ASSIGNMENT 6

FILE HANDLING

Problem Statement :-
Develop an object oriented program in C++ to create a database of the personnel information system containing the following information: Name, Date of Birth, Blood group, Height, Weight, Insurance Policy number, Contact address, telephone number, driving licence no. etc Construct the database with suitable member functions for initializing and destroying the data viz constructor, default constructor,copy constructor, destructor, static member functions friend class, this pointer, inline code and dynamic memory allocation operators-new and delete.
Learning Objectives :-

- File handliong
  Etc.,

Theory :-

Abstractly, a file is a collection of bytes stored on a secondary storage device, which is generally a disk of some kind. The collection of bytes may be interpreted, for example, as characters, words, lines, paragraphs and pages from a textual document; fields and records belonging to a database; or pixels from a graphical image. The meaning attached to a particular file is determined entirely by the data structures and operations used by a program to process the file. It is conceivable (and it sometimes happens) that a graphics file will be read and displayed by a program designed to process textual data. The result is that no meaningful output occurs (probably) and this is to be expected. A file is simply a machine decipherable storage media where programs and data are stored for machine usage.

Essentially there are two kinds of files that programmers deal with text files and binary files. These two classes of files will be discussed in the following sections.

ASCII Text files

A text file can be a stream of characters that a computer can process sequentially. It is not only processed sequentially but only in forward direction. For this reason a text file is usually opened for only one kind of operation (reading, writing, or appending) at any given time.

Similarly, since text files only process characters, they can only read or write data one character at a time. (In C Programming Language, Functions are provided that deal with lines of text, but these still essentially process data one character at a time.) A text stream in C is a special kind of file. Depending on the requirements of the operating system, newline characters may be converted to or from carriage-return/linefeed combinations depending on whether data is being written to, or read from, the file. 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 signalled the intention to process a text file.

Binary files

A binary file is no different to a text file. It is a collection of bytes. 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.

No special processing of the data occurs and each byte of data is transferred to or from the disk unprocessed.

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.

Binary files can be either processed sequentially or, depending on the needs of the application, they can be processed using random access techniques. In C Programming Language, processing a file using random access techniques involves moving the current file position to an appropriate place in the file before reading or writing data. This indicates a second characteristic of binary files. They a generally processed using read and write operations simultaneously.

For 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

The type FILE is used for a file variable and is defined in the stdio.h file. It is used to define a file pointer for use in file operations. Before we can write to a file, we must open it. What this really means is that we must tell the system that we want to write to a file and what the file name is. We do this with the fopen() function illustrated in the first line of the program. The file pointer, fp in our case, points to the file and two arguments are required in the parentheses, the file name first, followed by the file type.

Reading (r)

When an r is used, the file is opened for reading, a w is used to indicate a file to be used for writing, and an a indicates that you desire to append additional data to the data already in an existing file. Most C compilers have other file attributes available; check your Reference Manual for details. Using the r indicates that the file is assumed to be a text file. Opening a file for reading requires that the file already exist. If it does not exist, the file pointer will be set to NULL and can be checked by the program.

Writing (w)

When a file is opened for writing, it will be created if it does not already exist and it will be reset if it does, resulting in the deletion of any data already there. Using the w indicates that the file is assumed to be a text file.

Appending (a)

When a file is opened for appending, it will be created if it does not already exist and it will be initially empty. If it does exist, the data input point will be positioned at the end of the present data so that any new data will be added to any data that already exists in the file. Using the a indicates that the file is assumed to be a text file.

Outputting to the file

The job of actually outputting to the file is nearly identical to the outputting we have already done to the standard output device. The only real differences are the new function names and the addition of the file pointer as one of the function arguments. In the example program, fprintf replaces our familiar printf function name, and the file pointer defined earlier is the first argument within the parentheses. The remainder of the statement looks like, and in fact is identical to, the printf statement.

Closing a file

To close a file you simply use the function fclose with the file pointer in the parentheses. Actually, in this simple program, it is not necessary to close the file because the system will close all open files before returning to DOS, but it is good programming practice for you to close all files in spite of the fact that they will be closed automatically, because that would act as a reminder to you of what files are open at the end of each program.

You can open a file for writing, close it, and reopen it for reading, then close it, and open it again for appending, etc. Each time you open it, you could use the same file pointer, or you could use a different one. The file pointer is simply a tool that you use to point to a file and you decide what file it will point to. Compile and run this program. When you run it, you will not get any output to the monitor because it doesnâ€™t generate any. After running it, look at your directory for a file named TENLINES.TXT and type it; that is where your output will be. Compare the output with that specified in the program; they should agree! Do not erase the file named TENLINES.TXT yet; we will use it in
some of the other examples in this section.

fopen()

```
FILE *fopen(const char *path, const char *mode);
```

The fopen() function is used to open a file and associates an I/O stream with it. This function takes two arguments. The first argument is a pointer to a string containing name of the file to be opened while the second argument is the mode in which the file is to be opened. The mode can be :

- ‘r’ : Open text file for reading. The stream is positioned at the beginning of the file.
- ‘r+’ : Open for reading and writing. The stream is positioned at the beginning of the file.
- ‘w’ : Truncate file to zero length or create text file for writing. The stream is positioned at the beginning of the file.
- ‘w+’ : Open for reading and writing. The file is created if it does not exist, otherwise it is truncated. The stream is positioned at the beginning of the file.
- ‘a’ : Open for appending (writing at end of file). The file is created if it does not exist. The stream is positioned at the end of the file.
- ‘a+’ : Open for reading and appending (writing at end of file). The file is created if it does not exist. The initial file position for reading is at the beginning of the file, but output is always appended to the end of the file.

The fopen() function returns a FILE stream pointer on success while it returns NULL in case of a failure.

fread() and fwrite()

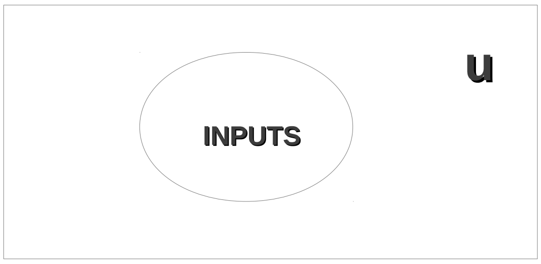
```
size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream);

size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream);
```

The functions fread/fwrite are used for reading/writing data from/to the file opened by fopen function. These functions accept three arguments. The first argument is a pointer to buffer used for reading/writing the data. The data read/written is in the form of ‘nmemb’ elements each ‘size’ bytes long.

In case of success, fread/fwrite return the number of bytes actually read/written from/to the stream opened by fopen function. In case of failure, a lesser number of byes (then requested to read/write) is returned.

MATHEMATICAL MODEL:
1) PHASE 1: - INPUTS
Here we take many inputs such as name,address,mobile_no,date-of-birth,city,weight,height,blood_group,state,nation,insurance_no,license_no,policy_no.

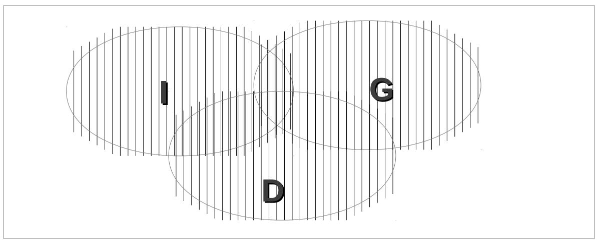
INPUTS={name,address,mobile_no,dateofbirth,city,weight,height,blood_group,state,nation,insurance_no,license_no,policy_no.}



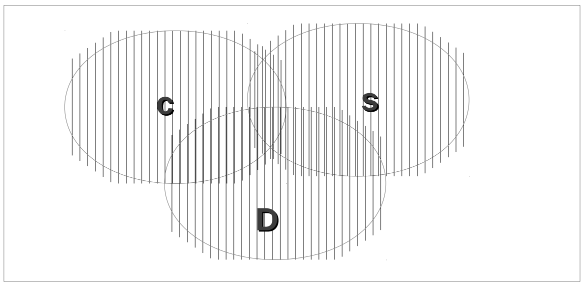
2) PHASE 2:- PROCESSING

Functions:-	Getdata()	set G
	Display()	set D
	Compare()	set C
	Search ()	set S

To accept data and display data we take all the inputs from the user and then display them on the screen.



To search a record we first compare the name of the person and then display other all details .



-
-
-
-
-
-
-
-
- **Algorithm** :-
- step 1:-start.
- step 2:-we declare a global variable so that it can be used anywhere in the program.
- step 3:-we define a class,where we declare pointer char variable for name,birth

date,blood group and float variable for height and weight.

- step 4:-we declare a friend class named data.
- step 5:-the friend class contains the pointer variables to keep the information about city,state,nation,pincode. etc.
- step 6:-we have getdata & display functions in both files.classes to accept the data from the user & write on terminal as well as in the file created named as "data.txt"
- step 7:-we have comare and search function to find only record stored in the file"data.txt".
- step 8:-we use fout function to write the entries into "file.txt".
- step 9:-in main,
- step 10:-we call the getdata & display function,then there is a choice to search a record.
- step 11:-stop

Conclusion :-
Using file handling we can perform matrix operations.