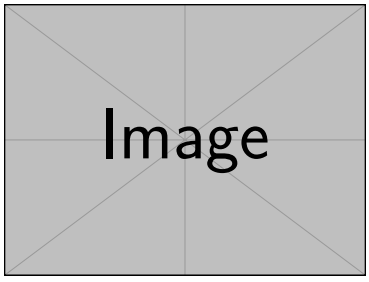
printf() Function Documentation

Formatting & Printing in C

Complete Guide to Format Specifiers

Integer, Float, String, and Character Formatting



December 22, 2025

Version 1.0
Last updated: December 22, 2025

Contents

Introduction

Introduction

The `printf()` function is a fundamental part of the C Standard Library, used for formatted output to the standard output stream. This documentation covers all aspects of formatting with comprehensive examples.

1 Integer Formatting

1.1 Printing Strings and Integer Variables

```c
#include <stdio.h>

int main() {
    int age = 25;
    int count = 150;

    // Basic integer printing
    printf("Age: %d\n", age);           // Output: Age: 25

    // Multiple integers
    printf("Count: %d, Age: %d\n", count, age);

    // Integer with text
    printf("There are %d items in stock\n", count);

    // Hexadecimal and octal
    printf("Hex: %x, Octal: %o\n", 255, 255);

    return 0;
}
```

Listing 1: Basic Integer Formatting

1.2 Width Specification

```c
#include <stdio.h>

int main() {
    int num1 = 42;
    int num2 = 1234;

    // Right-aligned with width
    printf("|%5d|\n", num1);     // |   42| - Width of 5
    printf("|%8d|\n", num2);     // |    1234| - Width of 8

    // Left-aligned with width
    printf("|%-5d|\n", num1);    // |42   | - Left aligned

    // Zero padding
    printf("|%05d|\n", num1);    // |00042| - Zero padded

    // Sign display
```

```
    printf("|%+5d|\n", 42);      // |  +42|
    printf("|%+5d|\n", -42);     // |  -42|

    return 0;
}
```

Listing 2: Integer Width Specification

> ⚠ **Note:** The width includes all characters (digits, signs, and spaces). Make sure your width is sufficient for the entire number.

2 Float Formatting

2.1 Precision Specification

```c
#include <stdio.h>

int main() {
    float price = 19.99;
    double pi = 3.14159265359;

    // Default precision (6 decimal places)
    printf("Price: %f\n", price);        // Price: 19.990000

    // Specify decimal precision
    printf("Price: %.2f\n", price);      // Price: 19.99
    printf("Pi: %.4f\n", pi);            // Pi: 3.1416

    // Width and precision combined
    printf("|%10.2f|\n", price);         // |     19.99|
    printf("|%-10.2f|\n", price);        // |19.99     |

    // Different float specifiers
    printf("Default: %f\n", 123.456789);  // 123.456789
    printf("Scientific: %e\n", 123.456789); // 1.234568e+02
    printf("Shorter: %g\n", 123.456789);   // 123.457
    printf("Shorter: %g\n", 0.00001234);   // 1.234e-05

    return 0;
}
```

Listing 3: Float Precision Control

3 String and Character Formatting

3.1 Printing Strings and Characters

```c
#include <stdio.h>

int main() {
    char letter = 'A';
    char name[] = "John";
    char message[] = "Hello, World!";
```

```c

    // Character printing
    printf("Letter: %c\n", letter);      // Letter: A
    printf("ASCII value: %d\n", letter); // ASCII value: 65

    // String printing
    printf("Name: %s\n", name);          // Name: John

    // Multiple strings
    printf("%s %s\n", "Hello", "World");

    // Escape sequences
    printf("Tab:\tColumn1\tColumn2\n");
    printf("New\nLine\n");
    printf("Path: C:\\Users\\Name\\file.txt\n");
    printf("Quotes: \"%s\"\n", name);

    return 0;
}
```

Listing 4: String and Character Formatting

3.2 Setting the Width for Strings

```c
#include <stdio.h>

int main() {
    char str[] = "Hello";
    char name[] = "Alice";
    char city[] = "New York";

    // Right-aligned string
    printf("|%10s|\n", str);        // |     Hello|

    // Left-aligned string
    printf("|%-10s|\n", str);       // |Hello     |

    // Maximum characters (truncation)
    printf("|%.4s|\n", str);        // |Hell|

    // Width and maximum combined
    printf("|%10.4s|\n", str);      // |      Hell|

    // Table formatting example
    printf("%-15s %-10s %8s\n", "Product", "Category", "Price");
    printf("%-15s %-10s %8.2f\n", "Laptop", "Electronics", 999.99);
    printf("%-15s %-10s %8.2f\n", "Book", "Education", 29.99);

    return 0;
}
```

Listing 5: String Width Control

4 Format Specifiers Reference

4.1 Complete Format Specifiers Table

Specifier	Description
%d or %i	Signed integer
%u	Unsigned integer
%f	Floating point (decimal notation)
%e or %E	Scientific notation
%g or %G	Shorter of %e or %f
%c	Single character
%s	String
%p	Pointer address
%x or %X	Hexadecimal integer
%o	Octal integer
%%	Percent sign

Table 1: Format Specifiers Summary

4.2 Format Syntax Structure

The complete format for `printf()` specifiers is:

`%[flags][width][.precision][length]specifier`

Component	Symbol	Description
Flags	-	Left-justify within width
	+	Always show sign (+ or -)
	0	Pad with zeros
	space	Space for positive numbers
	#	Alternate form
Width	number	Minimum field width
Precision	.number	Precision for floats, max chars for strings

Table 2: Format Modifiers

5 Advanced Examples

5.1 Complex Formatting Combinations

```c
#include <stdio.h>

int main() {
    // Employee records with mixed formatting
    printf("\n%-20s %-15s %8s %12s %10s\n",
           "Employee Name", "Department", "ID", "Salary", "Rating");
    printf("%-20s %-15s %08d $%11.2f %9.1f%%\n",
           "John Smith", "Engineering", 1001, 75500.50, 95.5);
    printf("%-20s %-15s %08d $%11.2f %9.1f%%\n",
           "Maria Garcia", "Marketing", 1002, 62000.00, 88.0);
    printf("%-20s %-15s %08d $%11.2f %9.1f%%\n",
```

```
            "David Chen", "Sales", 1003, 58000.75, 92.5);

    // Product catalog
    printf("\n%-15s %-10s %-8s %12s\n",
           "Product", "Category", "Stock", "Price");
    printf("%-15s %-10s %8d $%11.2f\n",
           "Laptop Pro", "Electronics", 25, 1299.99);
    printf("%-15s %-10s %8d $%11.2f\n",
           "Desk Chair", "Furniture", 150, 299.50);
    printf("%-15s %-10s %8d $%11.2f\n",
           "Coffee Maker", "Appliances", 42, 89.99);

    // Scientific data
    printf("\nScientific Data:\n");
    printf("Speed of light: %.2e m/s\n", 299792458.0);
    printf("Avogadro's number: %.4e mol^-1\n", 6.02214076e23);
    printf("Planck constant: %.3e J·s\n", 6.62607015e-34);

    return 0;
}
```

Listing 6: Advanced Formatting Example

5.2 Common Pitfalls and Solutions

⚠ Common Issues:

- **Mismatched types**: Using %d for float or %s for int
- **Insufficient width**: Number gets truncated or misaligned
- **Missing arguments**: More format specifiers than arguments
- **Buffer overflow**: Using %s without length limits on user input

Best Practices

1. **Always check format specifiers** match the variable types
2. **Use appropriate width** to ensure proper alignment in tables
3. **Consider localization** for decimal separators in international applications
4. **Validate user input** before using in format strings
5. **Use constants** for format strings that are reused
6. **Test edge cases**: large numbers, negative values, empty strings

Conclusion

The `printf()` function is a powerful tool for formatted output in C. Mastering its formatting capabilities is essential for creating professional-looking console applications. Always refer to the C standard library documentation for the most up-to-date information and platform-specific behaviors.

Quick Reference Card

`%[flags][width][.precision]specifier`

`%-10.2f` = Left-align, width 10, 2 decimal places

`%08d` = Zero-padded, width 8 integer

`%.10s` = First 10 characters of string