Formatted I/O: printf

CSC 100 Introduction to programming in C/C++, Summer 2022

Marcus Birkenkrahe

April 14, 2023

Contents

1 README

- There is much more to `scanf` and `printf` than we've seen
- I/O is where the pedal hits the metal - where man meets machine
- In this notebook: conversion specifications for `printf`
- Practice workbooks, input files and PDF solution files in GitHub

2 printf

When it is called, `printf` must be supplied with:

1. a format string, like `"The output is:  %d\n"` (actually only "%d")
2. any number of values to be inserted into the string at printing
3. the values can also be computed and assigned

3 Conversion specification

- A **conversion specification** is a placeholder like `d`
- Binary (machine) format is converted to printed (human) format
- General form: `%m.pX` where

	WHAT	EXAMPLE
m	minimum field width	`%4d` prints `123` as `_123`
p	precision after point	`%.3f` prints `3.141593` as `3.142`
X	conversion specifier	`d`, `e`, `f`, `g`

4 Examples:

```
printf("....|....|....|\n");

printf("%8d\n", 123); // print 123 on 8 places (right-aligned)

printf("%-8d\n", 123); // print 123 on 8 places (left-aligned)

printf("%10.3f\n", 3.141593); // print 3 decimals on 10 places (right)

printf("%-10.3f\n", 3.141593); // print 3 decimals on 10 places (left)
```

```
....|....|....|
     123
123
     3.142
3.142
```

5 Integer decimal `d`

- `d` displays an integer in decimal (= base 10) form. `p` is the minimum number of digits to display the integer. Default is `p=1`.
- For example, the code below **??** prints numbers with different precision values:
  - `%d` displays `int` in decimal form (minimum amount of space)
  - `%5d` displays `int` in decimal form using 5 characters
  - `%-5d` displays `int` on 5 characters, left-justified
  - `%5.3d` displays `int` on 5 characters, at least 3 digits

```
int i = 40;
printf("....|....|\n");
printf("%d\n",i); // decimal form (minimum amount of space)
printf("%5d\n",i); // decimal form using 5 characters
printf("%-5d\n",i); // on 5 characters, left-justified
printf("%5.4d\n",i); // on 5 characters, at least 3 digits
```

```
....|....|
40
   40
40
 0040
```

6 Floating point exponential "e"

- `e` displays a floating-point number in exponential ("scientific") notation, e.g. `10. * 10. * 10. = 1000. = 1.0e+03`.
- `p` indicates the digits after decimal point. If `p=0`, no decimal point is displayed.

What went wrong in the first two statements?

```
printf("....|....|....|\n");
printf("%.e\n", 1.);
printf("%-15.3e\n", 1000.);
printf("%e\n", 1000000000000000.);
printf("%15.e\n", 1000000000000000.);
```

```
....|....|....|
1e+00
1.000e+03
1.000000e+15
          1e+15
```

7 Floating point fixed decimal "f"

That's `f` as we already know it from many other examples. The precision p is defined as for `e`. Trailing zeroes are shown.

```
printf("....|....|\n");
printf("%10.3f\n", 100.1);
```

```
....|....|
   100.100
```

8 Variable floating point "g"

- `g` displays a floating point number in either exponential format or fixed decimal format depending on the number's size.
- `p` is the maximum number of **significant** digits (**not** digits after the decimal point!) to be displayed.
- No trailing zeroes are shown. If there are no decimal digits after the decimal point, no decimal point is shown.
- How many lines and number are you expecting?

  ```
  printf("%g\n%g\n%g\n", 299792458., 1.45e+03, 8000.);
  ```

  ```
  2.99792e+08
  1450
  8000
  ```