C - INPUT & OUTPUT

When we are saying **Input** that means to feed some data into program. This can be given in the form of file or from command line. C programming language provides a set of built-in functions to read given input and feed it to the program as per requirement.

When we are saying **Output** that means to display some data on screen, printer or in any file. C programming language provides a set of built-in functions to output the data on the computer screen as well as you can save that data in text or binary files.

The Standard Files

C programming language treats all the devices as files. So devices such as the display are addressed in the same way as files and following three file are automatically opened when a program executes to provide access to the keyboard and screen.

Standard File	File Pointer	Device
Standard input	stdin	Keyboard
Standard output	stdout	Screen
Standard error	stderr	Your screen

The file points are the means to access the file for reading and writing purpose. This section will explain you how to read values from the screen and how to print the result on the screen.

The getchar() & putchar() functions

The **int getchar(void)** function reads the next available character from the screen and returns it as an integer. This function reads only single character at a time. You can use this method in the loop in case you want to read more than one characters from the screen.

The **int putchar(int c)** function puts the passed character on the screen and returns the same character. This function puts only single character at a time. You can use this method in the loop in case you want to display more than one character on the screen. Check the following example:

```
#include <stdio.h>
int main( )
{
   int c;

   printf( "Enter a value :");
   c = getchar( );

   printf( "\nYou entered: ");
   putchar( c );

   return 0;
}
```

When the above code is compiled and executed, it waits for you to input some text when you enter a text and press enter then program proceeds and reads only a single character and displays it as follows:

```
$./a.out
Enter a value : this is test
You entered: t
```

The gets() & puts() functions

The **char *gets(char *s)** function reads a line from **stdin** into the buffer pointed to by **s** until either a terminating newline or EOF.

The **int puts(const char *s)** function writes the string s and a trailing newline to **stdout**.

```
#include <stdio.h>
int main( )
{
   char str[100];

   printf( "Enter a value :");
   gets( str );

   printf( "\nYou entered: ");
   puts( str );

   return 0;
}
```

When the above code is compiled and executed, it waits for you to input some text when you enter a text and press enter then program proceeds and reads the complete line till end and displays it as follows:

```
$./a.out
Enter a value : this is test
You entered: This is test
```

The scanf() and printf() functions

The **int scanf(const char *format, ...)** function reads input from the standard input stream **stdin** and scans that input according to **format** provided.

The **int printf(const char *format, ...)** function writes output to the standard output stream **stdout** and produces output according to a format provided.

The **format** can be a simple constant string, but you can specify %s, %d, %c, %f, etc., to print or read strings, integer, character or float respectively. There are many other formatting options available which can be used based on requirements. For a complete detail you can refer to a man page for these function. For now let us proceed with a simple example which makes things clear:

```
#include <stdio.h>
int main( )
{
   char str[100];
   int i;

   printf( "Enter a value :");
   scanf("%s %d", str, &i);

   printf( "\nYou entered: %s, %d ", str, i);

   return 0;
}
```

When the above code is compiled and executed, it waits for you to input some text when you enter a text and press enter then program proceeds and reads the input and displays it as follows:

```
$./a.out
Enter a value : seven 7
You entered: seven 7
```

Here, it should be noted that scanf() expect input in the same format as you provided %s and %d, which means you have to provide valid input like "string integer", if you provide "string string" or "integer integer" then it will be assumed as wrong input. Second, while reading a string scanf() stops reading as soon as it encounters a space so "this is test" are three strings for scanf().