Mastering printf in C - Sample Solution

Programming assignment CSC 100 Lyon College Spring 2024

Marcus Birkenkrahe

March 10, 2024

Format and Print Integers

Problem

- Write a program that declares an integer variable.
- Assign the value 2 to the variable.
- Print this integer in four different ways using `printf`:
  - With a minimum field width of 8 characters.
  - Right-aligned and left-aligned within a field width of 10 characters.
  - With a field length of 10 and at least 4 digits, padding with zeros if necessary.
- Include your code in an Org-mode code block and run it.

Solution

```
int i = 2; // declare integer variable

// print ruler
printf("....|....|....|....|\n");

// print statements
printf("%8d\n",i); // field length 8
printf("%10d\n",i); // right aligned 10 fields
printf("%-10d\n",i); // left aligned 10 fields
printf("%10.4d\n",i); // left aligned 10 fields
```

```
....|....|....|....|
       2
         2
2
      0002
```

Observation: If you don't assign a value to the variable, any value can be found, and every time you run the block, the value changes.

```
//int i = 2; // declare integer variable
int i;

// print ruler
printf("....|....|....|....|\n");

// print statements
printf("%8d\n",i); // field length 8
printf("%10d\n",i); // right aligned 10 fields
printf("%-10d\n",i); // left aligned 10 fields
printf("%10.4d\n",i); // left aligned 10 fields
```

```
....|....|....|....|
   22004
     22004
22004
     22004
```

Floating Point Numbers

Problem

- Declare a floating-point variable and assign the value 3.141593 to it. What happens if you don't assign a value?
- Print this variable in three different formats:
  - As a floating-point number with 2 decimal places.
  - In exponential format with 3 digits after the decimal point.
  - Using the 'g' conversion specifier with a precision of 5 significant digits.
- Include your code in an Org-mode code block and run it.

Solution

```
float x = 3.141593f; // declare variable and assign value

// print statements
printf("%.2f\n", x); // two significant digits
printf("%.3e\n", x); // three significant digits, e notation
printf("%.5g\n", x); // five digits, variable format
```

```
3.14
3.142e+00
3.1416
```

If I don't assign a value, and only declare a `float` variable, the `%f` formatted statement is 0.00, and the other two statements are approximately also 0, because 1.e-41 is a very small number. Instead of picking any number, undefined floating-point variables seem to be set to practically zero.

```
//float x = 3.141593f; // declare variable and assign value
float x;

// print statements
printf("%.2f\n", x);
printf("%.3e\n", x);
printf("%.5g\n", x);
```

```
0.00
3.067e-41
3.0669e-41
```

Challenging Formatting

Problem

- Using a single `printf` statement, print a sequence that includes:
  - An integer with left alignment in a 15-character wide field.
  - A floating-point number in fixed decimal format, with 3 decimal places.
  - A floating-point number in exponential format with the width of 20 characters, ensuring the exponent is always displayed.
- Include your code in an Org-mode code block and run it.

Solution

To use a single `printf` statement, I use one string with multiple format specifiers, and add the variables separated by comma.

I have to declare and assign values to the integer and floating point variables. I can put the string and the variables on different lines for better readability.

```
// declare variables
int i = 2;
float x1 = 2.718282f;
float x2 = 1000000000.f;

// print ruler
printf("....|....|....|....|\n");

// print statements:
// 1. left aligned integer on 15 fields
// 2. floating-point number with 3 significant digits
// 3. exponential (scientific) format on 20 fields
printf("%-15d\n%.3f\n%20e\n",
       i, x1, x2);
```

```
....|....|....|....|
2
2.718
        1.000000e+09
```