Formatted Input & Output

1) Format for Integer Input :-

Syntax

%wd.

Here 'd' is the conversion specification char for integer value and 'w' is an integer no. specifying the maximum field width of input data. If the length of input is more than this max. field width then the values are not stored correctly

→ scanf ("%2d %3d", &a, &b);

a) when input data length is less than or equal to given field width, the input values are unaltered and stored in given variables

Input :-

6 394

Output :-

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

b) when input length is more than the given field width, the input values are altered and stored in the variables as.

Input :-

269 3845

Output :-

96 is stored in a and 9 will stored in b and rest input is ignored.

2) Format for Integer Output

Syntax

%.wd

Here 'w' is the integer no. specifying the field width of the output data. If the length of the variable is less than the specified field width, the variable is right justified with leading blanks.

→ printf ("a = %.3d, b = %.4d", a, b);

a) when the length of variable is less than the width specifier

Input :-

78 9

output :-

a	=		7	8	,	b	=			9

b) when the length is equal

Input :-

263 1941

output :-

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

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

Input :-

2691 19412

Output :-

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

3) Format for floating point Numeric Input

Syntax :-

%.wf.

Here 'w' is total width (including the digits before and after decimal and the decimal itself)

→ scanf ("%.3f %.4f", &x, &y);

a) when the input data length is less than or equal to the given width, values are unaltered and stored in the variables

Input :-

5 5.92

Output :-

5.0 is stored in x and 5.92 in y.

b) when Input data length is more than the given width, the entered values are altered and store in the given variables

Input :-

5.93 65.875

Output :-

5.9 is stored in x and 3.00 in y.

4) Format for floating point Numeric output

Syntax:

%w.nf

Here w is integer no specifying the total width of the input data and n is the no. of digits to be printed after decimal points. By default 6 digits are printed after decimal.

→ 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 no. of digit after decimal is more than n, the digits are rounded off.

Input :-

8 5.9

Output :-

~~x = 8~~

x	=		8	.	0	,	y	=			5	.	9	0

Input :-

25.3 1635.92

Output :-

x	=	2	5	.	3	,	y	=	1	6	3	5	.	9	2

Input :-

15.231 65.875948

Output :-

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

5) Format for string Input

Syntax :-

%.ws

Here w is total no. of char that will be stored in the string

char str[8];

→ scanf ("%3s", str);

Input :-

Srivastava

Only first three char of this input will be stored in the string

'S', 'r', 'i', '\0'

The Null character '\0' is automatically stored at the end.

6) Format for string Output

Syntax :-

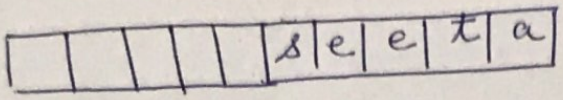
%.wns

Here w is width, Decimal point and n are optional. If present then n specifies that only first n char of the string will be displayed and (w-n) leading blanks are displayed before string

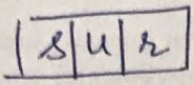
1) printf ("%.3s", "SureshKumar");

S	u	r	e	s	h	K	u	m	a	r

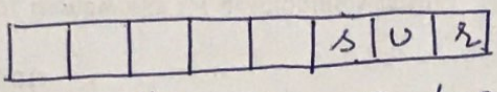
2) printf ("%10s", "seeta");



3) printf ("%.3s", "sureshkumar");



4) printf ("%8.3s", "sureshkumar");



(8-3 = 5 leading blanks)

Suppression character in scanf()

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

for example

scanf ("%d %*d %d", &a, &b, &c);

Input :-

25 30 35

25 is stored in a
30 is skipped
and 35 is stored in b
and in c garbage value is stored.