Common and Advanced Output Formats in C Using `printf`

Generated by ChatGPT

August 11, 2025

Introduction

This document covers common and advanced formatting techniques for output in C using the `printf` function. It shows various format specifiers, flags, width, precision, and how to print different data types, including some complex examples.

1 Basic Output Formats Recap

2 Extended Integer Formats

- `%ld`, `%li`: Signed `long` integer.
- `%lu`: Unsigned `long` integer.
- `%lld`, `%lli`: Signed `long long` integer.
- `%llu`: Unsigned `long long` integer.

```
// Printing long and long long integers
long lnum = 1234567890L;
long long llnum = 1234567890123456789LL;
printf("Long int: %ld\n", lnum);
printf("Long long int: %lld\n", llnum);
```

3 Field Width, Precision, and Flags in Depth

- `%5d` - Minimum width 5, right-aligned by default.
- `%-5d` - Left-align within width 5.
- `%05d` - Pad with zeros to width 5.
- `%+.2f` - Always show sign and 2 decimal places.
- `%,d` - Locale-specific thousands separator (not standard C, supported in some implementations).

```c
int num = 42;
float pi = 3.14159;

printf("Right align, width 5: '%5d'\n", num);
printf("Left align, width 5:  '%-5d'\n", num);
printf("Zero pad, width 5:    '%05d'\n", num);

printf("Signed float, 2 decimals: '%+.2f'\n", pi);

// Example of thousands separator - may not work on all systems
// printf("Thousands separator: '%,d'\n", 1000000);
```

4 Escaping the Percent Sign

To print a literal percent sign, use `%%`.

```c
printf("Progress: 90%% complete\n"); // prints: Progress: 90% complete
```

5 Printing Multiple Variables

You can print multiple values by listing format specifiers and corresponding arguments.

```c
int x = 10, y = 20;
printf("x = %d, y = %d\n", x, y);
```

6 Printing Binary (Custom)

C standard `printf` has no `%b` specifier for binary output. Use a helper function:

```c
void print_binary(unsigned int n) {
    for (int i = sizeof(n)*8 - 1; i >= 0; i--) {
        putchar((n & (1 << i)) ? '1' : '0');
    }
    putchar('\n');
}

int main() {
    unsigned int val = 13;
    printf("Binary of %d is: ", val);
    print_binary(val);
    return 0;
}
```

7 Advanced Floating-Point Examples

```c
double val = 12345.6789;

// Scientific notation uppercase
printf("Scientific (%%E): %E\n", val);

// Use %g to switch automatically
printf("Auto format (%%g): %g\n", val);

// Width and precision combined
printf("Width 12, precision 4: '%12.4f'\n", val);
```

8 Summary Table of Advanced Specifiers and Flags

Specifier	Description
`%ld`, `%li`	Signed long integer
`%lu`	Unsigned long integer
`%lld`, `%lli`	Signed long long integer
`%llu`	Unsigned long long integer
`%05d`	Pad integer with zeros, width 5
`%-5d`	Left-justify integer within width 5
`%+.2f`	Always show sign for float, 2 decimals
`%%`	Literal percent sign
`%p`	Pointer address
`%E`	Floating-point scientific notation (uppercase)
`%g`	General format, switches between `%f` and `%e`