cout和printf的区别

今天无意中发现cout和printf是有点区别的，一个是输出语句，一个是输出函数

我编了个函数如下：

#include "iostream.h"

#include "stdio.h"

main()

{

int a;

for(a=0;a<24;a++)

{

printf("++++++++++++\n");

cout<<"============\n";

printf("############\n");

}

printf("@@@@@@@@@@@\n");

}

运行结果如下：

++++++++++++

++++++++++++

############

############

@@@@@@@@@@@

============

============

cout先是把输出结果存到缓存区，然后一次性输出，其实COUT输出的时候也是调用了printf函数；

pintf函数是每次输出结果;

这就是为了么在TIME程序中cout运行的时间要比printf函数快的原因；

当时有一点需要注意：cout<<"============\n";和cout<<"============"<<endl;也有不同，\n只是一个字符，而endl会将缓存区的数据全部输出并清零；在TIME程序中，如果采用cout<<"============"<<endl;结构反而会比Printf慢，因为多了一道调用缓存手续

下面打个很形象的比喻，大家都会明白

比如有100个鸡蛋，需要从A拿到B点，相当于程序中的输出

Printf把鸡蛋一次一个用手拿到B点

cout<<" \n"先把鸡蛋全部放到篮子，然后一次性拿过去取出

cout<<" "<<endl先把鸡蛋放到篮子，然后一次一个拿过去再取出

尤其可见哪个快那个慢显而易见

# 一、Printf输出格式

C中格式字符串的一般形式为：%[标志][输出最小宽度][.精度][长度]类型，其中方括号[]中的项为可选项。各项的意义介绍如下：  
1.类型类型字符用以表示输出数据的类型，其格式符和意义下表所示：  
表示输出类型的格式字符　　　　　格式字符意义  
a                                                                 浮点数、十六进制数字和p-计数法(C99)  
A                                                                 浮点数、十六进制数字和p-计数法(C99)  
c　　　　　　　　　　　　　　　　输出单个字符  
d　　　　　　　　　　　　　　　　以十进制形式输出带符号整数(正数不输出符号)  
e　　　　　　　　　　　　　　　　以指数形式输出单、双精度实数  
E　　　　　　　　　　　　　　　　以指数形式输出单、双精度实数  
f　　　　　　　　　　　　　　　　以小数形式输出单、双精度实数  
g　　　　　　　　　　　　　　　　以%f%e中较短的输出宽度输出单、双精度实数,%e格式在指数小于-4或者大  于等于精度时使用  
G　　　　　　　　　　　　　　　　以%f%e中较短的输出宽度输出单、双精度实数,%e格式在指数小于-4或者大于等于精度时使用  
i                                                                  有符号十进制整数(与%d相同)  
o　　　　　　　　　　　　　　　　以八进制形式输出无符号整数(不输出前缀O)  
p                                                                 指针  
s　　　　　　　　　　　　　　　　输出字符串  
x　　　　　　　　　　　　　　　　以十六进制形式输出无符号整数(不输出前缀OX)  
X　　　　　　　　　　　　　　　　以十六进制形式输出无符号整数(不输出前缀OX)  
u　　　　　　　　　　　　　　　　以十进制形式输出无符号整数  
/\*  测试常见的输出类型  \*/  
#include"stdio.h"  
#include "conio.h"  
main()  
{  
    printf("The program test print style!\n");  
    /\*  以十进制形式输出带符号整数(正数不输出符号)  \*/  
    printf("%d\n" , 223);  
    printf("%d\n" , -232);  
    printf("\n");  
     /\*  以八进制形式输出无符号整数(不输出前缀O)  \*/  
    printf("%o\n" , 223);  
    printf("%o\n" , -232);  
    printf("\n");  
    /\*   以十六进制形式输出无符号整数(不输出前缀OX)  \*/  
    printf("%x\n" , 223);  
    printf("%x\n" , -232);  
    printf("\n");  
     /\*  以十进制形式输出无符号整数  \*/  
    printf("%u\n" , 223);  
    printf("%u\n" , -232);  
    printf("\n");  
    /\*   以小数形式输出单、双精度实数  \*/  
    printf("%f\n" , 223.11);  
    printf("%f\n" , 232.11111111);  
    printf("%f\n" , -223.11);  
    printf("%f\n" , -232.11111111);  
    printf("\n");  
    /\*   以指数形式输出单、双精度实数  \*/  
    printf("%e\n" , 223.11);  
    printf("%e\n" , 232.11111111);  
    printf("%e\n" , -223.11);  
    printf("%e\n" , -232.11111111);  
    printf("\n");  
    /\*   以%f%e中较短的输出宽度输出单、双精度实数  \*/  
    printf("%g\n" , 223.11);  
    printf("%g\n" , 232.111111111111);  
    printf("%g\n" , -223.11);  
    printf("%g\n" , -232.111111111111);  
    printf("\n");  
    /\*   输出单个字符  \*/  
    printf("%c\n" , 'a');  
    printf("%c\n" , 97);  
    printf("\n");  
     /\*  输出单个字符  \*/  
    printf("%s\n" , "this is a test!");  
    printf("%s\n" , "2342o34uo23u");  
    printf("\n");  
    getch();  
}  
2.标志  
标志字符为-、+、#、空格和0五种，其意义下表所示：  
标格 输出值为正时冠以空格，为负时冠以负号  
#　　　　　　　　　对c，s，d，u类无影响；对o类，在输出时加前缀0；对x类，  
                                        在输出时加前缀0x或者0X；对g，G类防止尾随0被删除；  
                                        对于所有的浮点形式，#保证了即使不跟任何数字，也打印一个小数点字符  
0                                      对于所有的数字格式，用前导0填充字段宽度，若出现-标志或者指定了精度(对于整数),忽略  
3.输出最小宽度  
用十进制整数来表示输出的最少位数。若实际位数多于定义的宽度，则按实际位数输出，若实际位数少于定义的宽度则补以空格或0。  
/\*  测试标志字符为-、+、#、空格四种  \*/  
#include"stdio.h"  
#include "conio.h"  
main()  
{  
      /\*   以十进制形式输出带符号整数(正数不输出符号)  \*/  
     printf("\*%-10d\*\n",223);  
     printf("\*%+10d\*\n" ,-232);  
     printf("\*%2d\*\n" ,223);  
     printf("\*%#d\*\n" ,-232);  
     printf("\n");  
    getch();  
    /\*   以八进制形式输出无符号整数(不输出前缀O)  \*/  
     printf("\*%-10o\*\n" ,223);  
     printf("\*%+10o\*\n" ,-232);  
     printf("\*%o\*\n" ,223);  
     printf("\*%#o\*\n" ,-232);  
     printf("\n");  
    getch();  
     /\*  以十六进制形式输出无符号整数(不输出前缀OX)  \*/  
     printf("$%-10x$\n" ,223);  
     printf("$%010x$\n" ,-232);  
     printf("$% x$\n" ,223);  
     printf("$%#x$\n" ,-232);  
     printf("\n");  
    /\*   以十进制形式输出无符号整数  \*/  
    printf("%-10u\n" , 223);  
    printf("%+10u\n" , -232);  
    printf("% u\n" , 223);  
    printf("%#u\n" , -232);  
    printf("\n");  
     getch();  
    /\*   以小数形式输出单、双精度实数  \*/  
    printf("%-10f\n" , 223.11);  
    printf("%+10f\n" , 232.11111111);  
    printf("% f\n" , -223.11);  
    printf("%#f\n" , -232.11111111);  
    printf("\n");  
     getch();  
    /\*   以指数形式输出单、双精度实数  \*/  
    printf("%-10e\n" , 223.11);  
    printf("%+10e\n" , 232.11111111);  
    printf("% e\n" , -223.11);  
    printf("%#e\n" , -232.11111111);  
    printf("\n");  
     getch();  
    /\*   以%f%e中较短的输出宽度输出单、双精度实数  \*/  
    printf("%-10g\n" , 223.11);  
    printf("%+10g\n" , 232.111111111111);  
    printf("% g\n" , -223.11);  
    printf("%#g\n" , -232.111111111111);  
    printf("\n");  
     getch();  
    /\*   输出单个字符  \*/  
    printf("%-10c\n" , 'a');  
    printf("%+10c\n" , 97);  
    printf("% c\n" , 'a');  
    printf("%#c\n" , 97);  
    printf("\n");  
     getch();  
    /\*   输出单个字符  \*/  
    printf("%-20s\n" , "this is a test!");  
    printf("%+20s\n" , "2342o34uo23u");  
    printf("% 20s\n" , "this is a test!");/\* 不足补空格\*/  
    printf("%#s\n" , "2342o34uo23u");  
    printf("\n");  
    getch();  
}  
4.精度  
精度格式符以“.”开头，后跟十进制整数。本项的意义是：如果输出数字，则表示小数的位数；如果输出的是字符，则表示输出字符的个数；若实际位数大于所定义的精度数，则截去超过的部分。  
/\*  测试精度  \*/  
#include"stdio.h"  
#include "conio.h"  
main()  
{  
    printf("%.3d\n" , 5555);  
    getch();  
     printf("%.3f\n" ,0.88888);  
     getch();  
    printf("%.3f\n" , 0.9999);  
    getch();  
     printf("%.4s\n" ,"this is a test!");  
    getch();  
}  
5.长度  
长度格式符为h,l两种，h表示按短整型量输出，l表示按长整型量输出。  
    h和整数转换说明符一起使用，表示一个short  int 或者unsignedshort int类型的数值，示例：  
%hu，%hx，%6.4hd  
    hh和整数转换说明符一起使用，表示一个short  int 或者unsignedshort类型的数值，示例：  
%hhu，%hhx，%6.4hhd  
    j和整数转换说明符一起使用，表示一个intmax\_t或者uintmax\_t类型的数值，示例：  
%jd,%8jx  
    l和整数转换说明符一起使用，表示一个longint 或者unsignedlong int类型的数值，示例：  
%ld,%8lu  
    ll和整数转换说明符一起使用，表示一个longint 或者unsignedlong int类型的数值(C99)，示例：  
%lld,%8llu  
    L和浮点转换说明符一起使用，表示一个longdouble的值，示例：%Lf，%10.4Le  
    t和整数转换说明符一起使用，表示一个ptrdiff\_t值(两个指针之间的差相对应的类型)(C99)，示例：  
%td,%12ti  
    z和整数转换说明符一起使用，表示一个size\_t值(sizeof返回的类型)(C99)，示例：%zd,%12zx  
main(){  
inta=15;  
float b=138.3576278;  
double c=35648256.3645687;  
chard='p';  
printf("a=%d,%5d,%o,%x\n",a,a,a,a);  
printf("b=%f,%lf,%5.4lf,%e\n",b,b,b,b);  
printf("c=%lf,%f,%8.4lf\n",c,c,c);  
printf("d=%c,%8c\n",d,d);  
}  
a<--15  
b<--138.3576278  
c<--35648256.3645687  
d<--'p'  
main()  
{  
    int a=29;  
     float b=1243.2341;  
    double c=24212345.24232;  
     char d='h';  
    printf("a=%d,%5d,%o,%x\n",a,a,a,a);  
     
    /\* 其中“%f”和“%lf”格式的输出相同，说明“l”符对“f”类型无影响  
     \* “%5.4lf”指定输出宽度为5，精度为4，由于实际长度超过5故应该按实际位数输出，小数位数超过4位       \*  部分被截去  
     \*/  
    printf("b=%f,%lf,%5.4lf,%e\n",b,b,b,b);  
     
    /\* 输出双精度实数，“%8.4lf”由于指定精度为4位故截去了超过4位的部分\*/  
    printf("c=%lf,%f,%8.4lf\n",c,c,c);  
     
    /\* 输出字符量d，其中“%8c”指定输出宽度为8故在输出字符p之前补加7个空格\*/  
    printf("d=%c,%8c\n",d,d);  
    getch();  
}  
使用printf函数时还要注意一个问题，那就是输出表列中的求值顺序。不同的编译系统不一定相同，可以从左到右，也可从右到左。TurboC是按从右到左进行的  
main(){  
inti=8;  
printf("%d\n%d\n%d\n%d\n%d\n",++i,--i,i--,i++,-i--);  
}  
   
6.特殊用法  
   
对于m.n的格式还可以用如下方法表示（例）  
charch[20];  
printf("%\*.\*s\n",m,n,ch);  
前边的\*定义的是总的宽度，后边的定义的是输出的个数。分别对应外面的参数m和n。我想这种方法的好处是可以在语句之外对参数m和n赋值，从而控制输出格式。  
今天(06.6.9)又看到一种输出格式%n可以将所输出字符串的长度值赋绐一个变量,见下例:  
intslen;  
printf("hello world%n", &slen);  
执行后变量被赋值为11。

# 二、c++cout 输出格式

　在c++程序里面经常见到下面的头文件  
　　#include<iomanip>   
　　io代表输入输出，manip是manipulator（操纵器）的缩写  
　　iomanip的作用:  
　　主要是对cin,cout之类的一些操纵运算子，比如setfill,setw,setbase,setprecision等等。它是I/O流控制头文件,就像C里面的格式化输出一样.以下是一些常见的控制函数的:  
　　dec置基数为10相当于"%d"  
　　hex置基数为16相当于"%X"  
　　oct置基数为8相当于"%o"  
　　setfill('c' ) 设填充字符为c  
　　setprecision(n ) 设显示小数精度为n位  
　　setw(n ) 设域宽为n个字符  
　　这个控制符的意思是保证输出宽度为n。如：  
　　cout<< setw( 3 ) << 1 << setw( 3 ) << 10 <<setw( 3 ) << 100 << endl; 输出结果为  
　　110100 （默认是右对齐）当输出长度大于3时(<<1000)，setw(3)不起作用。  
　　▲setw(n)用法：通俗地讲就是预设宽度  
　　如cout<<setw(5)<<255<<endl;  
　　结果是:  
　　(空格)(空格)255  
　　▲setfill(charc) 用法:就是在预设宽度中如果已存在没用完的宽度大小，则用设置的字符c填充  
　　如cout<<setfill(‘@‘)<<setw(5)<<255<<endl;  
　　结果是:  
　　@@255  
　　▲setbase(intn) : 将数字转换为n进制.  
　　如cout<<setbase(8)<<setw(5)<<255<<endl;  
　　cout<<setbase(10)<<setw(5)<<255<<endl;  
　　cout<<setbase(16)<<255<<endl;  
　　结果是:  
　　(空格)（空格)377  
　　(空格)(空格)255   
　　(空格)(空格)f f   
　　▲setprecision用法  
　　使用setprecision(n)可控制输出流显示浮点数的数字个数。C++默认的流输出数值有效位是6。  
　　如果setprecision(n)与setiosflags(ios::fixed)合用，可以控制小数点右边的数字个数。setiosflags(ios::fixed)是用定点方式表示实数。  
　　如果与setiosnags(ios::scientific)合用，可以控制指数表示法的小数位数。setiosflags(ios::scientific)是用指数方式表示实数。  
　　setiosflags(ios::fixed)固定的浮点显示  
　　setiosflags(ios::scientific)指数表示  
　　setiosflags(ios::left)左对齐  
　　setiosflags(ios::right)右对齐  
　　setiosflags(ios::skipws)忽略前导空白  
　　setiosflags(ios::uppercase)16进制数大写输出  
　　setiosflags(ios::lowercase)16进制小写输出  
　　setiosflags(ios::showpoint)强制显示小数点  
　　setiosflags(ios::showpos)强制显示符号  
　　举例：  
　　#include<iostream.h>   
　　#include<iomanip.h>   
　　usingnamespace std;   
　　intmain()   
　　{  
　　cout<<12345.0<<endl;//输出"12345"  
　　cout<<setiosflags(ios::fixed)<<setprecision(3)<<1.2345<<endl;输出"1.235"  
　　cout<<setiosflags(ios::scientific)<<12345.0<<endl;//输出"1.234500e+004"   
　　cout<<setprecision(3)<<12345.0<<endl;//输出"1.235e+004"（1.235e+004应改为1.23e+004）  
　　return0;   
　　}