INPUT AND OUTPUT OF C PROGRAMMING

In most of the C programs the reading and writing of data will be done through the terminal, disk files or database files. In this chapter we cover how input and output is performed using the standard keyboard and terminal as the I/O devices.

C does not have any special statements for I/O operations. Instead standard functions like printf, scanf, getchar and putchar are used. Confining all I/O operations to use these standard functions makes a program portable.

I would like to re-iterate that usage of standard I/O functions (like getchar() or putchar()) require the header file *stdio.h*. This include file contains declarations and macro definitions associated with the Standard I/O library. *printf* and *scanf* do not require this header file.

Character Input and Output

getchar reads one character at a time from the standard input, which is generally the keyboard, unless it is re-directed by the operating system. The *getchar* function reads the character one at a time until it encounters the special character EOF, ie. End-of-file. The *getchar* function returns EOF if the end of file is reached. On UNIX systems a control-D

generates an EOF, whereas on a MS-DOS system it is control-Z followed by the RETURN key.

putchar is the counterpart of *getchar* which is used for displaying one character at a time onto the terminal or console. The following program displays the usage of both the functions.

Program 12.1

```
#include <stdio.h>
main ()
{
  int answer;
  while((answer = getchar())!= EOF)
    putchar(answer);
}
```

Formatted Input and Output

Formatted output statement is nothing but the most commonly used *printf* function. This function consists of a literal string or value of a variable which has to be displayed on the standard terminal using a format specifier which describes how it has to displayed. Examples of *printf* statements are:

```
printf("Hello \n");
printf("Your salary is \t%f\n", sal);
```

Some of the commonly used format specifiers are mentioned below :

%d int

%ld long int

%c single character

%s Null terminated strings

%f float or double

%e same as %f but exponential notation is used

%g use %f or %e

%x hexadecimal value (base 16)

%o Octal value (base 8)

scanf is used for reading formatted data from the keyboard. Similar to *printf* it requires a format specifier, followed by the list of items to be read. One important point to remember here is, *scanf* requires the address of the items to be read. So one needs to prefix *&* to a variable whose value is being scanned. Arrays and character strings are an exception to this, since the name of an array itself is the address of the array.

gets and puts

Complete lines of text can be read or written to the output terminal using *gets* and *puts* functions respectively.

gets reads a complete line of text into a string until a end-of-file (EOF) is encountered. It is the responsibility of the programmer to ensure that the string which receives the input text read by *gets* is large enough.

puts displays a string onto the standard output or terminal and follows it with a newline character.

Program 12.2

```
#include <stdio.h>

main ()
{
char answer[256];
puts("Enter your name");
while((gets(answer))!= NULL)
  printf("Hello " %s, answer);
}
```

Source : http://www.peoi.org/Courses/Coursesen/cprog/frame12.html