EXERCISE

Q1. Select the best answer for the following MCQs.

1. Eof( ) stands for _____
   a. Errors-on-file
   b. End-of-file
   c. Exit-of-file
   d. None

2. In file handling open ( ) function is used _____
   a. To open C++ compiler
   b. To open a file
   c. Both a and b
   d. None

3. Default mode parameter for ofstream is _____
   a. Ios out
   b. Ios in
   c. Ios binary
   d. Both a and b

4. A text file has the extension _____
   a. Doc
   b. Docx

c. Txt

d. Bin

5. Ios::binary is used as an argument to open ( ) function for _____

a. Opening file for input operation

b. Opening file for output operation

c. Opening file in binary mode

d. None

Answers

1.	End-of-life	2.	To open a file
3.	Ios out	4.	Txt
5.	Opening file in binary mode		

EXTENSIVE QUESTIONS

Q3. Write long answers for the following questions

i. What is file handling? Explain different types of file.

Ans.

File Handling

File handling concept in C++ language is used for storing data permanently in computer. It provides a mechanism to write output of a program into a file and read from a file.

Types of files in C++

C++ divides into two types

i. text files
ii. binary files

1. **Text files**

   Text files can be a stream of characters that a computer can process sequentially. It is not only processed sequentially but only in forward direction.

   Processing on characters

   Text files only process characters. They can only read or write data one character at a time. A text stream in C++ is a special kind of life. Depending on the requirements of the operating system, newline characters may be converted to or from carriage-return combinations depending on whether data is being written to or read from the file.

   Character conversion

   Other character conversions may also occur to satisfy the storage requirements of the operating system. These translations occur transparently and they occur because the programmer has signaled the intention to process a text file

   Functions for text

   In C++ programming language, functions are provided that deal with lines of text but these still essentially process data one character at a time.

   Operations on text file

   A text file is usually opened for only one kind of operation (reading, writing or appending) at any given time

1. **binary files**

   Binary files is a collection of bytes

   Difference between byte and character

   In C programming language, a byte and a character are equivalent. Hence, a binary file is also referred to as a character stream but there are two essential differences.

1. No special processing of the data occurs and each byte of data is transferred to or from the disk unprocessed
2. C programming language places no constructs on the file and it may be read from, or written to in any manner chosen by the programmer

Processing of binary files

Binary files can be either processed sequentially or depending on the needs of the application they can processed using random access techniques.

Techniques of processing on files

In C++ programming language, processing a file using random access techniques involve moving the current file position to an appropriate place in the file before reading or writing data. This indicates a second characteristics of binary files.

They are generally processed using read and writes operations simultaneously.

Example

A database file will be created and processed as a binary file. A record update operation will involve locating the appropriate record. Reading the record into memory, modifying it in some way and finally writing the record back to disk at its appropriate location in the file.

These kinds of operations are common to many binary files but are rarely found in applications that process text files.

ii. Describe different types of operations performed on the files

Ans

You can perform four major operations on the file either text or binary

i. Creating new file
ii. Opening an existing file
iii. Closing a file
iv. Reading from and writing information to a file

1. Creating a new file

In C++ to create a file and write to it we will use ofstream class

Let us say we want to create a file called codebind.txt and write something in it

```
//creating and writing to a file - C++

#include <iostream>

#include <fstream>

Using namespace std ;

Int main ( )

Ofstream file;

File.open ("codebind.txt");

File < <"please write this text to a file.\n this text is written using C++\n";

File.close ( );

Return 0;

}
```

Output of the program

```
[file codebind.txt]
Please write this text to a file.
This text is written using C++
```

2. Opening an existing file

To open a file the function open ( ) is used whose syntax is

MyFile.open(filename):

Here myFile is an internal variable, actually an object used to handle the file whose name is written in the parenthesis. To declare this variable the following statement is used

Ifstream myFile :

The dot (.) operator is used between the variable myFile and open function myFile is an object of ifstream while open ( ) is the function of ifstream. The argument for open function is the name of the file on the disk which should be enclosed in double quotes. The file name can be simple file name like "sale.txt". it can be fully qualified path name like "C:\myprogs\sale.txt"

Program

```
#include <iostream>
#include <fstream>
Using namespace std;
Int main ( )
{
Ofstream file;
File.open ("example.txt");
Return 0;
}
```

We have opened the file example.txt to write on it example.txt file must be created in your working directory. We can aslo open the file for both reading and writing purposes. Let's see how to do this

```
#include <iostream>
#include <fstream>
Using namespace std;
Int main ( )
{
Fstream file;
File open ("example.txt, ios: :out | ios: :in);
Return 0;
}
```

3. Closing a file

Once we have read the file it must be closed. It is the responsibility of the programmer to close the file. We can close the file by using the following statement

```
myFile.close( );
```

the function close ( ) does not require any argument, as we are going to close the file associated with myFile. Once we close the file, no file will be associated with myFile

Program

```
#include <iostream>
#include <fstream>
Using namespace std;
Int main ( )
{
Oftsream file;
File.open("example.txt");
File.close ( );
Return 0;
}
```

4. Reading from and writing information to a file

To read a word from the file we can write as

Myfile>>c;

So the first word of the will be read in c, where c is a character array. It is similar as we used with cin. There are certain limitation to this. It can read just one word at one time. It means, on encountering a space it will stop reading further. Therefore, we have to use it repeadetly to read the complete file. We can also read multiple words at a time as

myFile>>c1>>c2>>c3;

the first word will be read in c1 second in c2 and third in c3. Before reading the file, we should know some information regarding the structure of the file. If we have a file of an employee, we should know that the first word is employee's name, second name is salary etc. so that we can read the first word in a string and second word in an int variable.

Program 1

Consider the following input file inputfile.txt shown in fig 9.2 which is read and displayed on the screen.

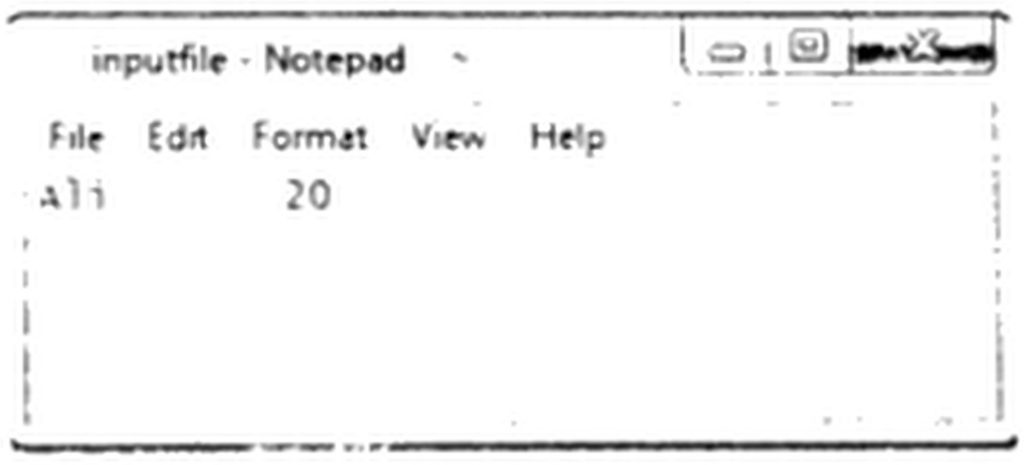

Figure 9.2: Input File to be Read

```
// Reading input file
#include <iostream.h>
#include <fstream.h>
#include <conio.h>
int main ()
{
ifstream myfile("c:\\inputfile.txt"),
char ch [20];
int m;
myfile>>ch>>m,
cout<<ch<<"/t"<<m,
myfile close();
getch();
return 0,
}
```

The output of the above program is shown in fig 9.3

Figure 9.3: Output of Reading Program

iii. Explain bof( ) and eof( ) functions

Ans

Bof ( ) – beginning-of-file

Bof ( ) – beginning-of-file

The **bof ( )** is a pointer which returns true if the current position of the pointer is at the beginning of the input file stream and false otherwise. It means that it tells the compiler whether the cursor is at the beginning of file or not.

Eof ( ) – end-of-file

The **eof ( )** is a pointer which returns true when there are no more data to be read from an input file stream and false otherwise. It means that this function checks whether control has reached to the end of file or not. This function is very useful in the case when we do not know the exact number of records in a file

Rules for using eof ( )

1. Always test for the end-of-file condition before processing data
2. Use a while loop for getting data from an input file stream

Program

```
//reading a text file using eof ( ) function

#include <iostream.h>

#include <conio.h>

#include <fstream.h>

Int main ( )

{

Char ch [50] :

Ifstream flag ("c:\\TestRecord.txt");

Cout < <"Output is"< < end1;

While(!flag.eof( ) )

{

Flag>>ch;

Cout<<ch<<end1;

}

Flag.close ( );

Getch( );
```

```
Return0;
}
```

Output of the program

```
Output is
Ali 20
Anwar 15
Zainab 17
Zonash 21
```

Explanation

In the above program, the input file **TestRecord** is loaded and the **eof ( )** function detects the end of the file. The program reads the file record by record into the string variable ch. Which is then displayed on the screen

iv. Define streams. Describe input and output streams in detail.

Ans

Stream

In C++ a stream is a sequence of bytes associated with a file. Most of the times streams are used to assure a good and secure flow of data between an application and file

Types of stream

1. Input stream
2. Output stream

1. **Input stream**

Input streams take any sequence of bytes from an input devices such as keyboard, a file or a network

2. **Output stream**

   Output streams are used to hold output for a particular data consumer such as monitor, a file, or a printer

v. **What is meant by the term mode of file opening? Describe different modes of opening file.**

   Ans

Modes of opening a file

While opening a file, we tell the computer what we want to do with it i.e we want to read the file or wrote into the file or want to modify it. In order to open a file in any desired mode the member function **open ( )** should take mode as an argument along with the file name

Syntax

Its general syntax is shown below

Open (filename, mode);

Here **filename** representing the name of the file to be opened and **mode** is an optional parameter with a combination of the following tags

Mode	Description
ios::in	*Open for input operations (Reading a file)*
ios::out	*Open for output operations (Writing a file)*
ios::binary	*Open in binary mode*
ios::ate	*Open a file for output and move the read/write 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 opened for output operations its existing contents are deleted and replaced by the new one*

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

Example

If we want to open the file **test.bin** in binary mode to add data we could do it by the following call to member function **open ( )**

```
Ofstream myfile:
Myfile.open ("test.bin", ios: :out | ios : :app|ios: :binary):
```

Open ( ) member functions

Each one of the **open ( )** member functions of the classes **ofstream, ifstream** and **fstream** has a default mode that is used if the file is opened without a second argument

Class	Default Mode Parameter
ofstream	ios out
ifstream	ios in
fstream	ios in | ios out

For **istream** and **ofstream** classes, **ios::in** and **ios::out** are automatically and respectively assumed, even if a mode that does not include them is passed as second argument to the **open ( )** member function

Program

Let us see an example program that creates a text file "**example1.txt**" and writes a line of text in it.

```
//opening a file and writing into it
#include <iostream.h>
#include <fstream.h>
#include <conio.h>
Int main ( )
```

```
{
Ofstream mylife;
Myfile.open ("example1.txt");
Myfile<< "This is a program that tells you how to write a file.\n";
Myfile.close( );
Getch();
Return 0;
}
```

Explanation

This code creates a file called **example1.txt** and inserts a sentence into it in the same way as we do with cout, but using the file stream myfile instead. Fig 9.1 shows the output of the above program in the form of a text file that is created in the same working directory.

Figure 9.1: Opening and Writing into a Text File

Practice all the programs given in the chapter.

Ans

Practical work