字符常量

转义字符相当于一个字符常量==‘c’

转义字符	作用
\n	换行
\f	清屏并换页
\r	回车
\t	Tab符，水平制表
\v	垂直制表
\b	退格
\0	空格，作为字符串结束标志
\a	提示报警提示音
\″	双引号
\’	单引号
\\	一个反斜线
\?	问号
\xhh	表示一个ASCII码用16进表示，其中hh是1到2个16进制数
\ddd	表示一个ASCII码用8进表示，其中ddd是三个8进制数

每次按下tab制表符，并不是从当前的光标位置向后移动一个tab制表符，而是移动到下一个制表位。

abc\n　　代表了4个字符而不是5个字符

字符的输入与输出一非格式化输入输出函数

字符的输入与输出又称为非格式化输入输出函数可以由上面讲述的标准格式化输入输出函数代替，但这些函数编译后代码少，相对占用内存也小，从而提高了速度，。

一、puts()和gets()函数

1. puts()函数

puts()函数用来向标准输出设备(屏幕)写字符串并换行，其调用格式为:

puts(s);

其中s为字符串变量(字符串数组名或字符串指针)。

puts()函数的作用与语printf("%s\n", s)相同。

例4:

```
main()
{
        char s[20], *f;  /*定义字符串数组和指针变量*/
        strcpy(s, "Hello! Turbo C2.0"); /*字符串数组变量赋值*/
        f="Thank you";/*字符串指针变量赋值*/
        puts(s);
        puts(f);
}
```

说明:

(1). puts()函数只能输出字符串, 不能输出数值或进行格式变换。

(2). 可以将字符串直接写入puts()函数中。如:

```
puts("Hello, Turbo C2.0");
```

2. gets()函数

gets()函数用来从标准输入设备(键盘)读取字符串直到回车结束读入回车符但不处理，scanf是不读入回车符号, 但回车符不属于这个字符串。其调用格式为:

```
gets(s);
```

其中s为字符串变量(字符串数组名或字符串指针)。

gets(s)函数与scanf("%s", &s)相似, 但不完全相同, 使用scanf("%s", &s)函数输入字符串时存在一个问题, 就是如果输入了空格会认为输入字符串结束, 空格后的字符将作为下一个输入项处理, 但gets() 函数将接收输入的整个字符串直到回车为止，包括空格。

例5

```
main()
{
        char s[20], *f;
        printf("What's your name?\n");
        gets(s);/*等待输入字符串直到回车结束*/
        puts(s);/*将输入的字符串输出*/
        puts("How old are you?");
        gets(f);
```

```
        puts(f);
    }
```

说明:

(1). gets(s)函数中的变量s为一字符串。如果为单个字符,编译连接不会有错误, 但运行后会出现"Null pointer asignmemt"的错误。

二、putchar()、getch()、getche()和getchar()函数

1. putchar()函数

putchar()函数是向标准输出设备输出一个字符, 其调用格式为:

putchar(ch);

其中ch为一个字符变量或常量。

putchar()函数的作用等同于printf("%c", ch);

例6:

```
#include<stdio.h>
main()
{
        char c:                         /*定义字符变量*/
        c='B';                          /*给字符变量赋值*/
        putchar(c);                     /*输出该字符*/
        putchar('\x42');        /*输出字母B*/
        putchar(0x42);              /*直接用ASCII码值输出字母B*/
}
```

从本例中的连续四个字符输出函数语句可以分清字符变量的不同赋值方法。

2. getch()、getche()和getchar()函数

(1) getch()和getche()函数

这两个函数都是从键盘上读入一个字符。其调用格式为:

```
getch();
getche();
```

两者的区别是: getch()函数不将读入的字符回显在显示屏幕上, 而getche()
函数却将读入的字符回显到显示屏幕上。

例7:

```
#include<stdio.h>
main()
{
        char c, ch;
        c=getch();          /*从键盘上读入一个字符不回显送给字符变量c*/
        putchar(c);         /*输出该字符*/
        ch=getche();       /*从键盘上带回显的读入一个字符送给字符变量ch*/
        putchar(ch);
}
```

利用回显和不回显的特点，这两个函数经常用于交互输入的过程中完成暂停等功能。

例8:

```
#include<stdio.h>
main()
{
        char c, s[20];
        printf("Name:");
        gets(s);
        printf("Press any key to confinue...");
        getch();    /*等待输入任一键*/
}
```

(2) getchar()函数

getchar()函数也是从键盘上读入一个字符，并带回显。它与前面两个函数的区别在于：getchar()函数等待输入直到按回车才结束,回车前的所有输入字符都会逐个显示在屏幕上。但只有第一个字符作为函数的返回值。

并且getchar没有参数，函数的返回值就是从终端键盘读入的字符。

getchar()函数的调用格式为:

```
getchar();
```

例9:

```
#include<stdio.h>
main()
{
        char c;
        c=getchar();      /*从键盘读入字符直到回车结束*/
        putchar(c);       /*显示输入的第一个字符*/
        getch();          /*等待按任一健*/
}
```

标准输入输出函数

1.1.1 格式化输入输出函数

标准库提供了两个控制台格式化输入、 输出函数printf( ) 和scanf(), 这两个函数可以在标准输入输出设备上以各种不同的格式读写数据。printf()函数用来向标准输出设备(屏幕)写数据; scanf() 函数用来从标准输入设备(键盘)上读数据。下面详细介绍这两个函数的用法。

一、printf()函数

printf()函数是格式化输出函数, 一般用于向标准输出设备按规定格式输出信息。在编写程序时经常会用到此函数。printf()函数的调用格式为:

printf("<格式化字符串>", <参量表>);

其中格式化字符串包括两部分内容: 一部分是普通字符,这些字符将按原样输出; 另一部分是格式化规定字符或称为格式转换说明, 以"%"开始, 后跟一个或几个规定字符,用来确定输出内容格式。

参量表是需要输出的一系列参数, 其个数必须与格式化字符串所说明的输出参数个数一样多, 各参数之间用","分开, 且顺序一一对应, 否则将会出现意想不到的错误。

1. 格式化规定符

标准库提供的格式化规定符如下:

符号	作用
%d	十进制有符号整数

%u　　十进制无符号整数

%x，%X　　无符号以十六进制（小写，大写）表示的整数

%o　　无符号以八进制表示的整数

%o　　输出一个百分号%

%f　　浮点数

%s　　字符串

%c　　单个字符

%p　　指针的值

%e，%E　　指数形式的浮点数（小写，大写）

%g　　自动选择f或e中宽度比较小的一种，且不输出无意义的0

说明:所有的修饰符都是在%之后的，包括*

(1)可以输出域宽m，显示精度.n。在"%"和字母之间插进数字表示最大场宽。

M用于指定输出项输出时所占据的列数，也就是最大的输出长度。

N用来控制精度，浮点数的小数位数的长度。

对于字符串，用于指定从左侧截取的子串的个数

例如:　%3d　表示输出3位整型数，不够3位右对齐。

%9.2f 表示输出场宽为9的浮点数，其中小数位为2，整数位为6,

小数点占一位，不够9位右对齐。

%8s　表示输出8个字符的字符串，不够8个字符右对齐。

如果字符串的长度、或整型数位数超过说明的场宽，将按其实际长度输出。但对浮点数，若整数部分位数超过了说明的整数位宽度，将按实际整数位输出;若小数部分位数超过了说明的小数位宽度，则按说明的宽度以四舍五入输出。

另外，若想在输出值前加一些0，就应在场宽项前加个0。，不写仅仅补空格

小数点也占据一个字符的位置

例如:

如果用浮点数表示字符或整型量的输出格式，小数点后的数字代表最大宽度，
小数点前的数字代表最小宽度。

例如：%6.9s 表示显示一个长度不小于6且不大于9的字符串。若大于9，则
第9个字符以后的内容将被删除。

(2). 可以在"%"和字母之间加小写字母l，表示输出的是长型数。大写L是用来输出long double型的数据。h用于输出short型的数据。

doxu-------l

feg--------L

dox--------h

例如：　　%ld　　表示输出long整数
　　%lf　　表示输出double浮点数

(3). M可以控制输出左对齐或右对齐，即在"%"和字母之间加入一个"-"号可
说明输出为左对齐，否则为右对齐。多余的位置补空格。如果M的前面有前导符0，则在空余的位置补上0.

例如：　　%7d　表示输出7位整数右对齐
　　%-10s　表示输出10个字符左对齐
　　%04d　输出一个小于4位的数值时，将在前面补0使其总宽度为4位。

2. 一些特殊规定字符

由本节所学的printf()函数，并结合上一节学习的数据类型，编制下面的程序，以加深对Turbo C2.0数据类型的了解。

例1

```
#include<stdio.h>
#include<string.h>
int main()
{
```

```
                    char c, s[20], *p;
                    int a=1234, *i;
                    float f=3.141592653589;
                    double x=0.12345678987654321;
                    p="How do you do";
                    strcpy(s, "Hello, Comrade");
                    *i=12;
                    c='\x41';
                    printf("a=%d\n", a);          /*结果输出十进制整数
a=1234*/
                    printf("a=%6d\n", a);        /*结果输出6位十进制数a=
1234*/
                    printf("a=%06d\n", a);      /*结果输出6位十进制数
a=001234*/
                    printf("a=%2d\n", a);        /*a超过2位, 按实际值输出
a=1234*/
                    printf("*i=%4d\n", *i);    /*输出4位十进制整数*i=
12*/
                    printf("*i=%-4d\n", *i); /*输出左对齐4位十进制整数
*i=12*/
                    printf("i=%p\n", i);          /*输出地址i=06E4*/
                    printf("f=%f\n", f);          /*输出浮点数f=3.141593*/
                    printf("f=6.4f\n", f);      /*输出6位其中小数点后4位的
浮点数

 f=3.1416*/
                    printf("x=%lf\n", x);        /*输出长浮点数x=0.123457*/
                    printf("x=%18.16lf\n", x);/*输出18位其中小数点后16位的
长浮点

 数x=0.1234567898765432*/
                    printf("c=%c\n", c);          /*输出字符c=A*/
```

```
                printf("c=%x\n", c);          /*输出字符的ASCII码值
c=41*/
                printf("s[]=%s\n", s);      /*输出数组字符串s[]=Hello,
Comrade*/
                printf("s[]=%6.9s\n", s);/*输出最多9个字符的字符串
s[]=Hello,

Co*/
                printf("s=%p\n", s);          /*输出数组字符串首字符地
址s=FFBE*/
                printf("*p=%s\n", p);        /* 输出指针字符串p=How do
you do*/
                printf("p=%p\n", p);          /*输出指针的值p=0194*/
                getch();
                retunr 0;
        }
```

上面结果中的地址值在不同计算机上可能不同。

二、scanf()函数

scanf()函数是格式化输入函数，它从标准输入设备(键盘) 读取输入的信息。

其调用格式为:

scanf("<格式化字符串>", <地址表>);

格式化字符串包括以下三类不同的字符;

1. 格式化说明符: 格式化说明符与printf()函数中的格式说明符基本相同。

2. 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。

3. 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。

地址表是需要读入的所有变量的地址，而不是变量本身。这与printf()函数完全不同，要特别注意。各个变量的地址之间同","分开。

(2). 可以在"%"和字母之间加小写字母l，表示输入的是长型数。大写L是用来输出long double型的数据。h用于输入short型的数据。

doxu-------l

fe--------L-double

dox--------h

*---------忽略输入修饰符

(3). 可以控制输出左对齐或右对齐，即在"%"和字母之间加入一个"-"号可说明输出为左对齐，否则为右对齐。

输入结束的标志：

- 遇到空格符，回车符，制表符
- 达到输入域宽
- 遇到非法字符输入
- 如果scanf的格式控制字符串中存在除了格式说明符以外的其他字符，那么这些字符必须在输入数据时由用户从键盘原样输入。

scanf("是的撒或大所多所多%d, %d", &i, &j);//请永远不要这样写

例2:

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
        int i, j;
        printf("i, j=?\n");
        scanf("%d, %*d", &i, &j);*---------忽略输入修饰符，对j的输入处理是无效的
        printf("%d, %d", i,j);
        system("pause");
        return 0;
}
```

上例中的scanf()函数先读一个整型数，然后把接着输入的逗号剔除掉，最

后读入另一个整型数。如果","这一特定字符没有找到，scanf()函数就终止。若

参数之间的分隔符为空格，则参数之间必须输入一个或多个空格。

也就是说，如果scanf的格式控制字符串中存在除了格式说明符以外的其他字符，那么这些字符必须在输入数据时由用户从键盘原样输入。

说明:

(1). 对于字符串数组或字符串指针变量，　由于数组名和指针变量名本身就
是地址，因此使用scanf()函数时，不需要在它们前面加上"&"操作符。

例3

```
  mian()
  {
        char *p, str[20];
        scanf("%s", p);                    /*从健盘输入字符串*/
        scanf("%s", str);
        printf("%s\n", p);                 /*向屏幕输出字符串*/
        printf("%s\n", str);
  }
```

(2). 可以在格式化字符串中的"%"各格式化规定符之间加入一个整数，表示
任何读操作中的最大位数。并且在输入函数下，没有精度的控制函数。

如例3中若规定只能输入10字符给字符串指针p，则第一条scanf() 函数语句变为

```
scanf("%10s", p);
```

程序运行时一旦输入字符个数大于10，　p就不再继续读入，而后面的一个读
入函数即scanf("%s", str)就会从第11个字符开始读入。

实际使用scanf()函数时存在一个问题，下面举例进行说明:

当使用多个scanf()函数连续给多个字符变量输入时，例如:

```
  main()
  {
        char c1, c2;
        scanf("%c", &c1);
```

```
        scanf("%c", &c2);
        printf("c1 is %c, c2 is %c", c2\1, c2);
    }
```

运行该程序，输入一个字符A后回车（要完成输入必须回车），在执行scanf("%c", &c1)时，给变量c1赋值"A"，但回车符仍然留在缓冲区内，执行输入语句scanf("%c", &c2)时，变量c2输出的是一空行，如果输入AB后回车，那么输出结果为：c1 is A, c2 is B。

要解决以上问题，可以在输入函数前加入**清除函数fflush()使用多个输入函数必备**（这个函数的使用方法将在本节最后讲述）。修改以上程序变成：

```
#include<stdio.h>
main()
{
        char c1, c2;
        scanf("%c", &c1);
        fflush(stdin);//清除终端输入
        scanf("%c", &c2);
        printf("c1 is %c, c2 is %c", c1, c2);
}
```

有一些问题需要了解scanf：

比如，其会将回车和空格当做有效的字符进行读入 。

- scanf函数的scanf和gets对于空格的不同
- 使用getchar读取空格的方式
- 在%c前加一个空格，感觉比较鸡肋

如果scanf的格式控制字符串中存在除了格式说明符以外的其他字符，那么这些字符必须在输入数据时由用户从键盘原样输入。

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int i, j;
    printf("i, j=?\n");
```

```
        scanf("%d,%d", &i, &j);//12，34
        scanf("%d %d", &i, &j);//12 34
        scanf("a = %d,b = %d", &i, &j);//a = 12,b = 34
        scanf("%2d%2d", &i, &j);//1234

        scanf("%d*C%d", &i, &j);//12“任意一个字节char型，该字节会被忽略”34
        scanf("%2d%*2d%d", &i, &j);//12“任意两个int型数字，会被忽略”34
    int d =  scanf("%2d%2d", &i, &j);//scanf会有一个返回值，返回值是输入的数据项的项数
```

当其的项数为-1也就是EOF，属于scanf的宏定义的内容时，即为读取数据时发生了错误，需要进行判断和处理。

```
        printf("a = %d,b = %d", i,j);
        system("pause");
        return 0;
}
```