Formatted I/O: printf

CSC 100 Introduction to programming in C/C++

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

printf

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

1. a format string, like `"The output is: %d\n`
2. any number of values to be inserted into the string at printing

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`

- `[X]`

  Let's check! Add the examples from the first two lines of the table in the code block 1 and run it.

```
printf("%4d\n", 123);
printf("%.3f\n", 3.141593);
```

```
 123
3.142
```

- The precision p depends on the conversion specifier X

Integer decimal "d"

`d` displays an integer in decimal (= base 10) form. p is the minimum number of digits to display the integer. Default p=1.

- [X]

  Get coding! Show that the default for `d` is p=1. Print the numbers 1, 1, 100 and 10000 with the specifiers `%d`, `%.1d`, `%.5d`, `%.2d`. Print each expression on its own line.

```
printf("%d\n %.1d\n %.5d\n %.2d\n", 1, 1, 100, 10000);
```

```
    1
    1
  100
10000
```

Floating point exponential "e"

`e` displays a floating-point number in exponential ("scientific") notation. p = digits after decimal point. If p=0, no decimal point is displayed.

- [X]

  Get coding! Print 1, 1000, and 1000000000000000 using %e%. Print each expression on its own line with its own print function.

```
printf("%e\n", 1); // forgot the decimal point
printf("%e\n", 1.);
printf("%e\n", 1000.);
printf("%e\n", 1000000000000000.);
```

```
1.421737e-312
          1.0
       1000.0
         1e+15
```

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.

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.

- [ ]

Get coding! Use `g` to display the numbers: 200, 3.142574654 with p=8, 2.71 - print each on a line of its own, but use only **one** `printf` statement to do it!

```
printf("%g\n %.8g\n %g\n", 200., 3.142574654, 2.71);
```

```
200
3.1425747
2.71
```

Author: Marcus Birkenkrahe
Created: 2022-03-11 Fri 09:02
Validate