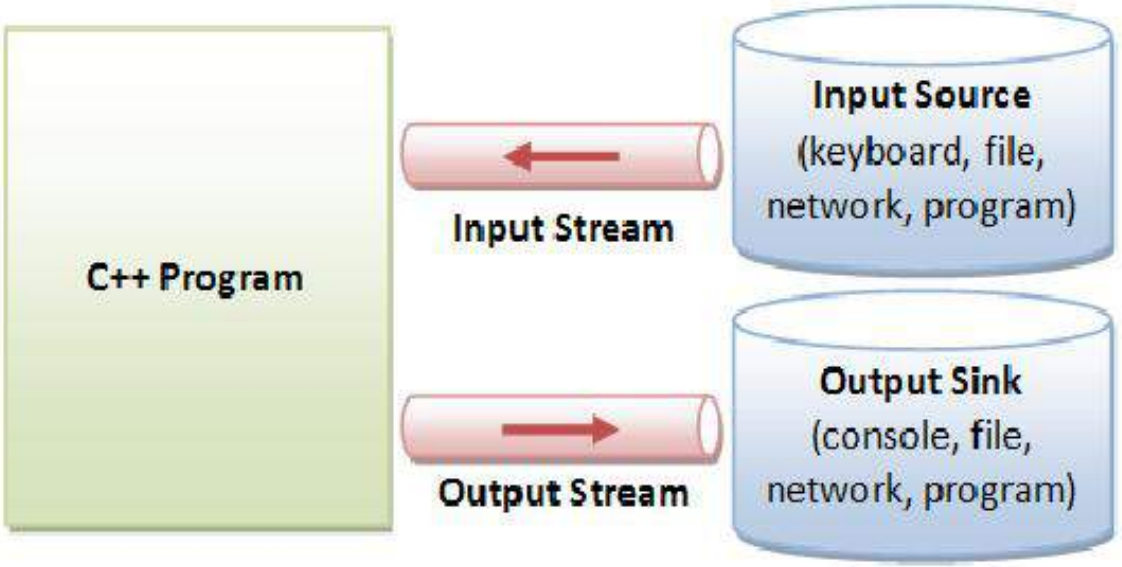
CHAPTER 12

DATA FILE HANDLING

OBJECTIVES

- **To understand the concepts of files.**
- **Usage of types of files.**
- **The role of text and binary files.**
- **Concept of opening and closing of files**
- **Concept of input and output operations in text files.**
- **Concept of input and output operations in binary files.**
- **Concept of file pointers and their manipulations.**



Internal Data Formats:

- Text: char, wchar_t
- int, float, double, etc.

External Data Formats:

- Text in various encodings (US-ASCII, ISO-8859-1, UCS-2, UTF-8, UTF-16, UTF-16BE, UTF16-LE, etc.)
- Binary (raw bytes)

12.1 Introduction:

We have already used the cin and cout for handling the input and ouput operations. They are used to accept inputs through keyboard and display outputs on the screen. C++ provides a rich set of operations for both unformatted and formatted I/O operations. In C++, these IO operations are implemented through iostream library.

The C++ standard libraries provide an extensive set of input/output capabilities which we will see in subsequent topics. This chapter will discuss very basic and most common I/O operations required for C++ programming.

C++ I/O occurs in streams, which are sequences of bytes. If bytes flow from device like a keyboard, a disk drive, or a network connection etc. to main memory, this is called **input operation** and if bytes flow from main memory to a device like a display screen, a printer, a disk drive, or a network connection etc., this is called **output operation**.

Most computer programs work with files. This is because files help in storing information permanently. Word processors create document files. Database programs create files of information. Compilers read source files and generate executable files. So, we see, it is the files that are mostly worked with, inside programs. A file itself is a bunch of bytes stored on some storage device like tape or magnetic disk etc.

In C++, file input/output facilities are implemented through a header file of C++ standard library. This header file is fstream.h.

In C++, a file, at its lowest level, is interpreted simply as a sequence, or stream of bytes. In C++, file I/O library manages the transfer of these bytes. At this level, the notion of a data type is absent.

On the other hand, file, at user level, consists of a sequence of possibly intermixed data types-characters, arithmetic values and class objects.

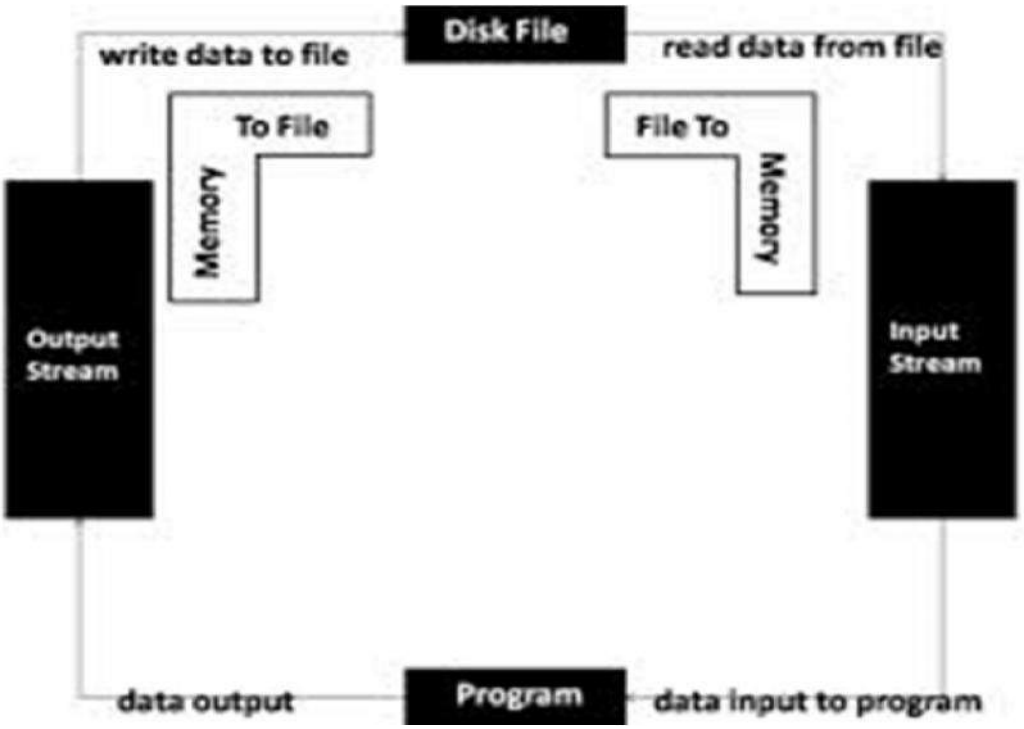
The fstream library predefines a set of operations for handling files related input and output. It defines certain classes that help to perform file input and output. For example, ifstream class ties a file to the program for input. ofstream class ties a file to the program for output and fstream classifies a file to the program for both input and output. File manipulation and related operations using streams are the topics we are going to discuss in this chapter.

12.2 fstream.h header file

The C++ input/output operations are very much similar to the console input and output operations. The file operations also make use of streams as an interface between the programs and the files.

A stream is a general name given to a flow of data at the lowest level; data is just the binary data without any notion of data type. Different streams are used to represent different kinds of data flow such as whether data is flowing into the memory or out of the memory. Each stream is associated with a particular class, which contains member functions and definitions for dealing with that particular kind of data flow. For example, the ifstream class represents input disk files.

The stream that supplies data to the program is known as **input stream**. It reads the data from the file and hand it over to the program. The stream that receives data from the program is known as **output stream**. It writes the received data to the file. Following figure shows it.



The information / data stored under a specific name on a storage device, is called a file.

Stream refers to a sequence of bytes

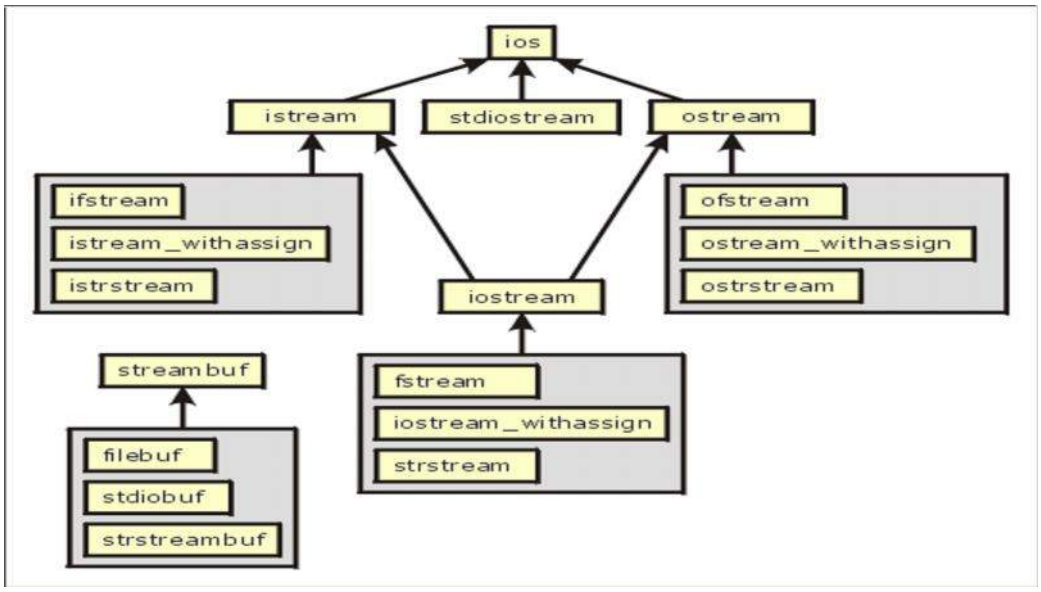


fig: Stream class hierarchy for I/O operations:

The fstream base is derived from the basic class ios. The class ifstream is derived from both istream and fstream base and similarly, the class ofstream is derived from both the ostream and fstream base. Both ifstream and ofstream act as base classes for fstream class. The class filebuf is derived from class streambuf. The complete class hierarchy is shown above.Central to file handling are three classes. They are ifstream, ofstream and fstream.

That is why, if you include fstream.h file in your file handling program, you need not to include iostream.h file as classes of fstream.h inherit from iostream.h only.The functions of these classes have summarized in the below table.

12.2.1 Classes for file stream operation

Classes	Meanings
filebuf	It sets the file buffers to read and write.
fstreambase	It provides the facilities for file operations and consists of open( ) and close( ) member functions. This is base class for fstream, ifstream, ofstream.
ifstream	Stream class to read from files. It provides input operations for file. It inherits the function get( ), getline( ),read( ) and functions supporting random access (seekg( ) and tellg( )) from istream class defined inside iostream.h file.
ofstream	Stream class to write on files. It provides output operations for file. It inherits the function put( ), write( )and functions

	supporting random access (seekp( ) and tellp ()) from ostream class defined inside iostream.h file.
fstream	Stream class to both read and write from/to files.It provides support for simultaneous input and output operations. It inherits all the functions from istream and ostream classes through iostream class defined inside iostream.h file.

12.3 Types of data Files:

Files are used to store data or information permanently for future use. Depending on how data are stored and retrieved, the files are classified into two types. They are **Text file** and **Binary file.**

12.3.1 Text file:

It is a file that stores information in ASCII characters. In text files, each line of text is terminated with a special character known as EOL (End- of-line) character or delimiter character. When this EOL character is read or written, certain internal translations take place.

12.3.2 Binary file:

It is a file that contains information in the same format as it is held in memory. In binary files, no delimiters are used for a line and no translations occur here.

12.4 Opening and Closing files:

In C++, while opening a file, we need the stream like input, output and input_output. To create an input stream you must declare the stream to be of class ifstream. To create an output stream you must declare the stream to be of class ofstream. Streams that will be performing both input and output operations must be declared as class fstream.

Opening a file can be accomplished in two ways:

- Opening file using constructor
- Opening file using open( ) member function

The first method is preferred when a single file is used with a stream. However for managing multiple files with the same stream, the second method is preferred.

12.4.1 Opening file using constructor:

The syntax of opening a file for output purpose only using an object of ofstream class and the constructor is as follows:

ofstream ofstream _object(“file_name”);

ofstream _object is an object of type ofstream and “file name” is any valid identifier of a file to be opened for output purpose only.

```
Example:  ofstreamfout(“results.dat”);      //output only
          ofstreamfout(“text.dat”);         //output only
```

fout is declared to be an object of ofstream type and it is made to represent the file results.dat and text.dat opened for output purpose only.

The syntax of opening a file for input purpose only using an object of ifstream class and the constructor is as follows.

ifstream ifstream _object(“file_name”);

ifstream _object is an object of type ifstream and “file name” is any valid identifier name of a file to opened for input purpose only.

```
Example:  ifstream fin(“results.dat”);      //input only
          ifstream fin(“text.dat”);         //input only
```

fin is declared to be an object of ifstream type and it is made to represent the file results.dat and text.dat opened for input purpose only.

12.4.2 Opening file using open( ):

The syntax for opening a file for output purpose only using an object of ofstream class and open( ) member function is as follows:

ofstream-object.open(“filename”)

ofstream-object is an object of type ofstream and “filename” is any valid name of a file to be opened for output purpose only.

```
Example:  ofstream ofile;
          ofile.open(“data1”);
          ofile.open(“text.dat”);
```

ofile is declared to be an object of ofstream type and is made to represent the file data1 or text.dat opened for output purpose only.

The syntax for opening a file for input purpose only using an object of ifstream class and open( ) member function is as follows:

ifstream-object.open(“filename”)

ifstream-object is an object of type ifstream and “file name” is any valid name of a file to be opened for input purpose only.

```
Example:  ifstream ifile;
          ifile.open(“data1”);
          ifile.open(“text.dat”);
```

ifile is declared to be an object of ifstream type and is made to represent the file data1 or text.dat opened for output purpose only.

If we have to open a file for both input and output operations, we use objects of fstream class. We know that the class fstream is derived from both ifstream and ofstream. As a result objects of the class can invoke all the members of the function of its base class.

The syntax for opening a file, an object of type fstream class and the constructor is as follows:

fstream fstream-object("filename", mode)

The syntax for opening a file an object of type fstream class and the open () member function is as follows:

fstream-object.open("filename",mode)

The need of mode is provided in the following table:

12.4.3. Concepts of file modes (in, out, app modes)

File mode parameter	Meaning	Stream type
ios::app	Append to end of file	ofstream
ios::in	open file for reading only	ifstream
ios::out	open file for writing only	ofstream
ios::ate	Open file for updation and move the file pointer to the end of file	ifstream , ofstream
ios:binary	Opening a binary file	ifstream , ofstream
ios::noreplace	Turn down opening if the file already exists	ofstream
ios::nocreate	Turn down opening if the file does not exists	ofstream
ios::trunc	On opening, delete the contents of file	ofstream

All these flags can be combined using the bitwise operator OR (|).

Example:
```
fstream fout("text.dat",ios::out);        open text.dat in output mode
fstream fin("text.dat",ios::in);          open text.dat in input mode
fstream file;
file.open ("example.bin", ios::out | ios::app | ios::binary);
```

If we want to open the file "example.bin" in binary mode to add data we could do it by the above call to member function.

12.4.4. Closing File:

We learn that opening a file establishes the linkage between a stream object and an operating system file. After the intended operations are done with the file, the file should be safely saved on the secondary storage for later retrieval. This is done by the member function close() The function on its execution removes the linkage between the file and the stream object.

Syntax: stream_object.close();
fout.close();
fin.close();

12.5. Input and output operation on text files:

We know that a text file consists of a group of ASCII values. The data in text files are organized into lines with new line character as terminator. Text files need following types of character input and output operations:

- put() function
- get() function

put(): The put() member function belongs to the class ofstream and writes a single character to the associated stream.

Syntax: ofstream_object.put(ch);

ch is character constant or char variable. The function writes ch onto the file represented by ofstream_object.

```
char ch = 'a';
ofstream fout("text.txt");
fout.put(ch);
```

Write the character stored in ch onto the file represented by the object fout. i.e., text.txt.

get( ): The get() member function belongs to the class ifstream and the function get() reads a single character from the associated stream.

Syntax: ifstream_object.get(ch);

ch is character constant or char variable. The function reads ch from the file represented by the ifstream_object into the variable ch.

```
char  ch = 'a';
ifstream fin("text.txt");
fin.get(ch);
```

Reads a character into the variable ch the current byte position from the file represented by the object fin, i.e., text.txt.

String I/O:

The getline( ) function:

It is used to read a whole line of text. It belongs to the class ifstream.

Syntax: fin.getline(buffer, SIZE);

Reads SIZE characters from the file represented by the object fin or till the new line character is encountered, whichever comes first into the buffer.

Example:
```
char book[SIZE];
fstream fin;
fin.getline(book, SIZE);
```

Input and output operation on binary files:

The binary files are of very much use when we have to deal with database consisting of records. Since the records usually comprise heterogeneous data types, the binary files help optimize storage space and file I/O would be faster when compared to text files. Binary files needs following types of input and output operations.

- write() member function
- read() member function

write( ): The write() member function belongs to the class ofstream and which is used to write binary data to a file.

Syntax: ofstream_object.write((char *) &variable, sizeof(variable));

fout is an object of type ofstream. The function requires two arguments. The first argument ofstream_object is the address of the variable, the contents of which are written to the file and the second argument is the size of the variable. The address of the variable is type casted to pointer to char type. It is because the write function does not bother to know the type of variable. It requires data in terms of only bytes. The function writes the contents of variable to the file represented by the object fout.

Example:
```
student s;
ofstream fout("std.dat",ios::binary);
fout.write((char*) &s, sizeof(s));
```

Write the contents of the object s to the file std.dat.

read( ): The read() member function belongs to the class ifstream and which is used to read binary data from a file.

Syntax: ifstream_object.read((char *) &variable, sizeof(variable));

ifstream_object is an object of type ifstream. The function requires two arguments. The first argument is the address of the variable, the contents of which are read from the file and the second argument is the size of the variable.

The address of the variable is type casted to pointer to char type. It is because the read function does not bother to know the type of variable. It requires data in terms of only bytes. The function reads a record from the file represented by the object fin to the object std.

Example:
```
student s;
ifstream fin("std.dat",ios::binary);
fin.read((char*) &s, sizeof(s));
```

Read a student record from the file std.dat into the object s.

12.6. Detecting end of file:

While reading the contents of a file, care has to be taken to see to it that the operation does not cross the end of file. The ios class provides a member function by name eof(), which helps in detecting the end of file. Once the end of file is detected with the use of eof() member function, we can stop reading further.

eof() returns true (non zero) if end of file is encountered while reading; otherwise return false(zero).

```
if(fin.eof())
{
        statements;
}
```

This is used to execute set statements on reaching the end of the file represented by the object fin.

```
while (!fin.eof())
{
        statements;
}
```

This is used to execute set statements as long as the end of the file fin is not reached.

12.7. File pointers and their manipulation:

In C++, the file I/O operations are associated with the two file pointers, known as get pointer and the put pointer. These are synonymous for input pointer and output pointer respectively. They are useful in traversing the opened file while reading or writing.

- ifstream, like istream, has a pointer known as the get pointer that points to the element to be read in the next input operation.
- ofstream, like ostream, has a pointer known as the put pointer that points to the location where the next element has to be written.

When an input or output operation is performed, the appropriate pointer is automatically advanced. So the programmer need not bother about incrementing the file pointer to the next location inside the file for the next action.

There are three modes under which we can open a file:

- Read only mode
- Write only mode
- Append mode

When a file is opened in a read only mode, the get pointer is automatically set to the very first byte (0^{th} byte) of the file. This helps to read the file contents from the beginning (The bytes in a file is numbered starting from zero).

Similarly, when a file is opened in a write only mode, the contents of file are erased (if it exists) and the put pointer is set to the first byte of the file, so that we can write data from the beginning. In some situations, it would be necessary for us to add new data (or text) to the existing file. In this case we have to open the file in append mode. When a file is opened in a append mode, the put pointer moves to the end-of-file, so that we write new data from that location.

These internal stream pointers that point to the reading or writing locations within a stream can be manipulated using the following member functions:

- seekg()
- seekp()
- tellg()
- tellp()

seekg():

Move the get pointer to a specified location from the beginning of a file. There are two types:

- seekg(long);
- seekg(offset, seekdir);

The seekg(long) moves the get pointer to a specified location from the beginning of a file.

Example: inf.seekg(20) ;

The above example tells that the get pointer points to 20^{th} byte in a file from 0^{th} byte.

The seekg(offset, seekdir) has two arguments: offset and seekdir. The offset indicates the number of bytes the get pointer is to be moved from seekdir position.

The offset takes long data type and seekdir (direction for seeking the offset position inside a file) takes one of the following three seek direction constants. These constants are defined in ios class.

Constant	Meaning
ios::beg	Offset specified from the beginning of the file
ios::cur	Offset specified from the current position of the get pointer
ios::end	Offset specified from the end of the file

Examples: inf.seekg(0, ios::beg);

Move the get pointer to the 0th byte (i.e., beginning of the file).

inf.seekg(20, ios::beg);

Moves the get pointer to the 20th byte (i.e., from current position of the file in forward direction).

inf.seeg(-20, ios::beg);

The above example tells that the get pointer points to 20th byte in a file from end of file in backward direction.

seekp():

Move the put pointer to a specified location from the beginning of a file. There are two types:

- seekp(long);
- seekp(offset, seekdir);

The seekp(long) moves the put pointer to a specified location from the beginning of a file.

Example: inf.seekg(20) ;

The above example tells that the put pointer points to 20th byte in a file from 0th byte.

The seekp(offset, seekdir) has two arguments offset and seekdir. The offset indicates the number of bytes the put pointer is to be moved from seekdir position.

The offset takes long data type and seekdir (direction for seeking the offset position inside a file) takes one of the following three seek direction constants. These constants are defined in ios class (see above table).

Examples: inf.seekp(0, ios::beg);

Move the put pointer to the 0th byte (i.e, beginning of the file) for writing.

inf.seekg(20, ios::beg) ;

Move the put pointer to the 20th byte (i.e, from current position of the file in forward direction) for writing.

```
inf.seekg(-20, ios::beg) ;
```

The above example tells that the put pointer points to 20th byte in a file from end of file in backward direction.

tellg( ) member function:

The ifstream class provides the member function name tellg(). The purpose of the function is to return current position of the get pointer.

Syntax:
```
int      position;
position = fin.tellg();
```

tellp() member function:

The ifstream class provides the member function name tellp( ); The purpose of the function is to return current position of the put pointer.

Syntax:
```
int     position;
position = fin.tellp();
```

Basic operation on binary file in C++

The basic Operation on Binary File in C++ are

1. Searching.
2. Appending data.
3. Inserting data in sorted files.
4. Deleting a record
5. Modifying data

Points to remember:

- File: The information / data stored under a specific name on a storage device, is called a file.
- Stream: It refers to a sequence of bytes.
- ifstream: Stream class to read from files. It provides input operations for file.
- ofstream: Stream class to write on files. It provides output operations for file.
- fstream: Stream class to both read and write from/to files.
- Text file: It is a file that stores information in ASCII characters.

- Binary file: It is a file that contains information in the same format as it is held in memory.
- File can be opened in two ways :
  a. Opening file using constructor: Useful when a single file used with stream.
  b. Opening file using open( ): Useful for managing multiple files with the same stream.
- The classes defined with inside fstream.h derive from classes under iostream.h.
- File can be closed using function close ( ).
- File modes: Describes how a file is to be used.
- ios::in: Open file for reading only.
- ios::out: Open file for writing only.
- ios::app: Append to end of file.
- The put() member function belongs to the class ofstream and writes a single character to the associated stream.
- The get() member function belongs to the class ifstream and the function get() reads a single character from the associated stream.
- getline(): It is used to read a whole line of text. It belongs to the class ifstream.
- The write() member function belongs to the class ofstream and which is used to write binary data to a file.
- The read() member function belongs to the class ifstream and which is used to read binary data from a file.
- eof():Helps in detecting the end of file.
- eof(): returns true (non-zero) if end-of-file is encountered while reading, otherwise return false(zero).
- seekg():Moves the get pointer to a specified location from the beginning of a file.
- The seekg(long) moves the get pointer to a specified location from the beginning of a file.
- seekp():Moves the put pointer to a specified location from the beginning of a file.

- tellg(): The ifstream class provides the member function name tellg(); The purpose of the function is to return current position of the get pointer.
- tellp(): The ifstream class provides the member function name tellp(); The purpose of the function is to return current position of the put pointer.

One marks questions:

1. Which header file is required for file handling functions in C++.
2. What is stream?
3. Name the streams generally used for file I/O.
4. What are output streams?
5. What are input streams?
6. Mention the methods of opening file within C++ program.
7. Write the member functions belonging to fstream class.
8. What is ifstream class
9. What is ofstream class.
10. Write the member functions belonging to ofstream class.
11. Write the member functions belonging to ifstream class.
12. Name the stream classes supported by C++ for file input.
13. Name the stream classes supported by C++ for output.
14. Mention the file modes.
15. What is ios :: in?
16. What is ios::out?
17. Mention the types of file.
18. What is text file.
19. What is binary file.
20. What is the use of write ( ) function.
21. What is the use of writeln ( ) function.
22. What is the use of get ( ) function.
23. What is the use of put ( ) function.
24. What is the use of getline ( ) function.
25. What is the use of read ( ) function.
26. What is the use of seekp ( ) function.
27. What is the use of seekg ( ) function.
28. What is the use of eof ( ) function.

Two marks questions:

1. What is stream? Name the streams generally used for file I/O.
2. What are input and output streams?
3. Mention the methods of opening file within C++ . Discuss any one.
4. Write the member functions belonging to fstream class.
5. Differentiate between ifstream class and ofstream class.
6. Differentiate between read ( ) and write ( ).
7. Differentiate between get ( ) and getline ( ).

8. Write the member functions belonging to ofstream class.
9. Write the member functions belonging to ifstream class.
10. Name the stream classes supported by C++ for file input and output.
11. What are the advantages of saving data in Binary form

Three marks questions:

1. Mention the methods of opening file within C++ program. Discuss.
2. Differentiate between ifstream class and ofstream class.
3. Differentiate between read ( ) and write ( ).
4. Differentiate between get ( ) and getline ( ).
5. Name the stream classes supported by C++ for file input and output.
6. Mention the types of file. Explain any one.
7. What are the advantages of saving data in 1.Binary form. 2. Text form.

Five marks questions:

1. What are input and output streams?
2. What is significance of fsream.h header file.
3. Mention the methods of opening file within C++, Discuss.
4. Differentiate between ifstream class and ofstream class.
5. Differentiate between read ( ) and write ( ) with example.
6. Differentiate between get ( ) and getline ( ) with example.
7. Explain any three file modes.
8. Differentiate between ios::in and ios::out.

*****