Formatted Input and Output

Formatted input and output means that is entered and displayed in a particular format. Though format specification, better presentation of result can be obtained. Formats for different specifications are as-

Format For Integer Input

%wd

Here ‘d’ is the conversion specification character for integer value and ‘w’ is an integer number specifying the maximum field width of input data. If the length of input is more than this maximum field then the values are not stored correctly. For example-

```
scanf (“%2d%3d”, &a, &b);
```

i) When input data length is than the given field width, then the input values are unaltered and stored in given variables.

Input

6 39

Result

6 is stored in a and 39 is stored in b.

ii) When input data length is equal to the given field width, then the input values are unaltered stored in given variables.

Input-

26 394

Result-

26 is stored in a and 394 is stored in b.

iii) When input data is more than the given field width, then the input values are altered stored in the variable as-

Input-

269 3845

Result-

26 is stored in a and 9 is stored in b and the rest of input is ignored.

Format For Integer Output

%wd

Here w is the integer number specifying the minimum field width of the output data. If the length of the variable is less than the specified field width, then the variable is less than the specified field width, then the variable is right justified with leading blanks. For example-

```
printf(“a=%3d, b=%4d”, a, b )
```

1) When the length of variable is less than the width specifier

Value of variables-

78 9

Output:

a	=		7	8	,	b	=				9

The width specifier of first data is 3 while there are only 2 digits in it, so there is one leading blank.
The width specifier of second data is 4 while there is only 1 digit, so there are 3 leading blanks.

ii) When the length of the variable is equal to the width specifier.

Value of variables-

263 1941

Output:

a	=	2	6	3	,	b	=	1	9	4	1

.iii) When length of variable is more than the width specifier, then also the output is printed correctly.

Value of variables-

2691 19412

Output:

a	=	2	6	9	1	,	b	=	1	9	4	1	2

More Examples :-

printf("%d",9876);

9	**8**	**7**	**6**

printf("%6d",9876);

		9	**8**	**7**	**6**

printf("%2d",9876);

9	**8**	**7**	**6**

printf("%-6d",9876);

9	**8**	**7**	**6**		

printf("%-06d",9876);

9	**8**	**7**	**6**		

printf("%06d",9876);

0	**0**	**9**	**8**	**7**	**6**

printf("%6d",-9876);

	-	**9**	**8**	**7**	**6**

```
#include<stdio.h>
main( )
{
     int  a=4000, b = 200, c = 15;
     printf("a  =  %d  \nb   =  %d  \nc  =  %d\n", a,b,c);
     printf("%a  =  %4d  \n%b   =  %4d  \nc  =  %4d\n", a,b,c);
}
```

The Output of the first printf would be

```
    a=4000
    b=200
    c=15
```

while the output of second printf would be-

```
   a=4000
   b=200
   c=15
```

Format For Floating Point Numeric Input

%wf

Here 'w' is the integer number specifying the total width of the input data (including the digits before and after decimal and the decimal itself). For example-

```
scanf("%3f%4f",&x,&y);
```

i) When input data length is less than the given width, values are unaltered and stored in the variables.

Input

5 5.9

Result

5.0 is stored in x and 5.90 is stored in y.

ii) When input data length is equal to the given width, then the given values are unaltered and stored in the given variables.

Input

5.3 5.92

Result

5.3 is stored in x and 5.92 is stored in y.

iii) When input data length is more than the given width then given values are altered and stored in the given variables as-

Input

5.93 65.87

Result

5.9 is stored in x and 3.00 is stored in y.

Format For Floating Point Numeric Output

%w.nf

Here w is the integer number specifying the total width of the input data and n is the number of digits to be printed after decimal point. By default 6 digits are printed after the decimal. For example-

```
printf("x = %4.1f, y = %7.2f", x, y);
```

If the total length of the variable is less than the specified width 'w, then the value is right justified with leading blanks. If the number of digits is more than 'n' then the digits are rounded off.

Value of variables-

8   5.9

Output:

X	=		8	.	0	,	Y	=				5	.	9	0

Value of variables-

25.3   1635.92

Output:

X	=	2	5	.	3	,	Y	=	1	6	3	5	.	9	2

Value of variables-

15.231   65.875948

Output:

X	=	1	5	.	2	,	y	=			6	5	.	8	8

More Exapmles

float y=98.7654;

printf("%7.4f",y);

9	8	.	7	6	5	4

printf("%7.2f",y);

		9	8	.	7	7

printf("%-7.2f",y);

9	8	.	7	7		

printf("%f",y);

9	8	.	7	6	5	4

printf("%10.2e",y);

		9	.	8	8	E	+	0	1

printf("%11.4e",-y);

-	9	.	8	7	6	5	E	+	0	1

printf("%-10.2e",y);

9	.	8	8	E	+	0	1		

printf("%e",y);

9	.	8	7	6	5	4	0	E	+	0	1

Some system support the following format
printf("%*.*f",width,precision,number);

int width=7,precision=2,number=1234;
printf("%*.*f",width,precision,number);

is equivalent to
printf("%7.2f",number);

Format For String Input

% ws

Here w specifies the total number of characters that will be stored in the string.

```
    char str [8] ;
    scanf ("%3s", str );
```

If the input is-

    Srivastava

only first three characters of this input will be stored in the string will be-

    'S', 'r', 'I', '\0'

The null character ('\0') is automatically stored at the end.

Format For String Output

%w.ns

Here w is the specified field width. Decimal point and ‘n’ are optional. If present then ‘n’ specifies that only first n characters of the string will be displayed and (w – n) leading blanks are displayed before string.

i) printf(“%3s”, “sureshkumar”);

s	u	r	e	s	h	k	u	m	a	r

ii) printf(“%10s”, “reeta”);

					r	e	e	t	a

iii) printf(“%.3s”, “sureshkumar");

s	U	r

iv) printf(“% 8.3s”, “sureshkumar“);

					s	u	r

(8 – 3 = 5 leading blanks)

More Exapmles
“NEW DELHI 110001”

%s	**N**	**E**	**W**		**D**	**E**	**L**	**H**	**I**		**1**	**1**	**0**	**0**	**0**	**1**				
%20s					**N**	**E**	**W**		**D**	**E**	**L**	**H**	**I**		**1**	**1**	**0**	**0**	**0**	**1**
%20.10s											**N**	**E**	**W**		**D**	**E**	**L**	**H**	**I**	

%.5s	**N**	**E**	**W**		**D**

%-20.10s	**N**	**E**	**W**		**D**	**E**	**L**	**H**	**I**											

%5s	**N**	**E**	**W**		**D**	**E**	**L**	**H**	**I**		**1**	**1**	**0**	**0**	**0**	**1**					

Suppression Character in scanf( )

If we want to skip any input field then we specify * between the % sign and the conversion specification. The input field is read but its value is not assigned to any address. This character * is called the suppression character. For example-

```
    scanf (“%d %*d %d”, &a, &b, &c);
```

Input:

```
     25  30  35
```

Here 25 is stored in ‘a’, 30 is skipped and 35 is stored in the ‘b’. Since no data is available for ‘c’ so it has garbage value.

```
scanf(“%d %*c%d %*c %d”, &d, &m, &y);
```

Input:

```
     3/1/2003
```

Here 3 will be stored in d, then / will be skipped, 11 will be stored in m, again / will be skipped and finally 2003 will be stored in y.

```
#include<stdio.h>
main (  )
{
    int  a, b, c;
    printf(“Enter  three numbers   :”);
    scanf(“%d  %*d  %d”, &a, &b, &c);
    printf (“%d  %d  %d”, a, b, c)’
}
```

Output:

```
Enter three numbers  :  25  30  35

       25  35  25381
```

The variable c has garbage value.

Character I/O

getchar ( ) and putchar ( )

These macros getchar() and putchar() can be used for character I/O. getchar() reads a single character from the standard input. putchar() outputs one character at a time to the standard output.

```
#include<stdio.h>
main (  )
{
     char  ch;
     printf("Enter  a  character   :  ");
     ch=getchar () ;
     printf  ("The entered character is  :  ") ;
     putchar (ch) ;
}
```

Output:

```
Enter a character  : B
The entered character is  :  B
```

Exercise

Assume stdio.h is included in all programs.

```
1)  #define  MSSG  "Hello  World\n"
    main ( )
    {
          printf (MSSG)  ;
    }
2) main ( )
   {
       printf ("Indian\b is great\n") ;
       printf( "New\rDelhi\n") ;
   }

3) main ( )
   {
         int a=11;
         printf ("a  =  %d\t",a) ;
         printf ("a  =  %o\t",a) ;
         printf ("a  =  %x\n",a) ;
   }

4) main ( )
   {
         int  a=50000;
         unsigned  int  b=50000;
         printf ("a  =  %d,  b  =  %u\n", a,b) ;
   }



5) main ( )
  {
        char  ch;
        printf ("Enter a character:"):
        scanf("%c",&ch) ;
        printf ("%d\n",ch) ;
  }
```

```
6) main ( )
   {
        float  b=123.1265;
        printf (“%f\t”,b);
       printf (“%.2f\t”,b);
       printf (“%.3f\n”,b);
   }

7) main ( )
   {
         int  a=625, b=2394, c=12345;
         printf ( “%5d,  %5d,  %5d\n”, a, b, c) ;
         printf ( “%3d,  %4d,  %5d\n”, a, b, c) ;
   }

8) main ( )
   {
        int  a=98;
        char  ch= ‘c’;
        printf (“%c,  %d\n”, a, ch );
   }

9) main ( )
   {
        float  a1, b1, a2, b2, a3, b3;
        a1=2;
        b1=6.8;
        a2=4.2;
        b2=3.57;
        a3=9.82;
        b3=85.673;
        printf (“%3.1f, %4.2f\n”, a1, b1) ;
        printf (“%5.1f, %6.2f\n”, a2,b2) ;
       printf (“%7.1f, %8.2f\n”, a3, b3) ;
  }

10) main ( )
     {
          printf(“%10s\n”, “India”) ;
          printf(“%4s\n”, “India”) ;
          printf (“.2s\n”, “India”) ;
          printf (“%5.2s\n”, “India”) ;
      }
```

Answers

1) Hello World

2) India is great

Delhi

\b takes the cursor to the previous position of current line, \r takes the cursor to the beginning of current line, \n takes the cursor to the beginning of next line.

3) a = 11 a = 13 a = b

4) a = -15536, b=50000

The value 50000 is outside the range of int data type.

5) This program enters a character and prints its ASCII value.

6) 123.126503 123.13 123.127

7)
```
     625,   2394,  12345
   625,  2394,  12345
```

8) b, 99

9)
```
 2.0,6.80
   4.2,    3.57
       9.8,     85.67
```

10)
```
          India
   India
   In
       In
```