C Format Specifier

The Format specifier is a string used in the formatted input and output functions. The format string determines the format of the input and output. The format string always starts with a '%' character.

**The commonly used format specifiers in printf() function are:**

|  |  |
| --- | --- |
| **Format specifier** | **Description** |
| %d or %i | It is used to print the signed integer value where signed integer means that the variable can hold both positive and negative values. |
| %u | It is used to print the unsigned integer value where the unsigned integer means that the variable can hold only positive value. |
| %o | It is used to print the octal unsigned integer where octal integer value always starts with a 0 value. |
| %x | It is used to print the hexadecimal unsigned integer where the hexadecimal integer value always starts with a 0x value. In this, alphabetical characters are printed in small letters such as a, b, c, etc. |
| %X | It is used to print the hexadecimal unsigned integer, but %X prints the alphabetical characters in uppercase such as A, B, C, etc. |
| %f | It is used for printing the decimal floating-point values. By default, it prints the 6 values after '.'. |
| %e/%E | It is used for scientific notation. It is also known as Mantissa or Exponent. |
| %g | It is used to print the decimal floating-point values, and it uses the fixed precision, i.e., the value after the decimal in input would be exactly the same as the value in the output. |
| %p | It is used to print the address in a hexadecimal form. |
| %c | It is used to print the unsigned character. |
| %s | It is used to print the strings. |
| %ld | It is used to print the long-signed integer value. |

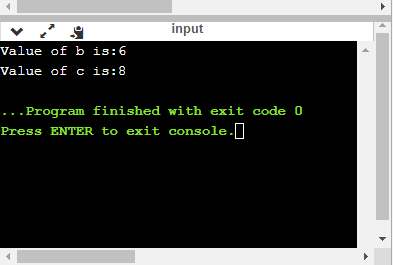
**Let's understand the format specifiers in detail through an example.**

* **%d**

1. **int** main()
2. {
3. **int** b=6;
4. **int** c=8;
5. printf("Value of b is:%d", b);
6. printf("\nValue of c is:%d",c);
8. **return** 0;
9. }

In the above code, we are printing the integer value of b and c by using the %d specifier.

**Output**

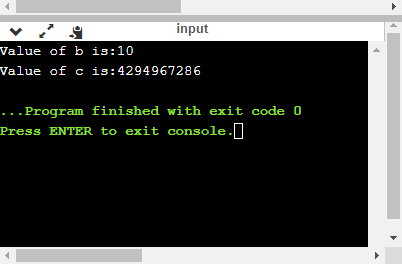


* **%u**

1. **int** main()
2. {
3. **int** b=10;
4. **int** c= -10;
5. printf("Value of b is:%u", b);
6. printf("\nValue of c is:%u",c);
8. **return** 0;
9. }

In the above program, we are displaying the value of b and c by using an unsigned format specifier, i.e., %u. The value of b is positive, so %u specifier prints the exact value of b, but it does not print the value of c as c contains the negative value.

**Output**

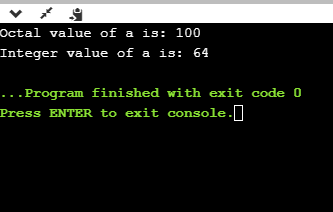


* **%o**

1. **int** main()
2. {
3. **int** a=0100;
4. printf("Octal value of a is: %o", a);
5. printf("\nInteger value of a is: %d",a);
6. **return** 0;
7. }

In the above code, we are displaying the octal value and integer value of a.

**Output**

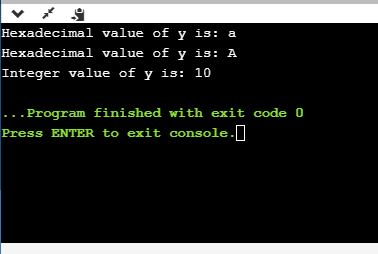


* **%x and %X**

1. **int** main()
2. {
3. **int** y=0xA;
4. printf("Hexadecimal value of y is: %x", y);
5. printf("\nHexadecimal value of y is: %X",y);
6. printf("\nInteger value of y is: %d",y);
7. **return** 0;
8. }

In the above code, y contains the hexadecimal value 'A'. We display the hexadecimal value of y in two formats. We use %x and %X to print the hexadecimal value where %x displays the value in small letters, i.e., 'a' and %X displays the value in a capital letter, i.e., 'A'.

**Output**

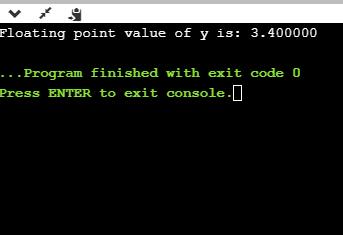


* **%f**

1. **int** main()
2. {
3. **float** y=3.4;
4. printf("Floating point value of y is: %f", y);
5. **return** 0;
6. }

The above code prints the floating value of y.

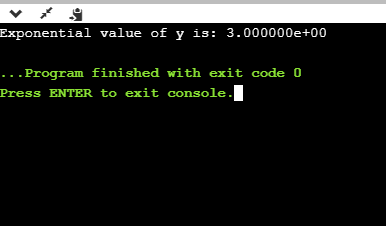
**Output**



* **%e**

1. **int** main()
2. {
3. **float** y=3;
4. printf("Exponential value of y is: %e", y);
5. **return** 0;
6. }

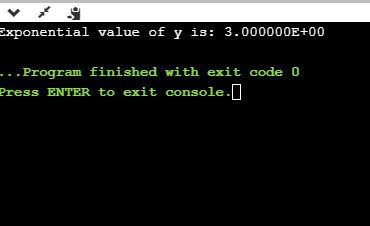
**Output**



* **%E**

1. **int** main()
2. {
3. **float** y=3;
4. printf("Exponential value of y is: %E", y);
5. **return** 0;
6. }

**Output**

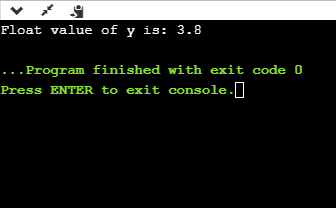


* **%g**

1. **int** main()
2. {
3. **float** y=3.8;
4. printf("Float value of y is: %g", y);
5. **return** 0;
6. }

In the above code, we are displaying the floating value of y by using %g specifier. The %g specifier displays the output same as the input with a same precision.

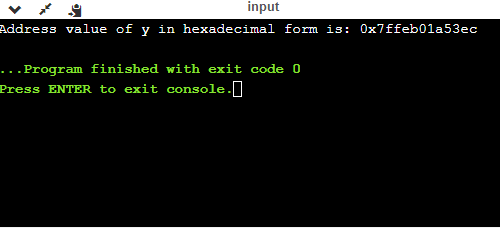
**Output**



* **%p**

1. **int** main()
2. {
3. **int** y=5;
4. printf("Address value of y in hexadecimal form is: %p", &y);
5. **return** 0;
6. }

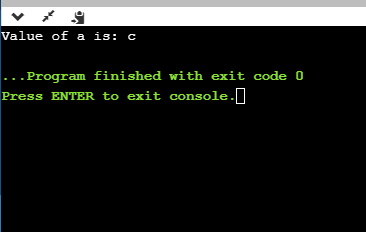
**Output**



* **%c**

1. **int** main()
2. {
3. **char** a='c';
4. printf("Value of a is: %c", a);
5. **return** 0;
6. }

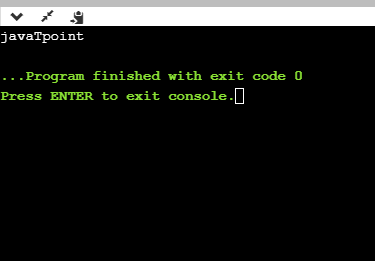
**Output**



* **%s**

1. **int** main()
2. {
3. printf("%s", "javaTpoint");
4. **return** 0;
5. }

**Output**



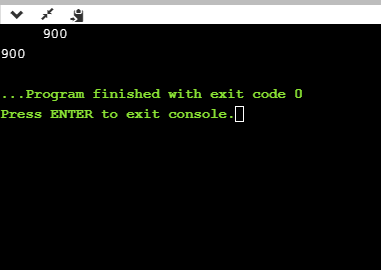
Minimum Field Width Specifier

Suppose we want to display an output that occupies a minimum number of spaces on the screen. You can achieve this by displaying an integer number after the percent sign of the format specifier.

1. **int** main()
2. {
3. **int** x=900;
4. printf("%8d", x);
5. printf("\n%-8d",x);
6. **return** 0;
7. }

In the above program, %8d specifier displays the value after 8 spaces while %-8d specifier will make a value left-aligned.

**Output**

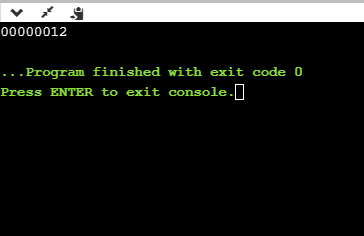


**Now we will see how to fill the empty spaces. It is shown in the below code:**

1. **int** main()
2. {
3. **int** x=12;
4. printf("%08d", x);
5. **return** 0;
6. }

In the above program, %08d means that the empty space is filled with zeroes.

**Output**



Specifying Precision

We can specify the precision by using '.' (Dot) operator which is followed by integer and format specifier.

1. **int** main()
2. {
3. **float** x=12.2;
4. printf("%.2f", x);
5. **return** 0;
6. }

**Output**

