**基本的标准输出**

在调用标准输入、输出函数之前，必须用编译预处理命令#include “stdio.h”包含系统关于标准输入、输出的头文件stdio.h。在stdio.h头文件中对标准输入、输出函数的原型进行了说明。

根据处理对象与用途的不同，C语言提供了三种基本的标准输出函数：putchar、puts以及printf。其中，putchar用来输出单个字符，puts用来输出字符串，printf用于格式输出。

**1. 单字符输出函数putchar**

**1.1 函数原型**

函数原型是指对函数的名称、返回值类型、参数的数目和参数类型的说明。函数原型规定了调用该函数的语法格式，即函数调用形式。putchar函数的原型为：

int putchar(int c);

原型的开关的int表明putchar函数返回值的类型，形式参数表int c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。putchar函数将所要输出字符的字符码转换为unsigned char类型并送到标准输出设备中去。如果函数正确执行，则返回所输出字符的字符码，否则返回EOF。

EOF是在头文件stdio.h中定义的一个符号常量，其值为-1，用作文件结束标志(End Of File)。在DOS系统中，当从终端键盘键入组合命令“Ctrl + z”，以及在UNIX系统中键入组合键命令“Ctrl + d”，可以输入一个文件结束标志EOF。

**1.2 函数的调用形式**

putchar函数的调用形式为：

putchar(c)

其中，c为实际参数，它可为char、short与int类型的表达式，其值是要输出字符的字符码。putchar可将字符c输出到标准输出设备上。示例如下：

// 使用putchar函数打印HUST

char c1 = 'H', c2 = 'U', c3 = 'S', c4 = 'T';

putchar(c1); putchar(c2); putchar(c3); putchar(c4); 、

// 输出字符'a'的图形符号

char c = 'a';

putchar('a'); putchar(c); putchar(97); putchar('\141');

// 系统扬声器响铃一次

putchar('\a');

**2. 字符串输出函数puts**

**2.1 函数原型**

puts函数的原型为：

int puts(const char \*s);

puts函数返回值类型为int，参数s是存放所要输出字符串的内存缓冲区的首地址，类型为字符指针。puts函数从s所指定的地址读取字符串输出到标准输出设备并在串尾输出一个换行符‘\n’。字符串在内存缓冲区存储时串尾以空字符‘\0’作为结束标志。puts取字符串时从s指定的内存区依次取字符直至取到空字符。puts函数正确执行时返回一个非负整数值，如果出错，则返回EOF。参数表const char \*s中的const表明字符指针s的值不会被该函数改变。

**2.2 函数的调用形式**

puts函数的调用形式为：

puts(s)

其中，s为实际参数，可以是字符串常量，字符数组名，或是指向某字符串的字符指针变量。在C语言中，字符串常量和字符数组名所表示的都是字符指针类型的常量值。字符串常量所表示的常量值是该字符串在内存缓冲区的首地址。字符数组名所表示的常量值是该字符数组在内存中的首地址（即数组的第一个字符元素的地址）。

**3. 格式输出函数printf**

**3.1 函数原型**

printf函数的原型为：

int printf(const char \* format, …);

printf函数的返回值类型为int。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；后面的“, …”表示该函数在调用时除前面的字符串参数必需之外，其余参数的数目可变，即可以是0个到多个。其余参数是要被输出的数据，参数的个数和数据类型应与格式字符串中转换说明的个数和转换字符一致。printf函数的返回值是函数调用时实际输出到标准输出设备的字符个数。

**3.2 函数调用形式**

printf函数的调用形式为：

printf(格式字符串，数据项1，数据项2，…，数据项n)

其中“(格式字符串，数据项1，数据项2，…，数据项n)”是实际参数表，第一个参数“格式字符串”是必需的，其余参数“数据项1”至“数据项n”是要输出的数据，每个数据项是一个基本类型或指针类型的表达式。数据项的个数可以是0个到任意多个，但是数据项在数目、数据类型和顺序上应与格式字符串中的格式转换说明一致；如果不一致，编译时不会报错，但是不能得到正确的结果。

**3.3 printf函数的格式字符串**

**3.3.1 格式字符串的组成**

总体来看，格式字符串包含两类字符：普通字符和用于转换说明的字符。普通字符是作为提示信息来输出的文字，可以与转换说明交替出现，并且普通字符按原样输出。转换说明以%字符开始，以转换字符结尾。%字符和转换字符之间还可以带有域宽说明，用来指出输出的对齐方向，输出数据域的宽度、小数部分的位数等格式要求。转换说明的语法形式为：

%[域宽说明]转换字符

标准C定义的转换字符及相应的输出格式如下表所示：

printf的转换字符

转换字符 参数类型 输出格式

d, i int 十进制整数

o int 八进制整数(不带前缀0)

x, X int 十六进制整数(不带前缀0x或0X)

u int 无符号十进制数

c char 单个字符

s char \* 字符串(必须以‘/0’结束或在域宽说明中给出 长度限制)

f double 小数形式的浮点数(小数部分位数由精度确 定，缺省为6位)

e, E double 标准指数形式的浮点数

g, G double 不输出无效0，按输出宽度较少的原则从%f 和%e两种格式中自动选择

p void \* 指针值(输出格式与具体实现有关)

n int \* 将到达该转换说明为止的已输出的字符数目 写入对应的参数中，不转换参数

% 不转换 输出一个%字符

值得注意的是，如果%后面的转换字符是%，则不进行参数转换而是输出一个%字符；如果%后面是%以外的非转换字符，则多数编译系统将其作为普通字符输出，而标准C对此没有明确规定。

下面是关于常用格式字符串用法的几点说明：

(1) 转换说明%d和%u的区别是，对于二进制最高位为0的整数，两者输出相同的结果；对于二进制最高位为1的整数，%d输出相应负数值，%u输出相应的正数值。

int i1 = 100, i2 = -1;

unsigned j1 = 100, j2 = -1;

printf("%d, %u\n", i1, i1); // 100, 100

printf("%d, %u\n", j1, j1); // 100, 100

printf("%d, %u\n", i2, i2); // -1, 65535

printf("%d, %u\n", j2, j2); // -1, 65535

(2) %f、%e、%g三种转换说明在缺省域宽说明时的输出精度为：%f输出小数形式的浮点数，小数部分为6位，多于6位采用四舍五入，少于6位则在末位补零，以保证6位小数；%e输出标准指数形式的浮点数，尾数部分为6个有效数字，包括1位非零的整数部分和5位小数部分，同样采用四舍五入和末位补零的方法确保6个有效数字；%g将按%f和%e两种转换说明输出的数据去掉无效零后进行比较，选取输出宽度较小的那种格式进行输出。

double x = 12.00004951, y = 12.4951, z = 12000000.4951;

printf("%f, %e, %g\n", x, x, x);

printf("%f, %e, %g\n", y, y, y);

printf("%f, %e, %g\n", z, z, z);

(3) %的用法举例。

printf("%%d\n", 100); // %d

printf("%%%%\n"); // %%

printf("%%%d\n", 100); // %100

printf("%m\n", 100); // %m

(4) 转换说明与输出参数不一致时可能导致的后果：因为printf函数根据格式字符串中的转换说明来决定输出的数据项的数目和类型，如果转换字符使用不当，或转换说明的个数多于数据项的个数，则会输出错误的数据；如果转换说明的个数少于数据项的个数，则多出的数据项不被输出。

**3.3.2 域宽说明**

在%和转换字符之间可能有域宽说明字符，用来指出输出数据的对齐方式、输出数据域的宽度、小数部分的位数等要求。

printf域宽说明字符

域宽说明字符 意义

– (减号) 在输出域宽内以左对齐方式输出，缺省为右对齐

+ 输出正数时前面要加符号+

空格 输出的第一个字符不是符号 + 或 – 时要输出一个 空格作为前缀

0(零) 在输出域宽范围内用前导0补齐空位

# 对于o格式和x格式输出前缀0或0x；对于g格式 的输出不删除尾部零

m(正整数) 指出输出数据的最小域宽。如果数据的实际宽度大 于m，则按实际宽度输出；如果实际宽度小于m， 则左边(左对齐为右边)补空格或零(有0域宽说明 时)

.(小数点) 分隔域宽与精度。小数点前可以没有域宽

n(正整数) 指出输出数据的精度。对于e、E、f格式的为小数 部分的位数，对于g、G格式为有效数据的个数；对 于整数为至少应输出的数字个数(用前导0补足)； 对于字符串为至多输出的字符数目

h 指出输出参数是短整型

l 指出输出参数是长整型

L 指出输出参数是长双精度浮点型

\* 代表一个整数，其值由对应的参数决定，可用于代 替m或n表示可变域宽或精度

【示例】\*作为可变域宽的用法。

int max = 6;

char s[10] = "123456789";

printf("%\*c\n", max, 'H'); // 输出5个空格和H

printf("%\*s\n", 2\*max, s); // 输出123456789，左边有3个空格

printf("%.\*s\n", max, s); // 输出123456