Day16 A

Young W. Lim

2017-12-08 Fri

Outline

1. Based on

2. C Formatted IO
   - Formatting Output
   - Formatting Input
   - Conversion Specifiers
   - Field Widths, Precision and Flags
   - Summary Tables

Based on

"C How to Program",
Paul Deitel and Harvey Deitel

Format Control String

- describes the formats in which the output values appear

conversion specifiers
flags
field widths
precisions
literal characters

Scan Set

- a scan set scans the characters in the input
  looking only for those characters in the scan set
- when a character matches, it is stored in a character array
- the input operation stops
  when a character that is not in the scan set is encountered

Inverted Scan Set

- inverted scan set : place ^ (caret) in the square brackets before the scan characters
- starts inputing characters
  which are not in the inverted scan set
  storing them in a character array
- stops when a character is encountered,
  which are in the inverted scan set

Conversion Specifier n (1)

- the conversion specifier `n`
  stores the number of characters
  which have been read in the current scanf statement
- the corresponding argument is the pointer to `int`

Conversion Specifier n (2)

```
#include <stdio.h>

int main()
{
  int val;

  printf("abcd %n efgh\n", &val);
  printf("val = %d\n", val);
}

----
abcd efgh
val = 5
```

from https://stackoverflow.com/questions/3401156/what-is-the-use-of-the-n-fo

Conversion Specifier n (3)

```
int n;
printf("%s: %nFoo\n", "hello", &n);
printf("%*sBar\n", n, "");

-----
hello: Foo
       Bar

-----
int n = printf("%s: ", "hello");
printf("Foo\n");
printf("%*sBar\n", n, "");

from https://stackoverflow.com/questions/3401156/what-is-the-use-of-the-n-fo
```

Integer Conversion Specifiers

d	display as a signed decimal integer
i	display as a signed decimal integer
o	display as an unsigned octal integer
u	display as an unsigned decimal integer
x, X	display as an unsigned hexadecimal integer
	place before any integer conversion specifier
h	short
l	long
ll	long long

Floating Point Number Conversion Specifiers

e, E	display a floating-point value in exponential notation
f, F	display a floating-point value in fixed-point notation
g, G	either f, F or e, E
u	display as an unsigned decimal integer
x, X	display as an unsigned hexadecimal integer
	place before any floating-point conversion specifier
l	long double

Character and String Conversion Specifiers

c	prints a character
s	pritns a string of characters ending in null characters

Other Conversion Specifiers

p	prints an address (usually in hexadecimal)
%%	pritns % literally

Field Widths

- if the field width is larger than the object being printed, the object is right justified by default
- field width can be used with any conversion specifiers
- %{field width}.{precision}{conversion specifier}
- possible to specify the field width and the precision by an integer expression in the argument list
- %*.*{conversion specifier}
- the matching argument in list is used in place of the astersk

Precisions

- integer number precision:
  - indicates the minimu number of digits printed
  - zeros are prefixed to the printed value
- floating point number precsion:
  - indicates the number of digits
    that appear after the decimal point (f, F, e, E)
  - indicates the number of significant digits (g, G)
- string precision:
  - indicates the number of characters to be printed

Flags

- – flag:
  - left justifies its argument in a field
- + flag:
  - prints + or – sign
- space flag:
  - prints a space for a positive number
    which are not displayed with the + flag
- # flag:
  - prefixes 0 to octal numbers
  - prefixes 0x to hexadecimal numbers
  - forces to print decimal points for floating point numbers
- 0 flag:
  - prints leading zeros for a value
    which does not occupy its entire field width

Escape Sequence

```
\'   single quote
\"   double quote
\?   question mark
\\   back slash
\a   alert bell
\b   backspace
\f   new page or form feed
\n   newline
\r   carriage return
\t   horizontal tab
\v   vertical tab
```

Integer Conversion Specifier - Printing

`d`	display as a signed decimal integer
`i`	display as a signed decimal integer (i and d are different in scanf)
`o`	display as an unsigned octal integer
`u`	display as an unsigned decimal integer
`x,X`	display as an unsigned hexadecimal integer
`h, l, ll`	place before any integer conversion specifier to indicate that a short, long, long long integer is displayed : length modifiers

Integer Conversion Specifier - Reading

`d`	read an optionally signed decimal integer	`int *`
`i`	read an optionally signed decimal integer or hexadecimal integer	`int *`
`o`	read an octal integer	`unsigned int *`
`u`	read an unsigned decimal integer	`unsigned int *`
`x, X`	read a hexadecimal integer	`unsigned int *`
`h,l,ll`	place before any integer converison specifiers to indicate that a short, long, long long integer is to be input	

FP Number Conversion Specifier - Printing

e, E	displays a floating point number in exponential notation
f, F	displays a floating point number in fixed-point notation
g, G	displays a floating point number in either fixed-point form f or exponential form e based on the magnitude of the value
L	place before any floating point conversion specifier to indicate that a long double floating point number is displayed

FP Number Conversion Specifier - Reading

e, E f, F g, G	read a floating point number	double *
l, ll	place before any floating point conversion specifiers to indicate that a double or long double number is to be input	double * long double *