UNIVERSITY OF MASSACHUSETTS
DARTMOUTH

ECE160: Foundations of Computer Engineering I

Lecture #6 – Formatted Output prinf()

Instructor: Dr. Liudong Xing
SENG-213C, lxing@umassd.edu
ECE Dept.

Administrative Issues

- Homework #1 due **Monday, Jan. 30 (Today)**
  - Please follow the "submission guidelines" available in the course website to submit your answers to your name folder at the class M: drive if you haven't
  - **Late submission is subject to penalty.**

- Lab#2 assigned
  - Due by **5pm, Wednesday, Feb. 1**

- Lab#1 grade is available from M: drive
  - Please send your grade concern to TA Peter (glv@umassd.edu) and cc to me if there is any

Review of Lecture #5

- Constants
  - Four types: integer, floating point, character, string
  - Three ways to code constants in the program:
    - Literal: an unnamed constant used to specify data
    - Defined: use the preprocessor command #define name expression (the expression that follows the name in the command replaces the name wherever it is found in the source program)
    - Memory: use a C type qualifier: const type identifier = value; (memory constants fix the contents of a memory location)

Outline

- Overview on formatted input/output
- Formatted output printf()
- Formatted input scanf() (Lecture#7)

Formatted Input/Output

- C gives us the ability to read data from the keyboard and print data on the monitor.
- The mechanism through which input and output happens in C is a file.
  - The keyboard is considered the standard input file, which is buffered, that is we can change our input, using the backspace key, prior to pressing the Enter key.
  - The monitor is the standard output file.

Formatted Input/Output

- Formatted input in C is done through function scanf()
- Formatted output in C is done through function printf()

- A function is a piece of code which performs a specific task.
  - A function is ***called*** or ***invoked***, which causes it to execute.
  - A function is composed of the function name, an open parenthesis, a set of function arguments separated by commas, and a close parenthesis:

    function_name(arg1, arg2, arg3….)

Format Output

Textbook: Chapter 7.1, 7.2

printf()

- printf() needs two things:
  - Instructions for formatting the data, and
  - The actual data to be printed.

- printf(format string, data list);

Example

printf("Hello! You are customer number: %d\n", a);

- The format string is the text that is to be displayed on the screen.
- It is enclosed in a set of quotation marks
- The % characters are called **placeholders**. They indicate the display position for variables whose values are to be displayed.
- \n prints a newline character.

- The variable names to be displayed are specified in the data list and appear in the same order as their placeholders.

Placeholders

- All placeholders begin with a “%”.
- The text after “%” indicates how to format the output, i.e., what kind of variable it is -- conversion code
- The number of conversion code should match the number of variables that follow the “format string”

Conversion Codes/Specifiers
(Examples)

%d	decimal number (int, e.g. 9, 10)
%f	floating-point number (float or double, e.g. 3.14)
%c	character (char, e.g., 'a', 'b')
%e	floating point number in scientific notation (e.g., 5.77748e+05 which equals $5.7748*10^5$)
%g	floating point number in e-format or f-format, whichever is shorter
%s	string (e.g., "Hello" or "abc")

Customizing Integer Output

Customizing Integer Output (1)

- "%d" is used to display an integer variable/value
- "%hd" is used to display a short integer variable/value
- "%Ld" is used to display a long integer variable/value

They can be altered to format how the number is displayed.

Customizing Integer Output (2)

- Instead of "%d", use a "%Xd" where the X is an integer that is the field width, specifying the number of digits to display.
- The negative sign (for negative integers) is also considered a digit here

- For example,
  - "%6d" displays 6 digits of the result (integer)
  - "%2hd" displays 2 digits of the result (short integer)
  - "%8Ld" displays 8 digits of the result (long integer)

Customizing Integer Output (3)

- When there are more places in the field width than digits to be displayed, the output is right-justified.

- When there are more digits than places, the output field width is ignored, and the entire integer is displayed.

Examples

Value	%d	%4d
12	12	12
123	123	123
1234	1234	1234
12345	12345	12345

right-justified

Exercises (1)

What will the following printf() print out?

```
printf("The number%d wins!", 5321);

printf("The number%6d wins!", 5321);
```

Customizing Integer Output (4)

- When there is a width specification, a flag modifier can be used to alter the display format
  - 0: the number will be printed with leading zeros
  - - (minus sign): the data are formatted left justified (data are pushed to the left and spaces are used to fill in the right portion of the width specification)

- For example,
  - "%-6d" displays 6 digits of the result, left justified
  - "%06d" displays 6 digits of the result, leading zeros

Exercises (2)

What will the following printf() print out?

```
printf("The number%6d wins!\n", 5321);
printf("The number%06d wins!\n", 5321);
printf("The number%-6d wins!\n", 5321);
```

Customizing Floating-Point Output

Customizing Floating-Point Output (1)

- Floating point output (float and double) can be formatted in the same manner, using "%X.Yf").
  - X is the total number of digits to display (i.e., the field width)
  - Y is precision width: the number of digits to display to the right of the decimal point, i.e., the number of decimal digits

- The same rules for field width apply as for integer formatting.

Customizing Floating-Point Output (2)

- The specified number of decimal digits is always displayed
- If no precision is specified, printf() prints 6 decimal positions
- “%Lf” indicates the type is long double

Examples

Value	Placeholder	Output
3.14159	%5.2f	3.14
3.14159	%3.2f	3.14
3.14159	%5.3f	3.142
0.1222	%4.2f	0.12
-0.009	%8.5f	-0.00900
The specified number of decimal digits is always displayed		
4.1	%f	4.100000

If no precision is specified, printf() prints 6 decimal positions

Customizing Character Output

Customizing Character Output

- The same rules for field width apply as for integer formatting

- For example:

```
printf("%c%6c", 'a', 'b');
a     b
printf(%c%06c", 'a', 'b');
a00000b
```

Exercises (3)

Show what the following printf statements print out:

- printf("%d%c%f", 23, 'a', 5.3);

- printf("%d %c %f", 23, 'a', 5.3);

- int num1=23;
  char bee = 'a';
  float num2=5.3;
  printf("%d %c %f", num1, bee, num2);

Exercises (4)

Show what the following printf statements print out:

- printf("%d\t%c\t%5.1f\n", 23, 'a', 51.3);
- printf("%d\t%c\t%5.1f\n", 107, 'A', 56.7);
- printf("%d\t%c\t%5.1f\n", 1753, 'D', 151.3);
- printf("%d\t%c\t%5.1f\n", 3, 'c', 0.3);

Exercises (5)

Show what the following printf statements print out:

- printf(“The number%dis my favorite number.”, 23);
- printf(“The number is %6d”, 23);
- printf(“The number is %06d”, 23);

Exercises (6)

Show what the following printf statements print out:

- printf("The tax is %6.2f this year.", 233.32);
- printf("The tax is %8.2f this year.", 233.32);
- printf("The tax is %08.2f this year.", 233.32);

Use of Special Characters in printf()

Use of \r in printf()

- printf("This line disappears.\r…A new line\n");
- The return character (\r) repositions the output at the beginning of the curent line without advancing the line. Therefore, all data that were placed in the output stream are erased

…A new line

Use of \0 in printf()

- printf("A null character\0kills the rest of the line\n");
- The null character (\0) effectively kills the rest of the line

A null character

Use of escape character \ in printf()

- Quotes are used to identify the format string in printf function, we cannot use them as print characters. To print them, we must use the escape character with the quote(\") which tell printf() that what follows is not the end of the string but a character to be printed.

- printf("This is \"it\" in double quotes\n");
- Output: This is "it" in double quotes

- Similarly, for single quote!

Common Output Errors

Exercise (7)

- Each of the following printf has at least one error. Try to find it.

```
Printf("%d %d %d\n", 33, 66);

printf("%d %d\n", 33, 44, 55)
```

Summary of Lecture #6

- Formatted output function printf()
  - Conversion codes %d %c %f etc
  - Field width specification
  - Flag modifiers: 0 and –
  - Use of special characters in printf()
  - Common errors

Things To Do

- Review lecture notes
- Homework & Lab assignments

Next Topic

- Formatted Input scanf()