LECTURE NOTE -6

INPUT-OUTPUT IN C

When we are saying **Input** that means we 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.

Functions *printf()* and *scanf()* are the most commonly used to display out and take input respectively. Let us consider an example:

```
#include <stdio.h>      //This is needed to run printf() function.
int main()
{
   printf("C Programming");  //displays the content inside quotation
return 0;
}
```

Output:

```
C Programming
```

Explanation:

- Every program starts from *main()* function.
- *printf()* is a library function to display output which only works if *#include<stdio.h>*is included at the beginning.
- Here, *stdio.h* is a header file (standard input output header file) and *#include* is command to paste the code from the header file when necessary. When compiler encounters *printf()*function and doesn't find *stdio.h* header file, compiler shows error.
- *return 0*; indicates the successful execution of the program.

Input- Output of integers in C

```
#include<stdio.h>
int main()
{
int c=5;
printf("Number=%d",c);
   return 0;
}
```

Output

```
Number=5
```

Inside quotation of *printf()* there, is a conversion format string "%d" (for integer). If this conversion format string matches with remaining argument, i.e, *c* in this case, value of *c* is displayed.

```
#include<stdio.h>
int main()
{  int c;
   printf("Enter a number\n");
   scanf("%d",&c);
   printf("Number=%d",c);
   return 0;
}
```

Output

```
Enter a number
4
Number=4
```

The *scanf()* function is used to take input from user. In this program, the user is asked an input and value is stored in variable *c*. Note the '&' sign before *c*. &c denotes the address of *c* and value is stored in that address.

Input- Output of floats in C

```
#include <stdio.h>
int main()
{
        float a;
        printf("Enter value: ");
        scanf("%f",&a);
        printf("Value=%f",a);    //%f is used for floats instead of %d
        return 0;
}
```

Output

```
Enter value: 23.45
Value=23.450000
```

Conversion format string "%f" is used for floats to take input and to display floating value of a variable.

Input - Output of characters and ASCII code

```
#include <stdio.h>
int main()
{
   char var1;
   printf("Enter character: ");
   scanf("%c",&var1);
   printf("You entered %c.",var1);
   return 0;
}
```

Output

```
Enter character: g
You entered g.
```

Conversion format string "%c" is used in case of characters.

ASCII code

When character is typed in the above program, the character itself is not recorded a numeric value (ASCII value) is stored. And when we displayed that value by using "%c", that character is displayed.

```
#include <stdio.h>
int main()
{
char var1;
printf("Enter character: ");
scanf("%c",&var1);
printf("You entered %c.\n",var1);
/* \n prints the next line(performs work of enter). */
printf("ASCII value of %d",var1);
return 0;
}
```

Output:

```
Enter character:
g
103
```

When, *'g'* is entered, ASCII value 103 is stored instead of *g*.

You can display character if you know ASCII code only. This is shown by following example.

```
#include <stdio.h>
int main()
{
int var1=69;
printf("Character of ASCII value 69: %c",var1);
return 0;
}
```

Output

```
Character of ASCII value 69: E
```

The ASCII value of 'A' is 65, 'B' is 66 and so on to 'Z' is 90. Similarly ASCII value of 'a' is 97, 'b' is 98 and so on to 'z' is 122.