Unit 3
Input and Output

Conversion specification in C

- The format specifier is used during input and output.
- It is a way to tell the compiler what type of data is in a variable during taking input using scanf() or printing using printf().
- Some examples are %c, %d, %f, etc.

/* sample program illustrating the use of input/output library functions */

```
#include <stdio.h>
int main ( )
{
        char c;                         /* declarations */
        float x;
        int i;
        c = getchar();                  /* character input */
        scanf ("%f", &x) ;              /* floating-point input */
        scanf ("%d", &i);               /* integer input */
        putchar(c);                     /* character output */
        printf("%03d %7.4f", i, x);     /* numerical output */
}
```

Output:

```
a
8
9
a      00000009  8.0000
```

Single Character Input: The getchar Function

- Single characters can be entered into the computer using the C library function getchar.
- The getchar function is a part of the standard C I/O library.
- It returns a single character from a standard input device (typically a keyboard).
- The function does not require any arguments, though a pair of empty parentheses must follow the word getchar.
- In general terms, a function reference would be written as

character variable = getchar ( ) ;

where character variable refers to some previously declared character variable.

Example:

```
char c;
.....
c = getchar();
```

Single Character Output: The putchar Function

- Single characters can be displayed (i.e., written out of the computer) using the C library function putchar.
- The putchar function, like getchar, is a part of the standard C I/O library.
- It transmits a single character to a standard output device.
- The character being transmitted will normally be represented as a character-type variable. It must be expressed as an argument to the function, enclosed in parentheses, following the word putchar.
- In general, a function reference would be written as:

    putchar (character variable);

    where character variable refers to some previously declared character variable.

Example:

```
char c;
.....
putchar(c);
```

Example:

```
/* read in a character and display it in uppercase */
#include <stdio.h>
#include <ctype.h>
main( )
{
        char letter;
        letter = getchar();
        putchar(toupper(letter));
}
```

Output:

```
a
A
```

Entering Input Data: The scanf Function

- Input data can be entered into the computer from a standard input device by means of the C library function scanf.
- This function can be used to enter any combination of numerical values, single characters and strings.

- In general terms, the scanf function is written as

  scanf(control string, arg1, arg2, . . . , argn)
- where control string refers to a string containing certain required formatting information, and arg1, arg2, . . . argn are arguments that represent the individual input data items.

Writing Output Data: The printf Function

- Output data can be written from the computer onto a standard output device using the library function printf.
- This function can be used to output any combination of numerical values, single characters and strings.
- It is similar to the input function scanf, except that its purpose is to display data rather than to enter it into the computer.
- That is, the printf function moves data from the computer's memory to the standard output device, whereas the scanf function enters data from the standard input device and stores it in the computer's memory.
- In general terms, the printf function is written as

  printf(control string, arg1, arg2, . . . , argn)
- where control stringrefers to a string that contains formatting information, and arg7, arg2,. . . ,argn are arguments that represent the individual output data items.

The gets And puts Function

- The gets and puts functions offer simple alternatives to the use of scanf and printf for reading and displaying strings.

Example:

```
/* read and write a line of text */
#include <stdio.h>
main( )
{
        char line[80];
        gets(line);
        puts(line);
}
```

Output:

```
Program to read and write a line.
Program to read and write a line.
```

Formatted and Unformatted Input/Output functions in C

Formatted I/O Functions

- Formatted I/O functions are used to take various inputs from the user and display multiple outputs to the user.
- These types of I/O functions can help to display the output to the user in different formats using the format specifiers.
- These I/O supports all data types like int, float, char, and many more.
- The following formatted I/O functions:
    - *printf()*
    - *scanf()*
    - *sprintf()*
    - *sscanf()*

Unformatted Input/Output functions

- Unformatted I/O functions are used only for character data type or character array/string and cannot be used for any other datatype.
- These functions are used to read single input from the user at the console and it allows to display the value at the console.
- The following unformatted I/O functions:
    - *getch()*
    - *getche()*
    - *getchar()*
    - *putchar()*
    - *gets()*
    - *puts()*
    - *putch()*

Exercise

1. Explain formatted I/O functions in detail. (5) [TU 2078]
2. What do you mean by unformatted I/O? Explain (5) [TU 2077]