Chapter # 9

FILE HANDLING

- Some time we need large data to process.
- It will be more convenient to process those data if stored on secondary memory such as hard disk.
- Once the data stored in hard disk, it can be use as many time as needed without being reentered from the keyboard.
- Also, data can be processed by a program and the output produced by the program can be stored on secondary memory which will be used later on.
- C language provides the facilities for the input and output the data to a disk.

File

- A file is a collection of data or bytes stored on the disk that is given a name.
- The bytes of the file is interpreted in different ways for example:
  - A text document interpret the bytes as characters, words, lines, paragraph or pages.
  - A graphical image may interpret the bytes as pixels.
  - A database interpret the bytes as fields and records.
- The information regarding data structure of file and operations used to process the file is attached with each file.
- **File Handling:**
  - The process of performing different operations on files is called file handling. It involves the following operations:
    - Opening file
    - Reading / Writing file
    - Closing file.

Types Of Files

1. **Text File:**

- A type of file that stores data as readable and printable characters is called text file.
- The user can view, read and print the text file.
- It is a stream of characters that can be processed by a computer sequentially in forward direction. That is why a text file is typically opened only for one type of operation at a time such as reading or writing etc.
- Only one character can be read or written to the text files at a time:

Types of Files

2. Binary Files:

- A type of file that stores data as non-readable binary code is called **binary file**.
- The user cannot read binary file. It can only be read and processed by computer.
- The binary file can be processed using sequential or random access methods.
- It can be opened for read and write operations at the same time.
- For a example, a database file is created and processed as binary file. The process of updating a record will be performed as follows:
  - Locating the appropriate record.
  - Reading the record into memory.
  - Modifying the record as required.
  - Writing the updated record back to the file

Stream

- In C++, a sequence of bytes/data associated with a file is called a stream.
- It consist of data that is exchanged between a file and the program.
- Two main types of stream in C++ are as follows:

1. **Input Stream:**
   - It is used to input data which take sequence of bytes/data from an input device such as keyboard or file and enters into a program.
2. **Output Stream:**
   - It is used to output data which sends sequence of bytes/data from file to an output device such as monitor, file or printer.

<fstream.h> Header file

- It is a header file that contains classes and functions which are used in file handling.
- To perform file processing or file handling in C++, <fstream.h> header file must be include in C++ source program.
- It consists of the following data types or classes:

1. **ifstream:**
   - It is only used for input which creates the file and reads information from files and enters into the program.
   - It consists of get() function , getline() function etc. which are used to read the contents of file.
2. **ofstream:**
   - It is only used for output which creates a file and writes information into a file.
   - It consists of put() function, write() function etc. which are used to write information into a file.
3. **fstream:**
   - It has the abilities of both ifstream and ofstream classes which means it can create files, write information to files and read information from files.

Opening a File

- A file can be opened by first creating an object of **ifstream**, **ofstream** or **fstream** class that is known as steam object which will be associated with a file.
- Any input or output or both operations performed on this object is applied to the file associated with it.
- The member function "**open()"** of stream object is used to open a file which has the following syntax:

```
object_name.open( file name , mode );
```

Modes of opening a File:

- A file can be opened in different modes using open() function which indicates the type of operations to be performed on the file after opening.
- Following are the different modes:

Mode	Description
ios::in	It is used to open a file for input operations
ios:out	It is used to open a file for output operations
ios::binary	It is used to open a file in binary mode.
ios::ate	It is used to open a file for output operations and move the control to the end of the file.
ios::app	Append mode. All output to that file to be appended to the end.
ios::trunc	If the file already exists, its contents will be truncated or deleted before opening the file.

- For a example to open a file:

  fstream obj;

  obj.open(“abc.txt”, ios::out);

- A file can be open in multiple mode using OR operator i.e.:

  fstream test;

  test.open(“c:\\xyz.txt”, ios::out | ios::in);

Default opening Mode:

- If mode parameter is not used by the user in open() member function, then the function uses default mode to open a file.
- The default modes are given below:

Class	Default mode parameter
ofstream	ios::out
Ifstream	ios::in
fstream	ios::in | ios::out

Closing a File

- The opened file should be close when the input or output operations on the file are finished.
- The member function **close()** of stream object is used to close a file which has the following syntax:

  object_name.close();

- For example:

  test.close();

  obj.close();

bof() function

- The word **bof** stands for Beginning of File.
- **bof()** function returns the value of **bof** pointer.
- **bof()** function is used to find if the control is at the beginning of the file or not.
- It returns **TRUE** if the control is at the beginning and returns **FALSE** otherwise.
- The syntax of using bof() function is as follows:

  object_name.bof();

eof() Function:

- The word **eof** stands for **End of File**.
- The **eof()** function returns the value of **eof** pointer.
- The **eof()** function is used to find if the controls has reached the end of file or not.
- It returns **TRUE** if the control has reached the end of file and returns **FALSE** otherwise.
- The syntax of using **eof()** function is as follows:

  object_name.eof();

Verifying File opening

- The stream object returns TRUE if the referred file is opened and returns FALSE if file not opened. For example:

```
fstream obj;
obj.open("test.txt", ios::in | ios::out );
if(obj)
        cout<<"file is opened";
else
        cout<<"file is not opened";
```

Example 1

- Write a program that creates a text file and writes a line of text in it.

```
#include <iostream.h>
#include <conio.h>
#include <fstream.h>
void main()
{
    clrscr();
    ofstream myfile;
    myfile.open("d:\\new\\example1.txt");
    myfile<<"Hello World";
    myfile.close;
    getch();
}
```

- The insertion operator << is used with stream object to store data in a file. It is used in the same way as it is used with **cout** to display output on screen.

Example 2:

- Write a program that inputs the name of five cities and stores them in a file **“city.txt”**

```
#include <iostream.h>
#include <conio.h>
#include <fstream.h>
void main()
{
    clrscr();
    char city[20];
    ofstream file;
    file.open("d:\\new\\city.txt");
    for( int i=1; i<=5; i++)
    {
        cout<<"Enter the city name: ";
        cin>>city;
        file<<city<<endl;
    }
    file.close;
    getch();
}
```

Reading from a file:

- The **extractor operator >>** is used to read from a file.
- It can read one word from the file at a time and store it in a variable or string.
- The extractor operator is need to be used repeatedly to read all the contents in a file.
- For example:

```
char ch[20];
ifstream obj;
obj.open("abc.txt");
obj>>ch;
cout<<ch;
```

Example 3:

- Write a program that display all the records in the file **city.txt**

```
#include <iostream.h>
#include <conio.h>
#include <fstream.h>
void main()
{
    clrscr();
    char city[20];
    ifstream file;
    file.open("d:\\new\\city.txt");

    while(!file.eof())
    {
        file>>city;
        cout<<city<<endl;
    }

    file.close;
    getch();
}
```

Reading / Writing files character by character

- The data can be read from or written to a file character by character.
- **get()** function is used to read single character from a file. Syntax of using this function is as under:

object_name.get(ch);

- put() function is used to write single character in a file. Syntax of using this function is as under:

object_name.put(ch);

Example:

- Write a program that inputs five characters from the user and stores them in a file.

```
#include <iostream.h>
#include <conio.h>
#include <fstream.h>
void main()
{
    clrscr();
    char ch;
    ofstream file;
    file.open("d:\\new\\write.txt");
    for( int i=1; i<=5; i++)
    {
        cout<<"Enter a character: ";
        cin>>ch;
        file.put(ch);
    }
    file.close;
    getch();
}
```

Example:

- Write a program that reads the characters from a text file and display them on screen.

```
#include <iostream.h>
#include <conio.h>
#include <fstream.h>
void main()
{
    clrscr();
    char ch;
    ifstream file;
    file.open("d:\\new\\write.txt");
    while(!file.eof())
    {
        file.get(ch);
        cout<<ch<<endl;
    }
    file.close;
    getch();
}
```