Enumeration :- (enum) :- An enum is a user defined data type that consists of integer constants.

To define an enumeration keyword enum is used.

Syntax

enum tagname {value1, value2, value3 ....}

- enum is a keyword
- tagname is variable name
- values1, ... are create set of enum values.

→ enum week {sun, mon, tue, wed, thu, fri, sat};

By default sun is set to 0, mon is set to 1, and so on.
User can change default value of enum elements during declaration

```
enum week {
        sun = 12,
        mon = 55,
          :
        sat = 81
};
```

```
#
#
enum ABC{x, y, z};
void main()
{int a;sum;
a= x+y+z;
printf("sum %d", a);
}.
```

output : 3

```
#
# week
enum{sun, mon, tue, wed, thu, fri, sat};
void main()
{enum week today;
today = tue;
printf("Day %d", today+1);
}
```

output : Day 3.

File Handling in C :- File is a collection of bytes that is stored on secondary storage devices like disk. There are two type of file in a system.

1. Text file (.txt) — contains ASCII codes of digits, alphabet & symbol → text files are easily created using notepad or simple text editor.
2. Binary file (.bin) - contains collection of bytes (0 & 1). Binary files are compiled version of text files.

Why files are needed.

1. When a program is terminated, entire data is lost. Storing in a file will preserve your data even if the program terminates.
2. If you have entered large amount of data, it will take a lot of time to enter them all. However if you have a file containing all the data you can easily access the contents of the files using few commands in C.
3. You can easily move your data from one computer to another without any changes

In C language, we can use a structure pointer of file type to declare a file.

```
FILE *fp;
```

Syntax

```
FILE file_pointer
```

File Operations

1. Create a new file
2. Open an existing file
3. Close a file
4. Reading from and writing information to a file

Functions for file Handling :-

1. fopen() → Opens new or existing file
2. fclose() → Close the file
3. fprintf() → write data into the file
4. fscanf() → reads data from the file.
5. fputc() → writes a character into the file.
6. fgetc() → read a character from the file.

Modes of file :-

file Mode	Meaning of Mode
r	opens a textfile in reading mode
w	opens or create a textfile in writing mode
a	opens a text file in append mode
r+	opens a text file in both reading & writing mode
w+	"
a+	"
rb	open a binary file in reading mode
wb	open or create a binary file in writing mode
ab	open a binay file in append mode.

Opening a file or Creating a file :-

We must open a file before performing any operation.

```
FILE *fp
fp = fopen("file name", "mode");
```

Closing a file :-

fclose() function is used to close an already opened file.

```
fclose (FILE *fp);
```

```
#include <stdio.h>
#include <conio.h>
void main()
{ FILE *fp;
  char ch;
  fp = fopen ("First", "w")
  printf("Input data");
  while ((ch = getchar()) != EOF)
  { putc (ch,fp);
  }
  fclose (fp);
  }
```

```
#include <stdio.h>
#in      <conio.h>
{ FILE *fp
  char ch;
  fp=fopen ("first", "r");
  while (1)
  {ch = fgetc (fp);
   if (ch == EOF)
    break;
   printf("%c", ch);
  }.
fclose (fp);
}
```

```
File *fp
 char ch
fp=fopen("first","r");
while((ch=getc(fp))!=EOF)
 { printf("%c", ch);
  }
fclose (fp);
 }.
```

fprintf() :-

Syntax :-

fprintf(file pointer, Control string, variables);

Control string → format specifier (%c, %s, %d)

```
#include<stdio.h>
#include<conio.h>
void main()
{
 FILE *fp;
  int roll;
  char name[10];
 fp=fopen("student","w");
 printf("Enter Roll no");
 scanf("%d",&roll);
 printf("Enter Student Name");
 scanf("%s",name);
 fprintf(fp,"%d %s",roll,name);
 fclose(fp);
 fp=fopen("student","r");
fscanf(fp,"%d%s",&roll,name);
printf("Student Rollno %d",roll);
 printf("Student Name %s",name);
 fclose(fp);
}
```

fprintf & fscanf() :-

```
#include<stdio.h>
void main()
{ FILE *fp;
  fp=fopen("first.","w");
  fprintf(fp,"File by fprintf");//write data into file
    fclose(fp);
}
```

```
#include<stdio.h>
main()
{FILE *fp;
 char buff[20];//create character array to store data of file
 fp=fopen("first",r);
 while(fscanf(fp,"%s",buff)!=EOF)
 {
  printf("%s",buff);
 }
 fclose(fp);
}
```

fscanf(FILE *stream, const char *format)