# Assignment-10
# Implementation of sequential file

**AIM**: Department maintains a student information. The file contains roll number, name, division and address. Write a program to create a sequential file to store and maintain student data. It should allow the user to add, delete information of student. Display information of particular employee. If record of student does not exist an appropriate message is displayed. If student record is found it should display the student details.

.

**Objective**

**Type of File**

- Binary File ○ The binary file consists of binary data
  - ○ It can store text, graphics, sound data in binary format ○ The binary files cannot be read directly ○ Numbers stored efficiently
- Text File ○ The text file contains the plain ASCII characters
  - ○ It contains text data which is marked by ‗end of line' at the end of each record ○ This end of record marks help easily to perform operations such as read and write ○ Text file cannot store graphical data.

**File Organization :**

The proper arrangement of records within a file is called as file organization. The factors that affect file organization are mainly the following:

- Storage device
- Type of query
- Number of keys
- Mode of retrieval/update of record Different types of File Organizations are as :
- Sequential file
- Direct or random access file
- Indexed sequential file
- Multi-Indexed file

**Sequential file :** In sequential file, records are stored in the sequential order of their entry. This is the simplest kind of data organization. The order of the records is fixed. Within each block, the records are in sequence . A sequential file stores records in the order they are entered. New records always appear at the end of the file.

***Features of Sequential files :***

- Records stored in pre-defined order.
- Sequential access to successive records.
- Suited to magnetic tape.
- To maintain the sequential order updating becomes a more complicated and difficult task. Records will usually need to be moved by one place in order to add (slot in) a

record in the proper sequential order. Deleting records will usually require that records be shifted back one place to avoid gaps in the sequence.

- Very useful for transaction processing where the hit rate is very high e.g. payroll systems, when the whole file is processed as this is quick and efficient.
- Access times are still too slow (no better on average than serial) to be useful in on-line applications.

***Drawbacks of Sequential File Organization***

- Insertion and deletion of records in in-between positions huge data movement
- Accessing any record requires a pass through all the preceding records, which is time consuming. Therefore, searching a record also takes more time.
- Needs reorganization of file from time to time. If too many records are deleted logically, then the file must be reorganized to free the space occupied by unwanted records

***Primitive Operations on Sequential files***

***Open***—This opens the file and sets the file pointer to immediately before the first record
***Read-next***—This returns the next record to the user. If no record is present, then EOF condition will be set.
***Close—***This closes the file and terminates the access to the file
***Write-next***—File pointers are set to next of last record and write the record to the file
***EOF***—If EOF condition occurs, it returns true, otherwise it returns false
***Search***—Search for the record with a given key
***Update***—Current record is written at the same position with updated values

- **Direct or random access file :** Files that have been designed to make direct record retrieval as easy and efficiently as possible is known as directly organized files. Though we search records using key, we still need to know the address of the record to retrieve it directly. The file organization that supports Files such access is called as direct or random file organization. Direct access files are of great use for immediate access to large amounts of information. They are often used in accessing large databases.
- ***Advantages of Direct Access Files :***
  - Rapid access to records in a direct fashion. ○ It doesn't make use of large index tables and dictionaries and therefore response times are very fast.
- **Indexed sequential file :** Records are stored sequentially but the index file is prepared for accessing the record directly. An index file contains records ordered by a record key. The record key uniquely identifies the record and determines the sequence in which it is accessed with respect to other records.
- A file that is loaded in key sequence but can be accessed directly by use of one or more indices is known as an indexed sequential file. A sequential data file that is indexed is called as indexed sequential file. A solution to improve speed of retrieving target is index sequential file. An indexed file contains records ordered by a record key. Each record contains a field that contains the record key.

- This system organizes the file into sequential order, usually based on a key field, similar in principle to the sequential access file. However, it is also possible to directly access records by using a separate index file. An indexed file system consists of a pair of files: one holding the data and one storing an index to that data. The index file will store the addresses of the records stored on the main file. There may be more than one index created for a data file e.g. a library may have its books stored on computer with indices on author, subject and class mark.

***Characteristics of Indexed Sequential File***

- Records are stored sequentially but the index file is prepared for accessing the record directly
- Records can be accessed randomly
- File has records and also the index
- Magnetic tape is not suitable for index sequential storage
- Index is the address of physical storage of a record
- When randomly very few are required/accessed, then index sequential is better
- Faster access method
- Addition overhead is to maintain index
- Index sequential files are popularly used in many applications like digital library

**Primitive operations on Index Sequential files (IS)**

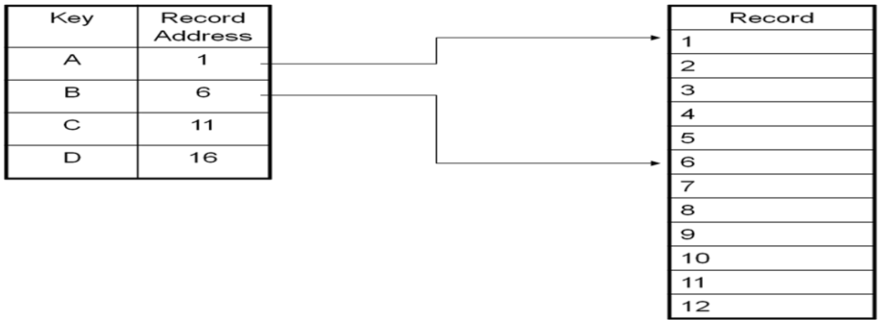
- ***Write (add, store) :*** User provides a new key and record, IS file inserts the new record and key.
- ***Sequential Access (read next) :*** IS file returns the next record (in key order)
- ***Random access (random read, fetch) :*** User provides key, IS file returns the record or "not there"
- ***Rewrite (replace) :*** User provides an existing key and a new record, IS file replaces existing record with new.
- ***Delete :*** User provides an existing key, IS file deletes existing record

**Types of Indexed Files :** There are two types of indexed files:

- Fully Indexed
- Indexed Sequential

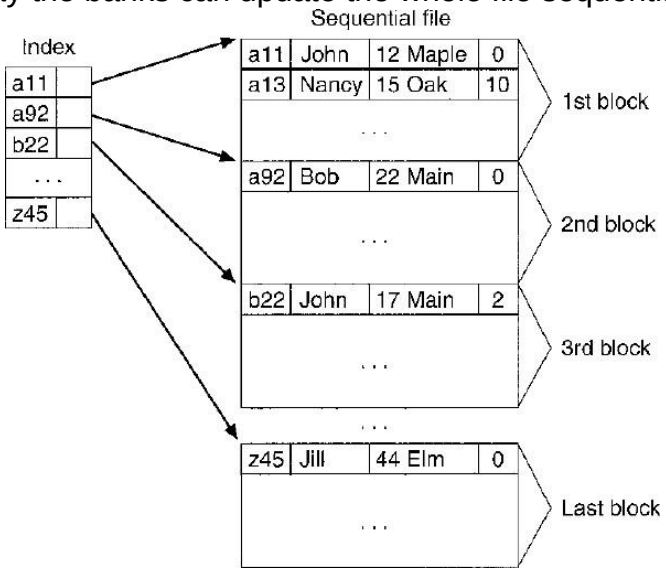
**Fully Indexed Files :**

An index to a fully indexed file will contain an entry for every single record stored on the main file. The records will be indexed on some key e.g. student number. Very large files will have correspondingly large indices. The index to a (large) file may be split into different index levels. When records are added to such a file, the index (or indices) must also be updated to include their relative position and change the relative position of any other records involved.




**Indexed Sequential Files :**

This is basically a mixture of sequential and indexed file organisation techniques. Records are held in sequential order and can be accessed randomly through an index. Thus, these files share the merits of both systems enabling sequential or direct access to the data. The index to these files operates by storing the highest record key in given cylinders and tracks. Note how this organisation gives the index a tree structure. Obviously this type of file organisation will require a direct access device, such as a hard disk. Indexed sequential file organisation is very useful where records are often retrieved randomly and are also processed in (sequential) key order. Banks may use this organisation for their auto-bank machines i.e. customers randomly access their accounts throughout the day and at the end of the day the banks can update the whole file sequentially.




***Advantages of Indexed Sequential Files***

1. Allows records to be accessed directly or sequentially.
2. Direct access ability provides vastly superior (average) access times.

***Disadvantages of Indexed Sequential Files***

1. The fact that several tables must be stored for the index makes for a considerable storage overhead.
2. As the items are stored in a sequential fashion this adds complexity to the addition/deletion of records. Because frequent updating can be very inefficient, especially for large files, batch updates are often performed.

**Multi-Indexed file :**In multi-indexed file, the data file is associated with one or more logically separated index files. Inverted files and multilist files are examples of multiindexed files

**Algorithms :**

**1. Algorithm for main function**

MAIN FUNCTION()
S1: Read the two filenames from user master and temporary.
S2: Read the operations to be performed from the keyboard
S3: If the operation specified is create go to the create function, if the operation specified is display go to the display function, if the operation specified is add go to the add function , if the operation specified is delete go to delete function, if the operation specified is
display particular record go to the search function, if the operation specified is exit go to step 4. S4: Stop

**2. Algorithm for create function**

S1: Open the file in the write mode ,if the file specified is not found or unable to open then display error message and go to step5 , else go to step2.
S2: Read the no: of records N to be inserted to the file .
S3: Repeat the step4 N number of times .
S4: Read the details of each student from the keyboard and write the same to the file .
S5: Close the file .
S6: Return to the main function

**3. Algorithm for displaying all records** S1:
Open the specified file in read mode.
S2: If the file is unable to open or not found then display error message and go to step 4 else go to Step 3
S3: Scan all the student details one by one from file and display the same at the console until end of file is reached.
S4: Close the file
S5: Return to the main function

**4. Algorithm for add a record**

S1: Open the file in the append mode ,if the file specified is not found or unable to open then display error message and go to step5 , else go to step2
S2: Scan all the student details one by one from file until end of file is reached.
S3: Read the details of the from the keyboard and write the same to the file
S4: Close the file .
S5: Return to the main function

**5. Algorithm for deleting a record**

S1: Open the file in the append mode ,if the file specified is not found or unable to open then display error message and go to step5 , else go to step2

S2:Accept the roll no from the user to delete the record
S3:Search for the roll no in file.If roll no. exits, copy all the records in the file except the one to be deleted in another temporary file.
S4:Close both files
S5:Now, remove the old file & name the temporary file with name same as that of old file name.

**6. Algorithm for displaying particular record(search)**

S1: Open the file in the read mode ,if the file specified is not found or unable to open then display error message and go to step6 , else go to step2.
S2: Read the roll number of the student whose details need to be displayed.
S3: Read each student record from the file.
S4: Compare the students roll number scanned from file with roll number specified by the user.
S5: If they are equal then display the details of that record else display required roll number not found message and go to step6.
S6: Close the file.
S7: Return to the main function.

**Test Conditions:**

1. Input valid filename.
2. Input valid record.
3. Check for opening / closing / reading / writing file errors

**Sample Input Output MENU**

Choice 1 : Create
Choice 2 : Display
Choice 3 : add
Choice 4 : Delete
Choice 5 : Display particular record
Choice 6 : Exit

**Create**

Accept the records and write into the file
File created successfully **Display**
Display all records present in Master file. **Add**
Accept the record to be add into the file at the end of the file. Record inserted successfully

**Delete**

Accept the record to be deleted.
Record deleted successfully **Search**
Accept the record to be displayed by search.
Display whether the record if it is present else record not present.

## Program-

```
#include<iostream>
#include<fstream>
using namespace std;
int roll_num;
char name[10];
char division[10];
char address[20];
fstream f,r,temp;
void create(){
int n,i;
 f.open("data.txt",ios::out);
cout<<"How many records you want to enter:-";
cin>>n;
for(i=0; i<n; i++){
cout<<"\nInfo for Student "<<i+1<<endl;
cout<<"Enter Roll No.: ";
cin>>roll_num;
cout<<"Enter Name : ";
cin>>name;
cout<<"Enter Division : ";
cin>>division;
cout<<"Enter Address : ";
cin>>address;
 f<<roll_num<<"\t"<<name<<"\t\t "<<division<<"\t\t "<<address<<"\n";
}
f.close();
 cout<<"\n** Data created succesfully **"<<endl;
}
int search(int roll) {
r.open("data.txt",ios::in);
while(!r.eof()) {
r>>roll_num>>name>>division>>address;
 if(roll == roll_num) {
r.close();
return 1;
}
}
r.close();
return 0;
}
void modify_data(){
 int x,roll;
```

```
 cout<<"Enter roll no: ";
 cin>>roll;
 x = search(roll);
 if(x==1) {
 f.open("data.txt",ios::in);
 temp.open("temp.txt",ios::out);
 f>>roll_num>>name>>division>>address;
 while(!f.eof()) {
 if(roll == roll_num) {
 cout<<"Enter name : ";
 cin>>name;
 cout<<"Enter the division : ";
 cin>>division;
cout<<"Enter address : ";
cin>>address;
 temp<<roll_num<<"\t"<<name<<"\t\t "<<division<<"\t\t"<<address<<"\n";
 f>>roll_num>>name>>division>>address;
 continue;
}
temp<<roll_num<<"\t"<<name<<"\t\t "<<division<<"\t\t"<<address<<"\n";
f>>roll_num>>name>>division>>address;
}
 f.close();
 temp.close();
 f.open("data.txt",ios::out);
 temp.open("temp.txt",ios::in);
 temp>>roll_num>>name>>division>>address;
 while(!temp.eof()) {
f<<roll_num<<"\t"<<name<<"\t\t "<<division<<"\t\t "<<address<<"\n";
temp>>roll_num>>name>>division>>address;
}
f.close();
 temp.close();
}
else{
cout<<"\nRecord not exist!\n";
}
}
void delete_data()
{
int x,roll;
cout<<"Enter roll no to delete: ";
 cin>>roll;
 x = search(roll);
 if(x==1) {
 f.open("data.txt",ios::in);
```

```
 temp.open("temp.txt",ios::out);
 f>>roll_num>>name>>division>>address;
 while(!f.eof()){
 if(roll != roll_num) {
 temp<<roll_num<<"\t"<<name<<"\t\t "<<division<<"\t\t" <<address<<endl;
}
f>>roll_num>>name>>division>>address;
}
f.close();
temp.close();
f.open("data.txt",ios::out);
temp.open("temp.txt",ios::in);
temp>>roll_num>>name>>division>>address;
 while(!temp.eof()) {
f<<roll_num<<"\t"<<name<<"\t\t "<<division<<"\t\t "<<address<<"\n";
 temp>>roll_num>>name>>division>>address;
 }
 f.close();
 temp.close();
 }
 else {
cout<<"\nRecord not exist!\n";
}
}
void insert(){
int x,i,roll;
cout<<"Enter Stduent Details : "<<endl;
cout<<"Roll no : ";
 cin>>roll_num;
 x = search(roll);
if(x==1) {
cout<<"\nRecord already exist!\n";
}
else {
f.open("data.txt",ios::app);
 cout<<"Name : ";
 cin>>name;
 cout<<"Division : ";
 cin>>division;
cout<<"Address : ";
cin>>address;
 roll_num = roll;
 f<<roll_num<<"\t"<<name<<"\t\t "<<division<<"\t\t "<<address<<endl;
f.close();
 cout<<"\n** Record added succesfully **"<<endl;
 }
```

```
}
void display()
{
f.open("data.txt",ios::in);
cout<<"Roll No"<<"\t "<<"Name"<<"\t "<<"Division"<<"\t"<<"Address"<<"\n";
f>>roll_num>>name>>division>>address;
 while(!f.eof()) {
cout<<roll_num<<"\t "<<name<<" \t\t"<<division<<"\t\t"<<address<<endl;
 f>>roll_num>>name>>division>>address;
}
f.close();
}
int main(){
int choice;
cout<<"Weclome to Student Database Management\n"<<endl;
do{
cout<<"What do you want to perform ? Choose from options given below :"<<endl;
cout<<"1] Create DB"<<endl;
cout<<"2] Display DB"<<endl;
cout<<"3] Insert Data"<<endl;
cout<<"4] Delete Data"<<endl;
cout<<"5] Modify Data"<<endl;
cout<<"6] Exit"<<endl;
cout<<"\nEnter a choice : ";
cin>>choice;
switch(choice){
case 1:
create();
cout<<"\n";
break;
case 2:
display();
cout<<"\n";
break;
case 3:
insert();
cout<<"\n";
break;
case 4:
delete_data();
cout<<"\n";
display();
break;
case 5:
modify_data();
cout<<"\n";
```

```
break;
case 6:
cout<<"Thank You !! Your Data has been saved .";
cout<<"\n";
break;
default:
cout<<"*** Enter a valid choice!! ***";
cout<<"\n";
}
}while(choice!=6);
}
```

**Output-**

```
Weclome to Student Database Management

What do you want to perform ? Choose from options given below :
1] Create DB
2] Display DB
3] Insert Data
4] Delete Data
5] Modify Data
6] Exit

Enter a choice : 1
How many records you want to enter:-3

Info for Student 1
Enter Roll No.: 12
Enter Name : harsh
Enter Division : g
Enter Address : pune

Info for Student 2
Enter Roll No.: 34
Enter Name : divya
Enter Division : s
Enter Address : mumbai

Info for Student 3
Enter Roll No.: 90
Enter Name : priya
Enter Division : h
Enter Address : nashik

** Data created succesfully **

What do you want to perform ? Choose from options given below :
1] Create DB
2] Display DB
3] Insert Data
4] Delete Data
5] Modify Data
6] Exit
```

```
Enter a choice : 2
Roll No  Name    Division       Address
12       harsh         g             pune
34       divya         s             mumbai
90       priya         h             nashik

What do you want to perform ? Choose from options given below :
1] Create DB
2] Display DB
3] Insert Data
4] Delete Data
5] Modify Data
6] Exit

Enter a choice : 3
Enter Stduent Details :
Roll no : 67
Name : om
Division : b
Address : abad

** Record added succesfully **

What do you want to perform ? Choose from options given below :
1] Create DB
2] Display DB
3] Insert Data
4] Delete Data
5] Modify Data
6] Exit

Enter a choice : 4
Enter roll no to delete: 90

Roll No  Name    Division       Address
12       harsh         g             pune
34       divya         s             mumbai
0        om            b             abad
What do you want to perform ? Choose from options given below :
1] Create DB
2] Display DB
3] Insert Data
4] Delete Data
```

5] Modify Data
6] Exit

Enter a choice : 5
Enter roll no: 67

Record not exist!

What do you want to perform ? Choose from options given below :
1] Create DB
2] Display DB
3] Insert Data
4] Delete Data
5] Modify Data
6] Exit

Enter a choice : 5
Enter roll no: 12
Enter name : Divya
Enter the division : w
Enter address : pune

What do you want to perform ? Choose from options given below :
1] Create DB
2] Display DB
3] Insert Data
4] Delete Data
5] Modify Data
6] Exit

**Conclusion:**

Thus we have implemented sequential file and performed all the primitive operations on it.