Using `printf()` to Print out Formatted Numbers

(exam included)

Motivation: A Simple Example

Print a multiplication table

```
1  int main( void )
2  {
3    int n , i ;
4    printf( "Enter an integer: " );
5    scanf( "%d" , &n );
6    for ( i = 1 ; i <= 10 ; ++i )
7      printf( "%d * %d = %d\n" , n , i , n*i );
8    return 0 ;
9  }
```

```
Input: 3
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27
3 * 10 = 30
```

Motivation: A Simple Example

Print a formatted multiplication table

```
1  int main( void )
2  {
3    int n , i ;
4    printf( "Enter an integer: " );
5    scanf( "%d" , &n );
6    for ( i = 1 ; i <= 10 ; ++i )
7      printf( "%2d * %2d = %2d\n" , n , i , n*i );
8    return 0 ;
9  }
```

```
Input: 3
 3 *  1 =  3
 3 *  2 =  6
 3 *  3 =  9
 3 *  4 = 12
 3 *  5 = 15
 3 *  6 = 18
 3 *  7 = 21
 3 *  8 = 24
 3 *  9 = 27
 3 * 10 = 30
```

- `printf()` supports many features for formatting the output.

- These slides show some of the useful features for formatting integers and floating point numbers.

- For a more complete reference, see
  - http://www.cplusplus.com/reference/cstdio/printf/
  - **Note**: even though the manual is for C++, most of the info there is applicable to C language.

Printing Integers

Here is a more general form of the format specifier for printing signed integers:

%[flags][width]d

where

[flags] can be any combination of the following:

+	Print the plus sign (+) for non-negative numbers.
-	Left-justify within the given field width; default is right-justification.

[width] – **Minimum number of characters** to be printed. If the value to be printed is shorter than this number, the result is padded with blank spaces.

Examples

```
printf( "%d" , 123 );
```

Output

1	2	3

By default, `printf()` uses as little width as it needs to print the value.

```
printf( "%5d" , 123 );
```

Output

␣	␣	1	2	3

When the minimum field width is set, the "unused" spaces are filled with space characters and the value is right-justified in the field.

```
printf( "%+d" , 123 );
```

Output

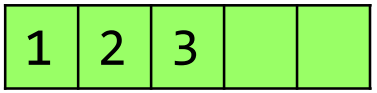
+	1	2	3

With the '+' flag, a plus sign is printed if the value to be printed is non-negative. Negative values are unaffected.

Examples

```
printf( "%-5d" , 123 );
```

Output

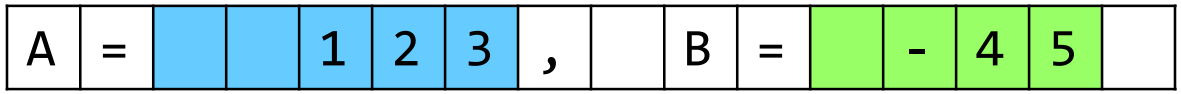


With the '-' flag, the value to be printed is left-justified in the field.

The following examples show combined used of flags and field width in the format specifiers.

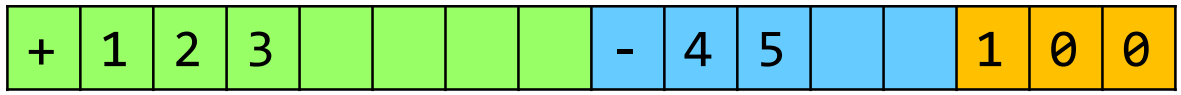
```
printf( "A=%5d, B=%4d" , 123 , -45 );
```

Output




```
printf( "%-+8d%-5d%-2d" , 123 , -45 , 100 );
```

Output



Note: For the last value, the specified minimum field width is not large enough, so it is ignored by `printf()`.

Printing Floating Point Numbers

Here is a more general form of the format specifier for printing a floating point number of type `double`:

```
%[flags][width][.precision]f
```

where

`[flags]` – Same as before

`[width]` – Same as before

`[.precision]` – Number of digits to be printed **after the decimal point (default is six).**

Examples

```
printf( "%f" , 1.234 );
```

Output

1	.	2	3	4	0	0	0

By default, `printf()` prints 6 digits after the decimal point.

```
printf( "%.2f" , 1.234 );
```

Output

1	.	2	3

Print only 2 digits after the decimal point.

```
printf( "%11.2f" , 1.0 );
```

Output

							1	.	0	0

1) Set minimum field width to 11, and
2) print 2 digits after the decimal point.

Note: The "unused" space in the field is filled with space characters.

Examples

```
printf( "%+11.0f" , 123.456 );
```

Output

1) Print a plus sign (+) if the value to be printed is positive,
2) set the minimum field width to 11, and
3) do not show decimal point and the digits after it.

Example: Formatting numbers in tabular form

```
1   int    i , j ;
2   double num ;
3
4   for ( i = 8 ; i < 12 ; i++ ) {
5       printf( "Row %2d:" , i );
6
7       for ( j = 1 ; j <= 5 ; j++ ) {
8           num = i * 8.5 + j * 5.2 ;
9           printf( "%10.2f" , num );
10      }
11
12      printf( "\n" );
13  }
```

This example shows how to neatly align numbers (in tabular form) in the output.

The numbers shown are just dummy values.

```
Row  8:      73.20     78.40     83.60     88.80     94.00
Row  9:      81.70     86.90     92.10     97.30    102.50
Row 10:      90.20     95.40    100.60    105.80    111.00
Row 11:      98.70    103.90    109.10    114.30    119.50
```