以 null 结尾、由读入字符组成的字符串。   
  
用字符 ^ 可以说明补集。把 ^ 字符放为扫描集的第一字符时，构成其它字符组成的命令的补集合，指   
  
示 scanf() 只接受未说明的其它字符。   
对于许多实现来说，用连字符可以说明一个范围。 例如，以下扫描集使 scanf() 接受字母 A 到 Z：   
%[A-Z]   
重要的是要注意扫描集是区分大小写的。因此，希望扫描大、小写字符时，应该分别说明大、小写字母   
  
。   
scanf() 返回等于成功赋值的域数的值，但由于星号修饰符而读入未赋值的域不计算在内。给第一个域   
  
赋值前已出错时，返回 EOF。   
  
C99 为 scanf() 增加了几个格式修饰符：hh、ll、j、z 和 t。hh 修饰符可用于 d、i、o、u、x、X 或   
  
n。它说明相应的变元是 signed 或 unsigned char 值，或用于 n 时， 相应的变元是指向 long char 型变   
  
量的指针。ll 修饰符也可用于 d、i、o、u、x、X 或 n。它说明相应的变元是 signed 或者 unsigned long   
  
long int 值。   
j 格式修饰符应用于 d、i、o、u、x、X 或 n，说明匹配的变元是类型 intmax\_t 或 uintmax\_t。这些   
  
类型在 <stdint.h> ; 中声明，并说明最大宽度的整数。   
z 格式修饰符应用于 d、i、o、u、x、X 或 n，说明匹配的变元是指向 size\_t 类型对象的指针。该类   
  
型在 <stddef.h> ; 中声明，并说明 sizeof 的结构。   
t 格式修饰符应用于 d、i、o、u、x、X 或 n，说明匹配的变元是指向 ptrdiff\_t 类型对象的指针。   
  
该类型在 <stddef.h> ; 中声明，并说明两个指针之间的差别。   
  
例子：   
  
# include <stdio.h> ;   
int main( void )   
{   
char str[80], str2[80];   
int i;   
/\* read a string and a integer \*/   
scanf( "%s%d ", str, &i );   
/\* read up to 79 chars into str \*/   
scanf( "%79s ", str );   
/\* skip the integer between the two strings \*/   
scanf( "%s%\*d%s ", str, str2 );   
return 0;   
}你的问题在这个部分：   
3.scanf()函数的参数输入类型不匹配问题   
这是我在csdn论坛上见到的问题，这个错误有时候会让人莫名其妙。   
#include <stdio.h>   
main()   
{   
int a=123;   
char c= 't ';   
printf( "input/n ");   
scanf( "%d%c ",&a,&c);   
scanf( "%d%c ",&a,&c);   
scanf( "%d%c ",&a,&c);   
printf( "%d/n%c/n ",a,c);   
return 0;   
}   
当输入a 回车 后，会直接跳过下面２个scanf语句，直接输出为   
123   
t   
原因：对于scanf( "%d%c ",&a,&c)，scanf语句执行时，首先试图从缓冲区中读入一个%d类型的数据，如果和   
  
第一个参数匹配，则继续从缓冲区中读取数据和第二个参数进行匹配，依次进行下去，直到匹配完所有的参   
  
数；如果其中有一个参数不匹配，那就从这个地方跳出，忽略这个scanf后面所有的参数，而去执行下一条语   
  
句。   
可以用下面的程序验证一下：   
#include <stdio.h>   
int main()   
{   
int a=123,b=1;   
char c= 't ';   
scanf( "%d%d ",&a,&b);   
scanf( "%c ",&c);   
printf( "%d/n%d/n%c/n ",a,b,c);   
return 0;   
}输入：2 回车a 回车   
结果是：   
2   
1   
a   
解决方法：scanf()函数执行成功时的返回值是成功读取的变量数,也就是说，你这个scanf()函数有几个变量   
  
，如果scanf()函数全部正常读取，它就返回几。但这里还要注意另一个问题，如果输入了非法数据，键盘缓   
  
冲区就可能还个有残余信息问题。   
比如：   
#include <stdio.h>   
main()   
{   
int a=123,b;   
while(scanf( "%d%d ",&a,&b)!=2)   
fflush(stdin);   
printf( "%d/n%d/n ",a,b);   
return 0;   
}