**基本的标准输入**

在C语言的标准I/O库函数中，与标准输出相对应的三个标准输入函数是getchar、gets以及scanf。其中，getchar用来输入单个字符，gets用来输入一行字符串，scanf处理格式输入。这三个标准输入函数都是从标准输入设备读入设备读取数据，功能各不相同。标准输入设备是键盘输入缓冲区，其中字符流形式的数据称为输入流。

**1. 单字符输入函数getchar**

**1.1 函数原型**

getchar函数的原型为：

int getchar(void);

该函数返回值的类型为int，参数表中的void表示函数不需要参数。

**1.2 函数调用形式**

getchar函数的调用形式为：

getchar()

getchar函数的执行流程为：首先检查输入流中是否有字符；若有则读取输入流中的第一个字符作为函数返回值，程序向后执行。若输入流中没有字符，则getchar函数将进入等待输入状态；此时，可用键盘输入字符，按下回车键结束输入。比如，在键盘上输入abc，则所键入的字符‘a’, ‘b’,‘c’以及按下的回车键所产生的换行字符‘\n’依次被送入输入流。按下回车键同时会激活处于等待状态的getchar函数，getchar函数从输入流读取第一个字符‘a’，将其值转换为int类型后作为函数返回值返回，程序继续向后执行。剩余字符‘b’，‘c’和‘\n’留存在输入流中。

【示例】在键盘上输入字符，若输入的字符是小写字母，则按照大写输出，否则直接输出。

#include <stdio.h>

int main(int argc, char \*argv[])

{

char c;

while ((c = getchar()) != EOF)

{

putchar((c >= 'a' && c <= 'z') ? c - 'a' + 'A' : c);

}

return 0;

}

**2. 字符串输入函数gets**

**2.1 函数原型**

gets函数的原型为：

char \*gets(char \*s);

该函数的返回值类型为char \*，参数s是一个字符指针，指向存放输入字符串的内存缓冲区的首地址。s所指向的内存缓冲区的长度应足够大，为满足极端情况下字符串输入的需要，缓冲区的长度应该不小于81个字节，以容纳输入的一行字符及一个空字符‘\0’。

**2.2 函数的调用形式**

gets函数的调用形式为：

gets(s)

gets函数从输入流中读取一行字符存放到s指定的内存缓冲区，结尾的换行字符‘\n’被空字符‘\0’所替换，以作为字符串的结束标志。输入流中的一行字符是从输入流的第一个字符开始直到其后的第一个换行字符‘\n’为止的字符序列。函数正确执行时返回该内存缓冲区的首地址，即s的值；如果遇到文件尾或出错，则返回空指针NULL。

注意：如果需要输入一个带空白字符的字符串，则必须使用gets函数来实现。

**3. 格式输入函数scanf**

**3.1 函数原型**

scanf函数的原型为：

int scanf(const char \* format, …);

scanf函数的返回值类型为int。参数表中，format是格式字符串，用来指定输入数据的数目、类型和格式，“,…”表示其他参数数目是可变的，其他参数必须是用来存放输入数据的内存地址。

**3.2 函数的调用形式**

scanf函数执行的是printf函数的逆操作，它从标准输入设备读取字符流，并按照格式字符串由转换字符规定的格式转换成相应类型的值后，存放到输入参数指定的内存单元。scanf函数正确执行时，返回值为被转换并赋值的数据的个数；遇到文件尾或出错时返回EOF。

scanf函数的调用形式为：

scanf(格式字符串,输入参数1,输入参数2,…,输入参数n)

调用scanf函数一般至少要读入一个数据，因此实际参数除格式字符串外至少应有一个输入参数。输入参数1至输入参数n可以为基本类型或指针类型变量的地址（即指针）。用于输入字符串数据的参数应该为字符类型的指针，可以是字符数组名或指向字符数据首元素的指针变量。此外，输入参数在类型、数目和次序上应与格式字符串中的转换说明一致。

scanf函数的格式字符串与printf函数相似，在组成上可以包含普通字符和转换说明，并且具体包含以下三类字符：

(1) 空白字符。包括如空格符、换行符、制表符（对数据的输入来说没有影响）。

(2) 非%且非空白字符的普通字符（在输入流中相应位置必须有相同的字符与之匹配）。

(3) 以%开头，以转换字符为结尾的转换说明，形式为：

%[选项]转换字符

常用的scanf函数转换字符如下表所示：

常用的scanf转换字符

转换字符 参数类型 输入数据

d int \* 十进制整数

i int \* 整数；可以是八进制（有前导0）或十六进制 （有前缀0x或0X）

o int \* 八进制整数（有无前导0均可）

u unsigned

int \* 无符号十进制整数

x int \* 十六进制整数（可无前缀）

c char \* 字符；输入字符数由域宽给定，未指定域宽 时输入一个字符，参数可为字符或int变量的 地址，存放时不在尾部添加‘\0’。不跳过输入 流中的空白字符。若需读入一非空白字符， 可使用%1s

s char \* 无空白字符的字符串（不加引号）；跳过空白 字符，尾部添加‘\0’

e, f, g float \* 浮点数；可以无符号，可以无小数点，也可以 无指数部分

p void \* 指针值

% 无参数 %；不赋值

在实际应用中，scanf函数的格式字符串一般只需包含转换说明。因为，对于scanf函数格式字符串中除空格和制表符外的其他普通字符，在输入流中相应位置必须输入相同的字符，否则scanf函数读不到正确的数据（系统不做检查）。如果在scanf函数的格式字符串中加入了除了空格以外的普通字符，则不仅给数据输入带来麻烦，而且容易出错。可是，如果在格式字符串中每个转换说明之间适当添加空格或制表符（按Tab键输入），则可以使转换说明看上去清晰明了，同时对数据的输入没有影响。

在同一行上输入数据时，数据之间可用一个到多个空格或制表符分隔，按回车键结束输入。还可以在多行上输入以上数据，scanf函数一直等到输入数据个数达到转换说明个数才开始执行。

如果转换说明为%%，则不从输入流读取数据，同时，对应参数指出的变量也将不被赋值；如果在其他完整的转换说明前面有两个相邻的%，则输入时需要一个%与之匹配。

转换说明与输入参数不一致时可能导致以下结果：如果转换说明与输入参数的类型不匹配，则会导致读入的数据值不正确或程序非正常终止；如果转换说明的个数比输入数据的个数少，则无对应转换说明的变量不被赋值；如果转换说明的个数比输入参数的个数多，则可能死机。

转换说明%c不会跳过输入流中的空白字符，而%s则可以跳过，转换说明%1s可以起到从输入流读取一个非空白字符的作用。

**3.3 输入形式**

输入域：构成一个被转换数据的字符序列称为一个数据域。即从输入流中当前位置开始，直到其后的第一个空白字符为止，或直至根据转换说明不能被转换的字符之前，或者直至指定域宽范围内的所有字符，都是一个输入域。

输入域的格式：

(1) 输入域之间一般可用空白字符（空格符、换行符、制表符）隔开；在整型、浮点型或字符型后面的字符串数据可以有或无空白字符；一个字符串内部不能有空白字符，因为空白字符是输入域的分隔符。

(2) 当输入完转换说明规定数目的数据并按回车键时，scanf函数开始执行；当用完格式字符串中的转换说明，或当某个输入域与转换说明不匹配时，scanf停止执行；下一次调用scanf函数时从上一次调用已经匹配转换的最后一个字符的下一个字符开始读取。

(3) 当scanf函数的格式字符串中包含有非空白字符的普通字符时，在输入流中的相应位置必须有相同的字符与之一一对应。

**3.4 转换说明中的可选项**

scanf函数转换说明中的选项

选项 意义

m(正整数) 指定输入域的宽度，从自然输入域中取前m个字符作 为实际输入域。若自然输入域不足m个字符，则用完 自然输入域时为止

h 修饰d、i、n、o、u或x，用于输入短整数，其对应的 参数类型为short \*

l 修饰d、i、n、o、u或x，用于输入长整数，其对应的 参数类型为long \*；或修饰转换字符e、f或g，用于输 入双精度浮点数，其对应参数类型为double \*

L 修饰转换字符e、f或g，用于输入ta 双精度浮点数， 其对应参数类型为long double \*

\* 抑制赋值。即跳过输入域，用于虚读

说明：

(1) 如果指定了域宽，则从自然输入域首字符开始取指定宽度的字符作为实际输入域。当自然输入域的宽度小于或等于指定域宽时，则将整个自然输入域作为实际输入域；当自然输入域的宽度大于指定域宽时，则从自然输入域取完指定宽度的字符后，剩下的字符作为下一个自然输入域。

(2) 如果输入短整数、长整数、双精度浮点数、长双精度浮点数，则相应的转换字符前必须分别加h、l或L。

(3) 如果在某个转换字符前面使用了“\*”，则与该转换说明对应的输入域被跳过。这种情况称为“虚读”。虚读用于从输入流中有选择地读取部分内容。