**scanf() 格式控制符汇总**

|  |  |
| --- | --- |
| %c | 读取一个单一的字符 |
| %hd、%d、%ld | 读取一个十进制整数，并分别赋值给 short、int、long 类型 |
| %ho、%o、%lo | 读取一个八进制整数（可带前缀也可不带），并分别赋值给 short、int、long 类型 |
| %hx、%x、%lx | 读取一个十六进制整数（可带前缀也可不带），并分别赋值给 short、int、long 类型 |
| %hu、%u、%lu | 读取一个无符号整数，并分别赋值给 unsigned short、unsigned int、unsigned long 类型 |
| %f、%lf | 读取一个十进制形式的小数，并分别赋值给 float、double 类型 |
| %e、%le | 读取一个指数形式的小数，并分别赋值给 float、double 类型 |
| %g、%lg | 既可以读取一个十进制形式的小数，也可以读取一个指数形式的小数，并分别赋值给 float、double 类型 |
| %s | 读取一个字符串（以空白符为结束） |

**printf() 格式控制符汇总**

在C语言中，有三个函数可以用来在显示器上输出数据，它们分别是：

* puts()：只能输出字符串，并且输出结束后会自动换行，在《[第一个C语言程序](http://c.biancheng.net/view/1734.html)》中已经进行了介绍。
* putchar()：只能输出单个字符，在《[在C语言中使用英文字符](http://c.biancheng.net/view/1766.html)》中已经进行了介绍。
* printf()：可以输出各种类型的数据，在前面的很多章节中都进行了介绍。

printf() 是最灵活、最复杂、最常用的输出函数，完全可以替代 puts() 和 putchar()，大家一定要掌握。前面的章节中我们已经介绍了 printf() 的基本用法，本节将重点介绍 printf() 的高级用法。

对于初学者，这一节的内容可能有些繁杂，如果你希望加快学习进度，尽早写出有趣的代码，也可以跳过这节，后面遇到不懂的 printf() 用法再来回顾。

首先汇总一下前面学到的格式控制符：

|  |  |
| --- | --- |
| **格式控制符** | **说明** |
| %c | 输出一个单一的字符 |
| %hd、%d、%ld | 以十进制、有符号的形式输出 short、int、long 类型的整数 |
| %hu、%u、%lu | 以十进制、无符号的形式输出 short、int、long 类型的整数 |
| %ho、%o、%lo | 以八进制、不带前缀、无符号的形式输出 short、int、long 类型的整数 |
| %#ho、%#o、%#lo | 以八进制、带前缀、无符号的形式输出 short、int、long 类型的整数 |
| %hx、%x、%lx %hX、%X、%lX | 以十六进制、不带前缀、无符号的形式输出 short、int、long 类型的整数。如果 x 小写，那么输出的十六进制数字也小写；如果 X 大写，那么输出的十六进制数字也大写。 |
| %#hx、%#x、%#lx %#hX、%#X、%#lX | 以十六进制、带前缀、无符号的形式输出 short、int、long 类型的整数。如果 x 小写，那么输出的十六进制数字和前缀都小写；如果 X 大写，那么输出的十六进制数字和前缀都大写。 |
| %f、%lf | 以十进制的形式输出 float、double 类型的小数 |
| %e、%le %E、%lE | 以指数的形式输出 float、double 类型的小数。如果 e 小写，那么输出结果中的 e 也小写；如果 E 大写，那么输出结果中的 E 也大写。 |
| %g、%lg %G、%lG | 以十进制和指数中较短的形式输出 float、double 类型的小数，并且小数部分的最后不会添加多余的 0。如果 g 小写，那么当以指数形式输出时 e 也小写；如果 G 大写，那么当以指数形式输出时 E 也大写。 |
| %s | 输出一个字符串 |

printf() 格式控制符的完整形式如下：

%[flag][width][.precision]type

[ ] 表示此处的内容可有可无，是可以省略的。

1. type 表示输出类型，比如 %d、%f、%c、%lf，type 就分别对应 d、f、c、lf；再如，%-9d中 type 对应 d。  
     
   type 这一项必须有，这意味着输出时必须要知道是什么类型。  
     
   2) width 表示最小输出宽度，也就是至少占用几个字符的位置；例如，%-9d中 width 对应 9，表示输出结果最少占用 9 个字符的宽度。  
     
   当输出结果的宽度不足 width 时，以空格补齐（如果没有指定对齐方式，默认会在左边补齐空格）；当输出结果的宽度超过 width 时，width 不再起作用，按照数据本身的宽度来输出。

下面的代码演示了 width 的用法：

1. #include <stdio.h>
2. int main(){
3. int n = 234;
4. float f = 9.8;
5. char c = '@';
6. char \*str = "http://c.biancheng.net";
7. printf("%10d%12f%4c%8s", n, f, c, str);
8. **return** 0;
9. }

运行结果：

234 9.800000 @http://c.biancheng.net

对输出结果的说明：

* n 的指定输出宽度为 10，234 的宽度为 3，所以前边要补上 7 个空格。
* f 的指定输出宽度为 12，9.800000 的宽度为 8，所以前边要补上 4 个空格。
* str 的指定输出宽度为 8，"http://c.biancheng.net" 的宽度为 22，超过了 8，所以指定输出宽度不再起作用，而是按照 str 的实际宽度输出。

3) .precision 表示输出精度，也就是小数的位数。

* 当小数部分的位数大于 precision 时，会按照四舍五入的原则丢掉多余的数字；
* 当小数部分的位数小于 precision 时，会在后面补 0。

另外，.precision 也可以用于整数和字符串，但是功能却是相反的：

* 用于整数时，.precision 表示最小输出宽度。与 width 不同的是，整数的宽度不足时会在左边补 0，而不是补空格。
* 用于字符串时，.precision 表示最大输出宽度，或者说截取字符串。当字符串的长度大于 precision 时，会截掉多余的字符；当字符串的长度小于 precision 时，.precision 就不再起作用。

请看下面的例子：

1. #include <stdio.h>
2. int main(){
3. int n = 123456;
4. double f = 882.923672;
5. char \*str = "abcdefghi";
6. printf("n: %.9d %.4d\n", n, n);
7. printf("f: %.2lf %.4lf %.10lf\n", f, f, f);
8. printf("str: %.5s %.15s\n", str, str);
9. **return** 0;
10. }

运行结果：

n: 000123456 123456

f: 882.92 882.9237 882.9236720000

str: abcde abcdefghi

对输出结果的说明：

* 对于 n，.precision 表示最小输出宽度。n 本身的宽度为 6，当 precision 为 9 时，大于 6，要在 n 的前面补 3 个 0；当 precision 为 4 时，小于 6，不再起作用。
* 对于 f，.precision 表示输出精度。f 的小数部分有 6 位数字，当 precision 为 2 或者 4 时，都小于 6，要按照四舍五入的原则截断小数；当 precision 为 10 时，大于 6，要在小数的后面补四个 0。
* 对于 str，.precision 表示最大输出宽度。str 本身的宽度为 9，当 precision 为 5 时，小于 9，要截取 str 的前 5 个字符；当 precision 为 15 时，大于 9，不再起作用。

4) flag 是标志字符。例如，%#x中 flag 对应 #，%-9d中 flags 对应-。下表列出了 printf() 可以用的 flag：

|  |  |
| --- | --- |
| **标志字符** | **含  义** |
| - | -表示左对齐。如果没有，就按照默认的对齐方式，默认一般为右对齐。 |
| + | 用于整数或者小数，表示输出符号（正负号）。如果没有，那么只有负数才会输出符号。 |
| 空格 | 用于整数或者小数，输出值为正时冠以空格，为负时冠以负号。 |
| # | * 对于八进制（%o）和十六进制（%x / %X）整数，# 表示在输出时添加前缀；八进制的前缀是 0，十六进制的前缀是 0x / 0X。 * 对于小数（%f / %e / %g），# 表示强迫输出小数点。如果没有小数部分，默认是不输出小数点的，加上 # 以后，即使没有小数部分也会带上小数点。 |

请看下面的例子：

1. #include <stdio.h>
2. int main(){
3. int m = 192, n = -943;
4. float f = 84.342;
5. printf("m=%10d, m=%-10d\n", m, m); //演示 - 的用法
6. printf("m=%+d, n=%+d\n", m, n); //演示 + 的用法
7. printf("m=% d, n=% d\n", m, n); //演示空格的用法
8. printf("f=%.0f, f=%#.0f\n", f, f); //演示#的用法
9. **return** 0;
10. }

运行结果：

m= 192, m=192

m=+192, n=-943

m= 192, n=-943

f=84, f=84.

对输出结果的说明：

* 当以%10d输出 m 时，是右对齐，所以在 192 前面补七个空格；当以%-10d输出 m 时，是左对齐，所以在 192 后面补七个空格。
* m 是正数，以%+d输出时要带上正号；n 是负数，以%+d输出时要带上负号。
* m 是正数，以% d输出时要在前面加空格；n 是负数，以% d输出时要在前面加负号。
* %.0f表示保留 0 位小数，也就是只输出整数部分，不输出小数部分。默认情况下，这种输出形式是不带小数点的，但是如果有了#标志，那么就要在整数的后面“硬加上”一个小数点，以和纯整数区分开。