1. **Define the purpose of File handling**
   - File handling is used to perform different operations on the files with the help of c program
   - It includes opening ,closing and reading ,writing data to files.
2. **what is text file**
   - A type of file that stores data as readable and printable characters is called text file.
   - A source program of C language is an example of text file
   - We can view , read and print the content of text files
3. **what is data file**
   - A data file is a collection of related records. Record contains data about an entity.
   - Data files are used to store any type of data
   - Files are stored on secondary storage.
4. **what is use of data files**
   - Data stored in a variable is temporary, so it is stored in data files to store it permanently
   - Data file can be used to provide input to program and store output of the program.
5. **what is stream**
   - A logical interface to a file is known as stream.
   - It is a flow of data between program and file
   - A stream is linked (منسلک) with a file using **open operation** and disassociated (الگ ہونا) using close operation
6. **what is two types of stream**
   - Two types of stream are : **text stream** and **binary stream**
7. **what is text stream**
   - A text stream is a sequence of characters
   - A certain character translation may occur in a text stream
   - For example : a new line is converted into carriage return
8. **what is binary stream**
   - A binary stream is a sequence of bytes
   - The translation is not performed in the binary stream,
   - It mean the number of bytes written or read are same
9. **Difference between text stream and binary stream**

Text Stream	Binary Stream
1. It is a sequence of characters.	**1.** It is a sequence of bytes.
2. A certain character translation may occur in a text stream.	**2.** No translation occurs in binary stream.
3. It can be used only for text data	**3.** It can be used for different types of data.

10. **What is sequential access method**
    - This method read and writes data in a sequence.
    - To read the last record, it is necessary to read all record stored before the last record.
    - This method is simple but slow.
11. **what is random access or direct access method**
    - This is fast method, as it access data from any location of the file
    - This method does not read and write data in a sequence.
12. **What is End of file (EOF) Marker? *OR* how the end of a text file is indicated.**
    - A constant EOF is a special character that represents **end of file character**.
    - It is used to indicate the end of any text file.
    - EOF is placed automatically after the last character in the file.
13. **what is new line character**
    - The **ENTER** key is used to move the cursor to the next line in any text editor
    - A newline character is placed at the end of each line when user press **ENTER** key
    - The new line is denoted by **\n** in C.
14. **what is string**
    - Collection of characters written in double quotation is called String.
    - data type for string variable is **char**
    - length of string is given in brackets []
    - **Example :** char str[10]; str=”HELLO”;

15. Define a Pointer with example

- Pointer is a type of variable that store memory addresses
- It normally stores the memory ***address of a variable or object.***
- Data type of a pointer variable must be same as data type of the variable whose memory address is stored in pointer.
    - **Example : int *ptr;**

16. what is File Pointer

- File pointer is a pointer that refers a to a file on the secondary storage
- File pointer is a variable of type FILE that is defined in a **stdio.h**
- It is used to access and manipulate a data file.
- One file pointer can be associated with only one data file.
    - **Example:** FILE *MyFile;

17. what is the Procedure to open a file in C *OR* write Protopyte of a fopen() function

- A function **fopen()** is used to open a file

Syntax

File_pointer = fopen(File_name,Mode);

Example

```
FILE *MyFile;
MyFile=fopne("book.txt","w");
```

18. How file is closed

- File is closed by using fclose() function
- **Syntax:** fclose(File_pointer);
- This function disassociate(الگ ہونا) the opened file from the program.

19. what are different opening modes of files

- different opening modes provided by C language are
    - read mode ,write mode ,append mode , read/write mode , append/read mode

20. what is mean by r+ and a+

- These are two modes for opening files
- r+ is used to open a file in read/write mode . the file must exist already
- a+ is used to open a file in append / read mode . the new data is written at the end of a file

21. write three ways of reading data form text file

- Data can be read from a text file one character at a time by using **fgetc()** function.
- Data can be read from file as string by using **fgets()** function.
- **fscanf()** function is used to read formatted data from file.

22. Write three ways to write data in text file.

- **fputc()** function is used to write one character to the file at one time
- **fputs()** function is used to write one string to a file at one time
- **fprintf()** is used to write formatted input to a file

23. write fopen() for reading

- fopen("book.txt","r");

24. write fopen() for writing

- fopen("book.txt","w");

25. write fopen() for appending

- fopen("book.txt","a");

26. write fopen() for overwriting

- fopen("book.txt","w+");

27. write fopen() for reading writing

- fopen("book.txt","r+");

28. write fopen() for reading and appending

- fopen("book.txt","a+");

29. what is meant by fgetc() function

- fgetc() is used to read one character at a time
- **syntax:** ch=fgetc(file_pointer);

30. which functions are used to read and write string to a file

- **fputs()** is used to write string in a file and **fgets()** function is used to read a string from file

31. which functions are used to read and write character to a file

- **fputc()** is used to write one character in a file and fgetc() function is used to read one character from file

32. why is it necessary to close a file in c

- A file needs to be closed after a read or write operation **to release the memory allocated by the program**. In C, a file is closed using the fclose() function. This returns 0 on success and EOF in the case of a failure.

Programs

```
1: character by character read/write          -> fgetc / fputc
2: as a string data read/write                 -> fgests / fputs
3: formatedd method                            -> fscanf / fprintf
```

1. Reading/writing data character by character in files

write a program that read text character by character from keyboard and write it to file.

```
#include<stdio.h>
#include<conio.h>
int main()
{
    FILE *out;
    char ch;
    out =fopen("d:\\test.txt","w");

    if(out==NULL)
    return 0;

    printf("type some text press enter key to finish \n");
    ch=getche();
    while(ch!='\r')
    {
        fputc(ch,out);
        ch=getche();
    }
    fclose(out);
}
```

****

```
/*
this program opens a file test.txt in reading mode
and read from file charachter by character using fgetc()
and print it on standard output .
*/
#include<stdio.h>
int main()
{
    FILE *in;
    char ch;
    in =fopen("d:\\test.txt","r");
    if(in==NULL)
    {
    printf("no such file exist");
    return 0;
    }
    while(!feof(in))
    {
        ch=fgetc(in);
        printf("%c",ch);
    }
    fclose(in);
}
```

```c
/*
this program opens a file test.txt in reading mode
and count all vowels in a file.
*/
#include<stdio.h>
int main()
{
    FILE *in;
    char ch;
    int v=0;
    in =fopen("d:\\test.txt","r");
    if(in==NULL)
    {
    printf("no such file exist");
    return 0;
    }
    while(!feof(in))
    {
        ch=fgetc(in);
        switch(ch)
        {
            case 'a':
            case 'e':
            case 'i':
            case 'o':
            case 'u':
            case 'A':
            case 'E':
            case 'I':
            case 'O':
            case 'U':
            v++;
        }
    }
    printf("There are %d Vowels ",v);
    fclose(in);
}
```

****

```c
/*
this program counts number of characters in a file
*/
#include<stdio.h>
int main()
{
FILE *in =fopen("bscs.txt","r");
int chr=0;
char c;
c=fgetc(in);
while (c!=EOF)
{
    c=fgetc(in);
    chr++;
}
printf("Total number of characters %d\n",chr);
fclose(in);
}
```

```c
/*
this program counts the Alphabets and digits
in a file
*/
#include<stdio.h>
int main()
{
FILE *in =fopen("bscs.txt","r");
int chr=0,dig=0;
char c;
c=fgetc(in);
while (c!=EOF)
{
    c=fgetc(in);
    if(c>='A' && c<='Z')
    chr++;
    if(c>='a' && c<='z')
    chr++;
    if(c>='0' && c<='9')
    dig++;
}
printf("number of characters %d\n",chr);
printf("number of digits %d\n",dig);
fclose(in);
}
```

****

```c
/*this program read data from test.txt file  character
by character and write them in upper case in upper.txt file
*/
#include<stdio.h>
#include<conio.h>
#include<ctype.h>
int main()
{
    FILE *out;
    FILE *in;
    char ch;
    in =fopen("d:\\test.txt","r");
    out =fopen("d:\\upper.txt","w");

    if(out==NULL || in == NULL)
    return 0;

    ch=fgetc(in);
    while(ch!=EOF)
    {
        fputc(toupper(ch),out);
        ch=fgetc(in);
    }
    printf("converted to upper");
    fclose(out);
    fclose(in);
}
```

****

```c
/*
Write a program that copy the  content of a text file Book1.txt
into new file Book2.txt character by character
*/
#include<stdio.h>
int main()
{
FILE *Book1 =fopen("Book1.txt","r");
FILE *Book2 =fopen("d:\\Book2.txt","w");
char c;
c=fgetc(Book1);
while (c!=EOF)
{
    fputc(c,Book2);
    c=fgetc(Book1);
}
fclose(Book1);
fclose(Book2);
}
```

2. Reading/writing data string by string in files

```c
/*
This program creates a file test1.txt and write to in it
as a string
*/
#include<stdio.h>
#include<conio.h>
int main()
{

    FILE *out;
    char ch[25];
    out =fopen("d:\\test1.txt","w");

    if(out==NULL)
    return 0;

        printf("Type some text, press ctrl z to finish \n");
        gets(ch);
        while (!feof(stdin))
        {
        fputs(ch,out);
        gets(ch);

        fputs("\n",out);
        }
    fclose(out);
}
```

****

```c
/*
This program read from a file one string at a time
and display it on standard output screen
*/
#include<stdio.h>
int main()
{
    FILE *in;
    char ch[25];
    in =fopen("d:\\test1.txt","r");

    if(in==NULL)
    return 0;

    while (fgets(ch,25,in)!= NULL)
        printf("%s",ch);
    fclose(in);
}
```

```c
// copies one file into another as a string
#include<stdio.h>
#include<conio.h>
int main()
{
    FILE *out,*in;
    char ch[25];

    in =fopen("d:\\test1.txt","r");
    out =fopen("d:\\test1-copy.txt","w");
    if(in ==NULL || out == NULL)
    return 0;

    while (fgets(ch,25,in)!=NULL)
    fputs(ch,out);

    fclose(out);
    fclose(in);
    printf("data copied");
}
```

****

3. Reading/writing data as Formatted I/O

Creating a Sequential-Access File

```
/*
This program inputs roll no ,name and marks of students
and store each record in a file until user press Ctrl-Z
*/
#include<stdio.h>

int main()
{

FILE *out;
int rno,marks;
char nm[20];
out =fopen("d:\\formatted.txt","w");
if(out==NULL)
return 0;

printf("enter roll no,Name and marks ::press ctrl z to finish::\n");
scanf("%d %s %d",&rno,nm,&marks);
while (!feof(stdin))
{
fprintf(out,"%d %s %d \n",rno,nm,marks);
printf("enter roll no,Name and marks ::press ctrl z to finish::\n");
scanf("%d %s %d",&rno,nm,&marks);
}
fclose(out);
printf("data store in d:\\formatted.txt");
}
```

****

```
/*
This program writes a series from 1 to 100 in a file
*/

#include<stdio.h>
int main()
{
FILE *out =fopen("d:\\series.txt","w");

for (int i=1;i<=100;i++)
fprintf(out,"%d\n",i);
fclose(out);
}
```

****

What is feof () in C?

- The feof() function **indicates whether the end-of-file flag is set for the given stream** .

Difference between feof () and EOF

- **EOF is a value to compare upon characters to check whether end of a stream has reached and feof is a function call to check a file pointer has reached end of that file.**

Reading and printing a sequential file

```c
/*
This program reads roll no ,name and marks of students
from a file and show on standard output screen
using fscanf()
*/
#include<stdio.h>

int main()
{
    FILE *in;
    int rno,marks;
    char nm[20];
    in =fopen("d:\\formatted.txt","r");

    if(in==NULL)
    return 0;
    printf ("RNO   Name      Marks\n");
    while (fscanf(in,"%d %s %d",&rno,nm,&marks)!=EOF)
        printf("%-5d %-6s %5d \n",rno,nm, marks);

    fclose(in);

}
```

Searching sequentially *from a file*

```c
/*
searching file sequently
This program searches the student recod from a file
and show on standard output screen
*/
#include<stdio.h>

int main()
{
    FILE *in;
    int rno,marks;
    char nm[20];
    int r;
    in =fopen("d:\\formatted.txt","r");

    if(in==NULL)
    return 0;
    printf("enter roll no to search , 0 to end ");
    scanf("%d",&r);
    while (r!=0)
    {
        printf ("RNO   Name      Marks\n");

        while (fscanf(in,"%d %s %d",&rno,nm,&marks)!=EOF)
        if(rno==r)
        printf("%-5d %-6s %5d \n",rno,nm, marks);

        rewind(in);
        printf("enter roll no to search , 0 to end ");
        scanf("%d",&r);
    }
    fclose(in);
}
```

Resetting the File Position Pointer

The statement

```
rewind( in );
```

Causes a program's **file position pointer** to be repositioned to the *beginning* of the file.

Random-Access Files

Individual records of a **random access file** are normally *fixed in length* and may be accessed directly (and thus quickly) without searching through other records. This makes random-access files appropriate for Airline reservation systems, banking systems, point-of-sale systems, and other kinds of **transaction-processing systems** that require rapid access to specific data.

fwrite() and fread() functions

Function **fwrite** transfers a specified number of bytes beginning at a specified location indicated by the file position pointer.
Function **fread** transfers a specified number of bytes from the location specified by the file position pointer .

Writing Data Randomly to a Random-Access File

```
/*
This program inputs roll no ,name and marks of students
and store each record in a file randomly
until user press  0
*/
#include<stdio.h>
struct student
{
int rno,marks;
char nm[20];
};

int main()
{
    struct student std;
FILE *out;

out =fopen("d:\\student.txt","wb");
if(out==NULL)
return 0;

printf("enter roll no ::press 0 to finish::\n");
scanf("%d",&std.rno);
while (std.rno!=0)
{
printf("Name and marks :\n");
scanf("%s %d",std.nm,&std.marks);
fseek(out,( std.rno - 1 ) * sizeof( struct student ), SEEK_SET );
fwrite( &std, sizeof( struct student ), 1, out );

printf("enter roll no ::press 0 to finish::\n");
scanf("%d",&std.rno);
}
fclose(out);
}
```

- above program uses the combination of **fseek** and **fwrite** to store data at specific locations in the file. Function fseek sets the file position pointer to a specific position in the file, then fwrite writes the data.
- byte location is calculated by :
  `fseek(out,( std.rno - 1 ) * sizeof( struct student ), SEEK_SET );`
- The value of this expression is called the **offset** or the **displacement**. as the roll no start from 1 but the byte positions in the file start with 0, so 1 is subtracted from the roll number when calculating the byte location of the record. Thus, for record 1, the file position pointer is set to byte 0 of the file.
- The symbolic constant **SEEK_SET** indicates that the file position pointer is positioned relative to the beginning of the file by the amount of the offset.
- SEEK_CUR indicates that the seek starts at the *current location* in the file; and SEEK_END indicates that the seek starts at the *end* of the file.

Reading Data from a Random-Access File

```
/*
This program read roll no ,name and marks of students
from a random access file
*/
#include<stdio.h>
struct student
{
int rno,marks;
char nm[20];
};

int main()
{
    struct student std;
FILE *in;
int result;
in =fopen("d:\\student.txt","rb");
if(in==NULL)
return 0;

printf ("RNO   Name      Marks\n");
while (!feof(in))
{
result = fread( &std, sizeof( struct student ), 1, in );
// display record
if ( result != 0 && std.rno != 0 )
    printf("%-5d %-6s %5d \n",std.rno,std.nm, std.marks);

}
fclose(in);

}
```

- Function **fread** reads a specified number of bytes from a file into memory. For example,

  `fread( &std, sizeof( struct student ), 1, in );`

- reads the number of bytes determined by **sizeof(struct student)** from the file referenced by **in** pointer, stores the data in **std** and returns the number of bytes read.
- if no record found **fread** return 0.