

# Especificadores de formato para tipos flotantes en C

Guillermo Hernández

2025-10-06

Versión en línea: <https://www.dih5.es/floating-specifiers.html>.

En C tenemos dos tipos de datos primitivos para representar números en **coma flotante**: `float` y `double`. Si queremos usarlos correctamente con las funciones de entrada y salida estándar, tendremos que utilizar los especificadores de formato adecuados, pero, ocurre algo interesante de comentar, que vamos a motivar con una pregunta inicial, como en otras entradas de esta página dedicadas al lenguaje C.

¿Cuál de las siguientes afirmaciones es *más correcta* sobre el especificador de formato que se debe usar con una variable de tipo `float`?

- Siempre se debe usar `%f` en `scanf` y `printf`.
- Se debe usar `%lf` en `scanf`, pero en `printf` se debe usar `%f`.
- Se debe usar `%f` en `scanf`, pero en `printf` se debe usar `%lf`.
- Es imposible pasarle un `float` a `printf`.

La respuesta *más correcta* es... ¡la última! (y la primera es más o menos correcta, con la “letra pequeña” de la última). Veamos por qué.

Si revisamos la tabla de [especificadores de formato de `scanf`](#) y [la de `printf`](#), podremos notar una ausencia en la segunda: la del tipo `float`. Si bien en `scanf` ocurre que `%f` se corresponde con `float` y `%lf` con `double`; en `printf` tan solo aparece `%f` asociado a `double`. ¿Por qué ocurre esto? Estas dos funciones son especiales en el sentido de que toman un **número variable de argumentos**, lo que a veces se denomina “variadicas”. En realidad es una característica que es parte de la biblioteca estándar de C, y se pueden construir usando las herramientas disponibles en [`stdarg.h`](#). Una peculiaridad de los argumentos adicionales de estas funciones siguen unas reglas de promoción automáticas, de modo que todo `float` se transforma automáticamente en `double`. Este es el motivo por el que el especificador `%f` se asocia a `double` en `printf`: es imposible que reciba un `float`, pues promocionará automáticamente en `double`. Observa que esto no ocurre en `scanf` porque lo que se recibe no es un `float`, sino un `float *`, esto es la dirección de memoria de un `float`.

En algunas tablas de printf es posible encontrar %lf como especificador de formato para double. En efecto, esto está permitido a partir de C99 (cf. p. xii en la ISO/IEC 9899:1999). En este estándar, se puede usar siempre %f **al trabajar con float y %lf al hacerlo con double** (aunque un float nunca llegue a printf como tal), lo que facilita escribir código correcto.

A continuación tienes un código con el que puedes probar cómo se comporta el tipo double ante estos dos especificadores. En particular, puedes usarlo para comprobar que usar %f con double en scanf daría un error, mientras que en printf todo funciona. Puedes modificarlo también para estudiar lo mismo con float.

```
#include <stdio.h>

/*
 * Programa de demostración de especificador de formato para tipo double
 * Más información:
 * https://stackoverflow.com/questions/4264127/correct-format-specifier-for-double-in-printf
 * */
int main()
{
    double x=0;

    puts("Muestra de error en especificadores de formato %f por %lf con doubles.");
    puts("Introducir un valor distinto de 0 para probarlo.");
    puts("Se puede comprobar que para la lectura es obligatorio usar %lf, pero para la escritura %f.");
    puts("Esto ocurre porque printf convierte los argumentos float a double.");

    puts("Lectura %lf, escritura %lf");
    printf("> ");
    scanf("%lf", &x);
    printf("%lf\n\n", x);
    x=0;

    puts("Lectura %f, escritura %lf");
    printf("> ");
    scanf("%f", &x);
    printf("%lf\n\n", x);
    x=0;

    puts("Lectura %lf, escritura %f");
    printf("> ");
    scanf("%lf", &x);
    printf("%f\n\n", x);
    x=0;

    puts("Lectura %f, escritura %f");
```

```
printf("> ");  
scanf("%f", &x);  
printf("%f\n\n", x);  
x=0;  
  
return 0;  
  
}
```