Introduction:

File represents storage medium for storing data or information. Streams refer to sequence of bytes. In Files we store data i.e. text or binary data permanently and use these data to read or write in the form of input output operations by transferring bytes of data. So we use the term File Streams/File handling. We use the header file <fstream>.

In C++, files are mainly dealt by using three classes fstream, ifstream, ofstream available in fstream header file.

- **ofstream:** This is used to create a file and write data on files
- **ifstream:** This is used to read data from files
- **fstream:** This is used to both read and write data from/to files

Operations in File Handling:

- Creating a file: open()
- Reading data: read()
- Writing new data: write()
- Closing a file: close()

How to achieve File Handling

For achieving file handling in C++ we need follow following steps

- Naming a file
- Opening a file
- Reading data from file
- Writing data into file
- Closing a file

File Opening mode

Mode	Meaning	Purpose
ios :: out	Write	Open the file for write only.
ios :: in	read	Open the file for read only.
ios :: app	Appending	Open the file for appending data to end-of-file.
ios :: ate	Appending	take us to the end of the file when it is opened.

Both ios :: app and ios :: ate take us to the end of the file when it is opened. The difference between the two parameters is that the ios :: app allows us to add data to the end of file only, while ios :: ate mode permits us to add data or to modify the existing data any where in the file.

The mode can combine two or more parameters using the bitwise OR operator (symbol |)

Example

```
fstream file;
file.Open("data . txt", ios :: out | ios :: in);
```

Creating/Opening a File

Example:

```
#include<iostream>
#include <fstream>

using namespace std;

int main()
{
    fstream st; // Step 1: Creating object of fstream class
    st.open("D:\studytonight.txt",ios::out);  // Step 2: Creating new file
    if(!st) // Step 3: Checking whether file exist
    {
        cout<<"File creation failed";
    }
    else
    {
        cout<<"New file created";
        st.close(); // Step 4: Closing file
    }
    return 0;
}
```

Writing to a File

Example:

```
#include<iostream>
#include <fstream>

using namespace std;

int main()
{
    fstream st; // Step 1: Creating object of fstream class
    st.open("D:\studytonight.txt",ios::out);  // Step 2: Creating new file
    if(!st) // Step 3: Checking whether file exist
    {
        cout<<"File creation failed";
    }
    else
    {
        cout<<"New file created";
        st<<"Hello";     // Step 4: Writing to file
        st.close(); // Step 5: Closing file
    }

    return 0;
}
```

Reading from a file

```
#include<iostream>
#include <fstream>

using namespace std;

int main()
{
    fstream st; // step 1: Creating object of fstream class
    st.open("D:\studytonight.txt",ios::in);    // Step 2: Creating new file
    st>> noskipws;
    if(!st) // Step 3: Checking whether file exist
    {
        cout<<"No such file";
    }
    else
    {
        char ch;
        while (!st.eof())
        {
            st >>ch;  // Step 4: Reading from file
            cout << ch;   // Message Read from file
        }
    }
    return 0;
}
```

Closing a file

```
#include<iostream>
#include <fstream>

using namespace std;

int main()
{
    fstream st; // step 1: Creating object of fstream class
    st.open("D:\studytonight.txt",ios::in);    // Step 2: Creating new file
    st.close();  //closing a file
    return 0;
}
```

Program to write and read from a file

Example:

```
#include <iostream>
#include <fstream>

using namespace std;

int main()
{
   fstream file; //object of fstream class

   //opening file "sample.txt" in out(write) mode
   file.open("D:\hero.txt",ios::out);

   if(!file)
   {
      cout<<"Error in creating file!!!"<<endl;
      return 0;
   }

   cout<<"File created successfully."<<endl;
   //write text into file
   file<<"Hello i am a good person.";
   //closing the file
   file.close();

   //again open file in read mode
   file.open("D:\hero.txt",ios::in);
   file>>noskipws;
   if(!file)
   {
      cout<<"Error in opening file!!!"<<endl;
      return 0;
   }

   //read untill end of file is not found.
   char ch; //to read single character
   cout<<"File content: ";

   while(!file.eof())
   {
      file>>ch; //read single character from file
      cout<<ch;
   }

   file.close(); //close file

   return 0;
}
```

Manipulation of file pointer

The read operation from a file involves **get** pointer. It points to a specific location in the file and reading starts from that location. Then, the **get** pointer keeps moving forward which lets us read the entire file.

Similarly, we can start writing to a location where **put** pointer is currently pointing.
The **get** and **put** are known as file position pointers and these pointers can be manipulated or repositioned to allow random access of the file. The functions which manipulate file pointers are as follows :

Function	Description
seekg( )	Moves the **get** pointer to a specific location in the file
seekp( )	Moves the **put** pointer to a specific location in the file
tellg( )	Returns the position of **get** pointer
tellp( )	Returns the position of **put** pointer

Example:

```cpp
#include<iostream>
#include<fstream>
using namespace std;

int main() {
   fstream fp;
   char buf[100];
   int pos;

   // open a file in write mode with 'ate' flag
   fp.open("random.txt", ios :: out | ios :: ate);

   fp << "This is a line"; // write a line to a file
   fp << "This is a another line"; // write another file
   pos = fp.tellp();
   cout << "Current position of put pointer : " << pos << endl;
   // move the pointer 10 bytes backward from current position
   fp.seekp(-10, ios :: cur);
   fp<<" How are you";
   // move the pointer 7 bytes forward from beginning of the file
   fp.seekp(7, ios :: beg);
   fp << " Hello World  ";
   fp.close(); // file write complete
   cout << "Writing Complete ... " << endl;

   // open a file in read mode with 'ate' flag
   fp.open("random.txt", ios :: in | ios :: ate);
   cout << "\nReading from the file ... " << endl;
   fp.seekg(0); // move the get pointer to the beginning of the file
   // read all contents till the end of file
   while (!fp.eof()) {
      fp.getline(buf, 100);
      cout << buf << endl;
   }
   pos = fp.tellg();
   cout << "\nCurrent Position of get pointer : " << pos << endl;
   return 0;
}
```

put() and get() function in file handling

- The put() function is member of ofstream class. It is used to write single character into file.
- The get() function is member of ifstream class. It is used to read a character from the file.

Example:

```
#include<fstream>
#include<iostream>
using namespace std;
int main()
{
    fstream file;
    char ch;
    file.open("D:\demo.txt",ios::out);          //Statement 1
    cout<<"Enter a character to write in file:"<<endl;
    cin>>ch;
    file.put(ch);
    cout<<"Data written successfully..."<<endl;
    file.close();

    file.open("D:\demo.txt");
    if(file)
    {
        cout<<"Opening file"<<endl;
        file.get(ch);
        cout<<ch;
    }
    else
    {
        cout<<"No such file exist!"<<endl;
    }

    file.close();

    return 0;
}
```

read() and write() function in file handling

The write() function is used to write object or record (sequence of bytes) to the file. A record may be an array, structure or class.

Syntax of write() function

```
fstream fout;
fout.write( (char *) &obj, sizeof(obj) );
```

The write() function takes two arguments.
&obj : Initial byte of an object stored in memory.
sizeof(obj) : size of object represents the total number of bytes to be written from initial byte.
The read() function is used to read object (sequence of bytes) to the file.

Syntax of read() function

```
fstream fin;
fin.read( (char *) &obj, sizeof(obj) );
```

The read() function takes two arguments.

&obj : Initial byte of an object stored in file.

sizeof(obj) : size of object represents the total number of bytes to be read from initial byte.

The read() function returns NULL if no data read.

Example:

```cpp
#include <iostream>
#include <fstream>

using namespace std;

class student
{
    private:
        char name[30];
        int age;
    public:
        void getData(void)
        { cout<<"Enter name:"; cin.getline(name,30);
          cout<<"Enter age:"; cin>>age;
        }

        void showData(void)
        {
        cout<<"Name:"<<name<<",Age:"<<age<<endl;
        }
};

int main()
{
    student s;

    fstream file;

    //open file in write mode
    file.open("aaa.txt",ios::out);

    if(!file)
    {
      cout<<"Error in creating file.."<<endl;
      return 0;
    }
    cout<<"\nFile created successfully."<<endl;

    //write into file
    s.getData();    //read from user
    file.write((char*)&s,sizeof(s));    //write into file

    file.close();   //close the file
    cout<<"\nFile saved and closed succesfully."<<endl;


    //again open file in read mode
    file.open("aaa.txt",ios::in);
    if(!file){
        cout<<"Error in opening file..";
        return 0;
    }
    //read data from file
    file.read((char*)&s,sizeof(s));

    //display data on monitor
    s.showData();
    //close the file
    file.close();

    return 0;
}
```