Tutorial Link https://codequotient.com/tutorials/Basic Input Output in C - Formatted IO/5a01b531cbb2fe34b77750c5

TUTORIAL

Basic Input Output in C - Formatted IO

Chapter

1. Basic Input Output in C - Formatted IO

Topics

1.2 printf()

1.8 scanf()

Unformatted IO supports functions those will work on characters or string data mainly. To input other kind of data we need formatted functions. Famous of these functions are scanf() and printf() for taking input and writing output respectively.

printf() and scanf() functions also use some escape sequences. These are special character constants used during printing output to screen. These are helpful in making the output more readable and more catchy. The following are the escape sequences defined: -

```
Code    Meaning
\b    Backspace       (delete the last character from
cursor)
\f    Form feed
\n    New line
\r    Carriage return    (same as \n except it remains in
same line)
\t    Horizontal tab      (prints a tab in output)
\"    Double quote      (prints double quotes)
\'    Single quote
\\    Backslash
\v    Vertical tab
\a    Alert
\?    Question mark
```

```
\N    Octal constant       (where N is an octal constant)
\xN    Hexadecimal constant (where N is a hexadecimal
constant)
```

These will be used in printf() function in next sections.

printf()

```c
#include<stdio.h>

int main()
{
  int a=10;
  char b='b';
  char str[100]="Hello and Welcome.";
  float f=10.4;
  double d=123456.9876;
  long int l=123456789;
  long double ld=28761872.67352;
  printf("int a=%d, char b=%c",a,b);
  printf("\nstr=%s",str);
  printf("\nfloat f=%f, double d=%lf",f,d);
  printf("\nlong int l=%ld, long double
ld=%Lf",l,ld);
  return 0;
}

```

It will print the following output: -

```
int a=10, char b=b
str=Hello and Welcome.
float f=10.400000, double d=123456.987600
long int l=123456789, long double ld=28761872.673520
```

We can also print the number in other base like octal or hexadecimal by using %o and %x or %X or %p for example: -

```c
#include<stdio.h>

int main()
{
  int a=10;
  int *p=&a;
  printf("a is in base 10=%d\n in base 8=%o\n in base 16=%x\n in base 16=%X\n Address of a in base 16=%p",a,a,a,a,p);
  return 0;
}
```

It will produce the following output: -

```
a is in base 10=10
 in base 8=12
 in base 16=a
 in base 16=A
 in base 16=0xbfd18208
```

There are also format modifiers for printf() like minimum field width, number of decimal places, alignment etc. these will be placed between % and format code. These are as follows: -

Minimum width specifier: - If we add a number between % sign and format code, it works like the minimum width of the output. It will then pads the output with spaces to make it happen. Although if the value is bigger than this number then it will printed as it is. For example “%5d”, %6f”

The precision specifier: - it will follow the minimum width specifier (if exists). It consists of a period followed by an integer. so “%10.4f” will display a floating number at least 10 characters wide with 4 decimal places.

Justification: -By default the justification in printf() is right alignment. We can change it by placing a ‘-‘ (minus) sign after % sign in format specifier for example “%-10.4f”.

Following program is using these modifiers: -

```c
#include<stdio.h>

int main()
{
  int a=10;
  float f=1087.46;
  printf("a=%d \n",a);
  printf("a with minimum width=%9d \n",a);
  printf("a with minimum width left=%-9d \n",a);
  printf("f=%f \n",f);
  printf("f with decimal fixed=%20.3f \n",f);
  printf("f with decimal fixed left aligned=%-20.3f \n",f);
  return 0;
}

```

It will produce the following output: -

```
a=10
a with minimum width=      10
a with minimum width left=10
f=1087.459961
f with decimal fixed=           1087.460
f with decimal fixed left aligned=1087.460
```

scanf()

It is used to read the data from keyboard in the same way as the printf() is used for all built-in types. The prototype of scanf() is: -

```
int scanf(const char *control_string,…);
```

similar to printf() the control_string will contain format specifiers. Which are defined below to read different kind of data types: -

```
Code       Meaning
%a       Reads a floating -point value (C99 only).
%c       Reads a single character.
%d       Reads a decimal integer.
%i       Reads an integer in either decimal, octal, or
hexadecimal format.
%e       Reads a floating -point number.
%f       Reads a floating -point number.
%g       Reads a floating -point number.
%o       Reads an octal number.
%s       Reads a string.
%x       Reads a hexadecimal number.
%p       Reads a pointer.
%n       Receives an integer value equal to the number of
characters read so far.
%u       Reads an unsigned decimal integer.
%[ ]       Scans for a set of characters.
%%       Reads a percent sign.
```

scanf() function will take the address of the variable in which the value needs to be saved so that the vlue entered from keyboard will be placed at that location in memory. So the variable name is passed preceded by a & (address of operator) sign.

The following program will show the use of printf() and scanf() functions: -

```
#include <stdio.h>

int main( )
{
 char str[100];
 int i;
 printf( "Enter a integer value :");
 scanf("%d", &i);
 printf( "Enter a String :");
 scanf("%s", str);   // as array name is the address
already so no need of & operator
 printf( "\nYou entered: %s %d ", str, i);
 return 0;
}
```