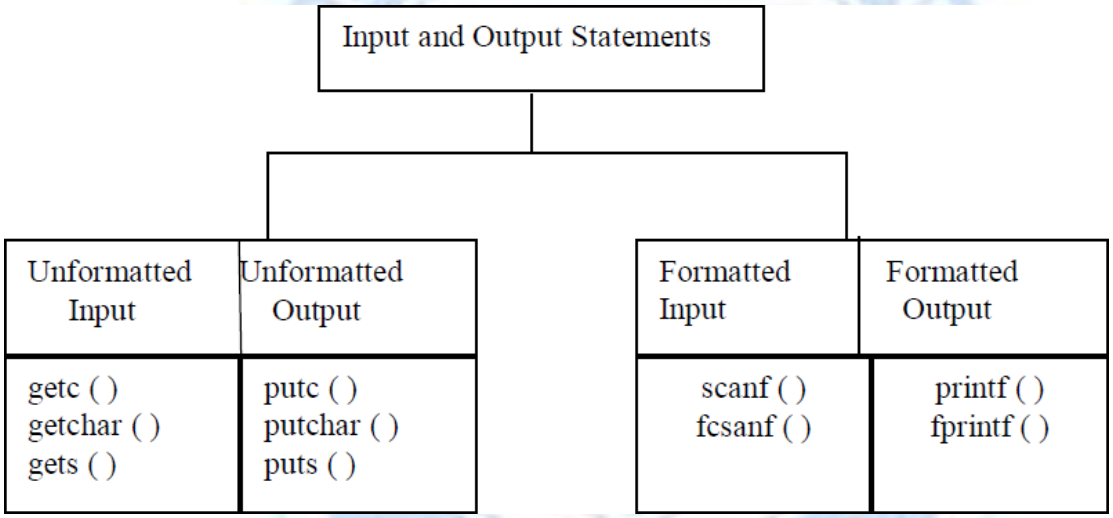
UNIT II - Managing simple Input and Output operations - Operators and Expressions - Decision Making: Branching statements, looping statements - Function: Declaration, Definition - Passing arguments by value - Recursion - Storage classes.

2.1 MANAGING SIMPLE INPUT AND OUTPUT OPERATIONS

Input and Output Statements

In „C" language, there are two types of input and output statements. They are:

- Unformatted I/O statements
- Formatted I/O statements



(I) Unformatted Input / Output Statements

In unformatted I/O statements, no need to specify the type & size of the data. It arranges the data in any format.

(a) getchar()

The getchar( ) is an input function that reads a single character from the standard input device (keyboard).

Syntax:

char variable;

variable = getchar ( );

Example:

char ch;

ch = getchar ( );

b) putchar()

The putchar( ) is an output function that writes a single character on the standard output device (monitor).

Syntax

putchar (variable);

Program :

```
# include<stdio.h>
#include<conio.h>
void main ( )
{
char ch;
printf (“Enter any one character : ”);
ch = getchar ( );
printf (“The character you typed is ”);
putchar(ch);
}
```

Output :

Enter any one character : S

The character you typed is S

c) gets()

This function is used to accept a string from standard input device until ENTER key is pressed.

Syntax

String_variable = gets();

d) puts()

This function is used to display a string to the standard output device.

Syntax

puts (String_variable)

Program :

```
# include<stdio.h>
```

```
#include<conio.h>
void main ( )
{
char name[20];
printf (“Enter your name : ”);
name = gets( );
printf (“Your name is : ”);
puts(name);
}
```

Output :

Enter your name : Anand

Your name is : Anand

e) getc()

The getc( ) is an input function that reads a single character from the standard input device (keyboard).

Syntax:

```
char variable;
variable = getc( );
```

Example:

```
char ch;
ch = getc( );
```

f) putc()

The putc( ) is an output function that writes a single character on the standard output device (monitor).

Syntax

```
putc(variable);
```

Program :

```
# include<stdio.h>
#include<conio.h>
```

```
void main ( )
{
char ch;
printf (“Enter any one character : ”);
ch = getc( );
printf (“The character you typed is ”);
putc(ch);
}
```

Output :

Enter any one character : S

The character you typed is S

(II) Formatted I/O Statements

In unformatted I/O statements, we need to specify the type & size of the data. It arranges the data in particular format.

(a) scanf ()

Input data can be read from standard input device (keyboard) using scanf ( ) function.

Syntax

scanf (“Control String”, &var1, &var2, . . .);

Example

scanf (“%d %d”, &a, &b);

Control String:

Control string specifies the type of data to be read and its size. The following list represents the possible control strings.

%c - To read single character

%s - To read a string

%d - To read an integer

%f - To read a floating point number

Rules for scanf()

- Each variable name must be preceded by symbol (&).
- The control string and variables data type should match each other.

(b) printf ( )

Output data can be displayed in the standard output device (monitor) using printf( ) function.

Syntax

printf (“Control String”, var1, var2, . . .);

Example

printf (“%d %d”, a, b);

printf (“Factorial = %d”, fact);

Rules for printf()

- The control string and variables data type should match each other.
- The variable must be separated by commas and need not be preceded with „&“ symbol.

Program:

```
# include<stdio.h>
#include<conio.h>
void main ( )
{
int A, B, C;
printf (“Enter values for A and B : ”);
scanf (“%d %d”, &A, &B);
C = A + B;
printf(“Sum is %d”, C);
getch();
}
```

Output:

Enter values of A and B : 4 3

Sum is 7

(c) fscanf

This function is used in file processing to read data from a file.

(d) fprintf

This function is used in file processing to write data into a file.

The fscanf() & fprintf() are similar to scanf and printf except that they are used in file processing.