Formatted and Unformatted Input and Output (I/O) in C

Gaurav Kr. suman

4/26/20

MAT07

Formatted and Unformatted Input and Output (I/O) in C

Input means to provide the program with some data to be used in the program and Output means to display data on screen or write the data to a printer or a file.

C programming language provides many built-in functions to read any given input and to display data on screen when there is a need to output the result.

C programming language has standard libraries that allow input and output in a program. The stdio.h or standard input output library in C that has methods for input and output.

Input output built-in functions in C falls into two categories, namely, formatted input output (I/O) functions and unformatted input output (I/O) functions.
printf() and scanf() are examples for formatted input and output functions and getch(), getche(), getchar(), gets(), puts(), putchar() etc. are examples of unformatted input output functions.

scanf() and printf() functions

The standard input-output header file, named stdio.h contains the definition of the functions printf() and scanf(), which are used to display output on screen and to take input from user respectively.

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.

```c
#include<stdio.h>

void main()
{
int a,b,c;
printf("Please enter any two integer numbers: \n");
scanf("%d %d", &a, &b);
c =  a + b;
printf("The addition of two number is: %d", c);
}
```

Output:

```
Please enter any two integer numbers:
15
3
The addition of two number is:18
```

When you will compile the above code, it will ask you to enter a value. When you will enter the value, it will display the value you have entered on screen.

You must be wondering what is the purpose of %d inside the scanf() or printf() functions. It is known as format string and this informs the scanf() function, what type of input to expect and in printf() it is used to give a heads up to the compiler, what type of output to expect.

Format String	Meaning
%d	Scan or print an integer as signed decimal number
%f	Scan or print a floating point number
%c	To scan or print a character
%s	To scan or print a character string. The scanning ends at whitespace.

Formatted Example:

```
#include<stdio.h>

Void main()
{
     printf("The color: %s\n", "blue");
     printf("First number: %d\n", 12345);
     printf("Second number: %04d\n", 25);
     printf("Third number: %i\n", 1234);
     printf("Float number: %3.2f\n", 3.14159);
     printf("Hexadecimal: %x\n", 255);
     printf("Octal: %o\n", 255);
     printf("Unsigned value: %u\n", 150);
     printf("Just print the percentage sign %%\n", 10);
}
```

Output:

```
The color: blue
First number: 12345
Second number: 0025
Third number: 1234
Float number: 3.14
Hexadecimal: ff
Octal: 377
Unsigned value: 150
Just print the percentage sign %
```

getchar() & putchar() functions

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

```
#include <stdio.h>

void main( )
{
    int c;
    printf("Enter a character");
    c = getchar();
    putchar(c);
}
```

displays the character passed to it on the screen and returns the same character. This function too displays only a single character at a time. In case you want to display more than one characters, use putchar() method in a loop.

gets() & puts() functions

The gets() function reads a line from stdin(standard input) into the buffer pointed to by str pointer, until either a terminating newline or EOF (end of file) occurs.
The puts() function writes the string str and a trailing newline to stdout.

str → This is the pointer to an array of chars where the C string is stored. (Ignore if you are not able to understand this now.)

```
#include<stdio.h>

void main()
{
    char str[100];
    printf("Enter a string");
    gets( str );
    puts( str );
    getch();
}
```

Difference between scanf() and gets()

The main difference between these two functions is that scanf() stops reading characters when it encounters a space, but gets() reads space as character too.

If you enter name as sanjay kumar using scanf() it will only read and store sanjay and will leave the part after space. But gets() function will read it completely.