格式化输入与输出

字符输入与输出

字符输入函数getchar

- **字符输入函数getchar**的功能是从标准输入(键盘)一次读取一个字符.其函数原型为:

  ```
  int getchar(void);
  ```

- 函数返回值的类型为int,参数表中void表示函数不需要参数.
- getchar函数在每次调用时,从输入流中读取一个字符,并将所读取**字符的ASCII码(int类型)作为函数返回值**.若遇到文件尾,则返回EOF.
- 符号常量EOF用作文件结束标志,再头文件<stdio.h>中定义,其值一般为-1,不过在程序中应该使用EOF检测文件是否结束,以保证程序同EOF特定值无关.
- 如果需要使用输入的字符数据,应该把输入字符的返回值(ASCII码)赋值给一个字符变量.同时也可以对输入的字符不予处理.
- 输入流中没有字符时,getchar函数进入**等待输入状态**.**回车键**将输入的字符送入输入流,并同时**激活处于等待输入状态**的getchar函数,每个getchar函数**从输入流中读取一个字符(包括'\n')**,直到执行完.

字符输出函数

- **字符输出函数putchar**的功能是从标准输出(显示器)一次输出一个字符.其函数原型为:

  ```
  int putchar(int ch);
  ```

- 函数返回值类型为int,形式参数int ch表明putchar函数需要一个int类型的参数,该参数为所要输出字符的ASCII码值.如果函数正确执行,则返回输出字符的**字符码**,否则返回EOF.
- putchar函数每次调用时,将实际参数ch的字符输出到标准输出设备上.实际参数ch可以是char,short和int类型的表达式,**其值是要输出字符的字符码**.
- 从键盘上输入一段文字后,按下Ctrl+Z和回车键,由于回车键的作用,Ctrl+Z前面的字符一并被送到**缓冲区**,其后字符不被送到缓冲区.如果Ctrl+Z字符前面有其他字符,Ctrl+Z在缓冲区的字符码是26,否则

是-1.Ctrl+Z只有在行首输入才表示结束输入流,否则表示结束本行.

格式化输入与输出

getchar 和 putchar函数只能输入输出**单个字符**,且输入输出过程中**不进行数据格式转换**.
scanf 和 printf函数能够按照用户**指定格式**输入和输出若干数据,称为**格式输入和格式输出**.

格式输入出函数printf

printf函数的功能是按照指定格式向标准输出输出若干数据,其函数原型为:

```
int printf(const char*format,...);
```

- printf在输出**格式format**的控制下,将其**参数进行转换与格式化**,其返回值是int类型,返回值为实际输出的字符个数.
- 第一个形式参数format是一个字符串,称为**格式字符串**,用来指定输出数据的**个数和输出格式**,",..."表示其余参数的数目可变.**其余参数**是要被输出的数据,参数个数和类型应该与格式字符串中转换说明的个数和转换字符一致.
- 格式字符串包含两种字符:**普通字符和转换说明字符**.
  普通字符按原样输出,转换说明字符不直接输出,用来控制其余参数的转换和输出.每个转换说明字符都有一个百分号%开始,以一个转换字符结束.
- 字符%和转换字符之间可以加域宽说明,用来指定输出时的对齐方式,输出数据的域宽,小数部分的位数等格式要求.常用下列字符的组合:**-m.nX**
  - **负号(-)**,用于指定被转换参数按照左对齐形式输出,无负号默认**右对齐**;
  - **正整数(m)**,用于指定输出数据的最小宽度(列数),转换后的参数将输出不小于m的宽度,如果数据的实际宽度小于m,则左边用空格填充以保证最小宽度(如果要求左对齐则右边用空格填充);当格式说明的最小宽度小于实际宽度时,按照实际宽度输出;
  - **小数点(.)**,用于将数据宽度和精度分开;
  - **正整数(n)**,用于指定精度,即指定输出的最大字符数,浮点数小数点后位数,整数最少输出的数字数目;
  - **字母(X)**,可以是h,l或L,h表示整数作位short输出,l表示整数作为long输出,L表示输出参数作为long double类型.
- 常见的转换字符.

转换字符	参数类型	输出格式
c	char	单个字符
s	char *	字符串
d或i	**int**	十进制输出带符号整数

转换字符	参数类型	输出格式
o	**int**	八进制输出无符号整数(无前导0)
x或X	**int**	十六进制输出无符号整数(无前导0x),如果要前导0x可以用%#x
u	**int**	十进制输出无符号整数
f	**double**	小数形式输出浮点数,小数位数默认为6位
e或E	**double**	标准指数形式输出浮点数,尾数部分默认为6位
g或G	**double**	在不输出无效0的情况下,按==输出域较小原则从%f和%e之间选择
p	void*	指针值
%	不转换参数	输出一个%字符

格式输入函数scanf

常见问题可以参考链接

格式输入函数scanf的功能是按指定格式,从标准输入设备中读取字符流,并存放到**输入参数指定的内存单元**.其函数原型为:

```
int scanf(const char*format,...);
```

- scanf函数从标准输入设备中读取字符序列,按照**format的格式**说明对字符序列进行解释,并把结果存放到其余参数的存储单元中,**其余参数必须是指针**,用于**指定经格式转换后对应的输入数据存放的位置**;scanf的返回值是int类型,值是成功输入的数据项个数.若出错则返回值是EOF;
- 在参数表中,format是**格式说明字符串**,用于指定输入数据的数目,类型和格式;",..."表示其它参数数目是可变的,**其它参数必须是用来存放输入数据的内存地址(指针)**.
- 调用函数scanf时,至少读入一个数据,也就是说除了格式说明字符串外,实际参数至少有一个输入参数,输入参数可以是基本类型变量的地址或指针类型变量.
- 用于输入字符串数据的参数应该是**字符类型的指针**,可以是**字符数组名**或者**指向字符数组首元素的指针变量**;输入参数在类型,数目和次序上应该**与格式说明字符串中的转换说明一致**;
- scanf的格式说明字符串与printf类似,用于控制输入的转换.格式说明字符串包括三个部分:
  - **空白字符**:空格符,制表符等,他们被忽略,对数据的输入没有影响;
  - **普通字符(不包括%)**:在输入流中的相应位置必须有相同的字符与之匹配;
  - **转换说明**:形式为:**%[选项]转换字符**,选项可供选择的字符有三种:
    - **赋值禁止字符***:表示跳过它所对应的某个输入域,称为**虚读**;
    - **正整数m**:用于指定输入数据的最大域宽.系统自动截取数据作为实际输入域,或用完自然输入域为止;

- **h,l和L字符**:用于个整数和浮点数转换字符一起输出各种其他类型的整型数或浮点数.
- l修饰e,f,g时,用于输入**double类型浮点数**;而L修饰时,输入**long double类型浮点数**(注意与printf函数中l修饰整型,L修饰双精度浮点型)

转换说明字符	参数类型	输入数据
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, 对应参数可以是字符或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**.遇到空格、 制表符或换行符等空白字符时停止读取, 存放在参数指定的内存地址时, 尾部加上'\0'
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(有前导0x)
o	**int ***	八进制整数(有无前导0均可)
x	**int ***	十六进制整数(有无前导0x均可)
u	**unsigned int ***	十进制无符号整数
e,f,g	**float *** (在printf函数中,e,f,g都对应double类型)	浮点数,无位宽限制时, 默认输入六位小数
%	无参数	输入%,无赋值操作

- scanf函数的格式说明字符串中一般只包含转换说明,对于除空白字符之外的其他普通字符,在输入流中相应位置必须输入相同的字符与之匹配,而空白字符(按Tab键等)会使scanf函数在读操作中**略去输入中的零个或多个空白字符**,使转换说明更加清晰;
- 在C语言中,可以**使用 %Ns 这样的格式说明符限制读取的字符数**,N是一个整数,表示最多读取的字符数.但是,这种格式说明符并不会在读取的字符串末尾自动添加字符串结束符 '\0',因此你需要在读取后手动添加.
    - %1s 是 scanf 函数中的格式说明符,表示读取一个字符串,但是**最多只读取一个字符**,并且不会自动在末尾添加字符串结束符'\0'.使用 %1s可能会导致缓冲区溢出,因为 scanf 不会在读取的字符后自动添加字符串结束符'\0'.所以在读取后,必须确保为字符串添加结束符以避免未定义行为.

    - 在 printf 函数中,%1s不是一个有效的格式说明符.在 printf 函数中,格式说明符是用来指定要输出的数据类型和格式的.在 printf 中,要打印字符串,可以使用 %s 格式说明符,它将打印整个字符串直到遇到字符串结束符为止.
- E.G.

  `scanf("%3d%*2d%3d%*d",&a,&b);`

  `scanf("%c %2c",&x,&y);`

  `scanf("%c%2c",&x,&y);`

  `scanf("%c,%2c",&x,&y);`

  `scanf("%d%s%d",&a,str,&b);` 注意:输入流中,在系统不能区分输入数据的情况下,不同数据间最好用空白字符隔开.