BIT Notes

By Karna Bahadur Shrestha

Input/Output

Every C program performs three main functions i.e. it accepts data as input, processes data and produces output. Input refers to accepting of data while output refers to the presentation of data. Normally input is accepted form keyboard and output is displayed to screen. Input output operations are most common task in every programming language. C languages has different type of input output functions for input-output operations. Out of different input output operations, in this section, we mainly focus on formatted and unformatted input output functions.

The input output functions are classified into two types:

- Formatted functions
- Unformatted functions

Formatted input and output functions

Formatted input output functions accept or present the data in a particular format. The standard library consists of different functions that perform output and input operations. Out of these functions, printf() and scanf() allow user to format input output in desired format. printf() and scanf() can be used to read any type of data (integer, real number, character etc.).

Formatted Input

Formatted input refers to an input data that has been arranged in a particular format. *scanf()* is formatted input function which is used to read formatted data from standard input device and automatically converts numeric information to integers and floats. It is defined in *stdio.h*.

```
scanf(“format_string”, &var1, &var2, &var3, …, &varN);
```

The `”format string”` refers to the field format in which the data is to be entered, & symbol in above syntax is called address operator. The format string consists of multiple parts

For Examples :

```
int a;
```
`scanf("%d",&a);` reads the integer value and stores it to variable **a**.

And, similarly
```
int a;
float b;
```
`scanf("%d%f", &a, &b);` reads integer and real or floating number and stores to **a** and **b**.

The "`format string`" consists of multiple parts.

- Ordinary characters(optional)

  Ordinary characters are included in scanf() function to take input in a certain pattern.
- Field width(optional)

  Field width specifies the maximum number of characters to be read. Any characters that exceeds the specified field width will be skipped out.
- Conversion Character

  The conversion character is used on the basis of datatype of variable or data item to be input. The commonly used conversion characters are:

Data Type	Format Specifier/Conversion Character
int	%d
char	%c
float	%f
double	%lf
short int	%hd
unsigned int	%u
long int	%li
unsigned long int	%lu
signed char	%c
unsigned char	%c
long double	%Lf

```
/* program that illustrate the special character in format
string*/
#include<stdio.h>
void main()
{
     int day,month,year;
     printf("Enter the date in sequence(day/month/year):");
     scanf("%d/%d/%d",&day,&month,&year);
     printf("THe date you entered is %d/%d/%d",day,month,year);

}
```

```
/*program that illustrate the field width*/
#include<stdio.h>
void main()
{
     int amount;
     printf("Enter the amount:(only 3 digits)");
     scanf("%3d",&amount);
     printf("THe date you entered is : %d",amount);
}
```

Formatted Output

Formatted output functions are used to display or store data in a particular specified format. *printf()* is formatted output function which is used to display some information on standard output unit. It is defined in standard header file stdio.h. So, to *use printf()*, we must include header file stdio.h in our program.

Syntax:

```
printf("format_string", var1, var2, var3, …, varN);
```

Where format_string may contain :

- Characters that are simply printed as they are.
- Format specifier that begin with a % sign.
- Escape sequences that begin with \ sign.

The format string indicates how many arguments follow and what their types are. The arguments var1, var2, …, varN are the variables whose values are formatted and printed

according to format specifications of the format string. The arguments must match in n**umber, order and type** with the **format specifications**.

Format string may also consists of following things.

- Field width
- Precision

```
/*program that illustrate the uses of printf function */
#include<stdio.h>
void main()
{
     int n=1234;
     printf("n=%d\n",n);
     printf("n=%6d\n",n);
     printf("%.3f",13.4446);


}
```

Unformatted Input Output

Unformatted input output functions cannot control the format of reading and writing the data. These functions are the most basic form of input and output and they do not allow to supply input or display output in user desired format that's why we call them unformatted input output functions. Unformatted input output functions are classified into two categories as

- **character input output functions**
- **string input output functions**.

Character input functions are used to read a single character from the keyboard and character output functions are used to write a single character to a screen. **getch(), getche(), and getchar()** are unformatted character input functions while **putch() and putchar()** are unformatted character output functions.

getchar() is also a character input functions. It reads a character and accepts the input until carriage return is entered (i.e. until enter is pressed). It is defined in header file **stdio.h.**

Syntax:

```
character_variable = getchar();
```

```
#include<stdio.h>
void main()
{
  char ch;
  printf("Press any character: ");
  ch = getchar();
  printf("\nPressed character is: %c", ch);
}
```

The **putchar()** function is used for printing character to a screen at current cursor location. It is unformatted character output functions. It is defined in header file **stdio.h.**

Syntax

```
putchar(character);
      or
putchar(character_variable);
```

```
#include<stdio.h>
void main()
{
  char ch;
  printf("Press any character:");
  ch = getchar();
  printf("\nPressed character is:");
  putchar(ch);
}
```

getch() is character input functions. It is unformatted input function meaning it does not allow user to read input in their format. It reads a character from the keyboard but does not echo the pressed character and returns character pressed. It is defined in header file **conio.h.**

```
character_variable = getch();
```

Like **getch(), getche()** is also character input functions. It is unformatted input function meaning it does not allow user to read input in their format. Difference between **getch() and getche()** is that getche() echoes pressed character. **getche()** also returns character pressed like **getch()**. It is also defined in header file **conio.h.**

```
character_variable = getche();
```

Prepared By: Karna Bahadur Shrestha

The **putch()** function is used for printing character to a screen at current cursor location. It is unformatted character output functions. It is defined in header file **conio.h.**

```
putch(character);
      or
putch(character_variable);
```

In programming, collection of character is known as **string** or **character array**. In C programming, there are different input output functions available for reading and writing of strings. Most commonly used are gets() and puts(). gets() is unformatted input function for reading string and puts() is unformatted output function for writing string.

gets() is string input functions. It reads string from the keyboard. It reads string until enter is pressed. It is defined in header file **stdio.h.**

```
gets(string_variable);
```

```
#include<stdio.h>
void main()
{
  char name[100]
  printf("Enter your name:\n");
  gets(name);
  printf("Hello %s, Have a nice day.", name);

}
```

puts() is unformatted string output functions. It writes or prints string to screen. It is defined in standard header file stdio.h.

```
puts(string or string_variable);
```

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

  printf("Enter your name:\n");
  gets(name);
     puts("Hello");
     puts(name);
     puts("Have a nice day.");

}
```