DATA STRUCTURES AND ALGORITHMS LABORATORY

Group F

Assignment No. 1

NAME :- OJUS PRAVIN JAISWAL
ROLL NO. :- SACO19108
DIVISION :- A

DATA STRUCTURE LABORATORY

Group F- Assignment 1

Title: Department maintains a student information. The file contains roll number, name, division and address. Allow 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 it is, then the system displays the student details. Use sequential file to maintain the data.

Objectives:

- The local and physical organization of files.
- To understand the concept of sequential files.
- To know about File Operations
- To understand the use of sequential files.
- Sequential file handling methods.
- Creating, reading, writing and deleting records from a variety of file structures.
- Creating code to carry out the above operations.

Outcome: Students will be able to perform various operations on Student Database using sequential file.

Operations: Implement sequential file for student Database and perform following operations on it:

Student Database:

1) Add New Record
2) Display All Records
3) Display by Roll No
4) Deleting a Record
5) Exit

Theory:

Types 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 helps 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:

1. Sequential file
2. Direct or random access file
3. Indexed sequential file
4. Multi-Indexed file

1. 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

2. 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.

3. 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

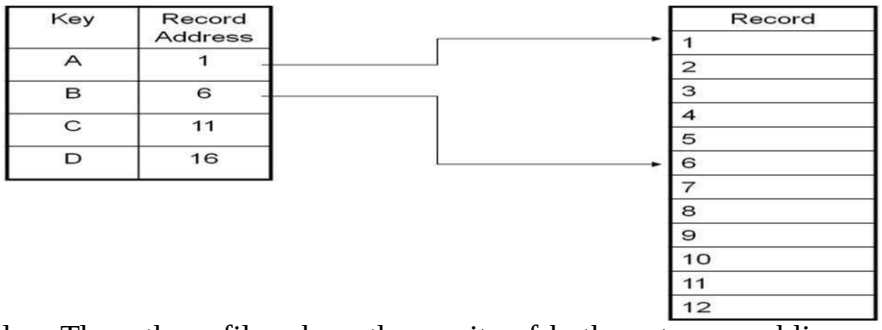
DATA STRUCTURE LABORATORY

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) f ile 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 organization techniques. Records are held in sequential order and can be accessed randomly

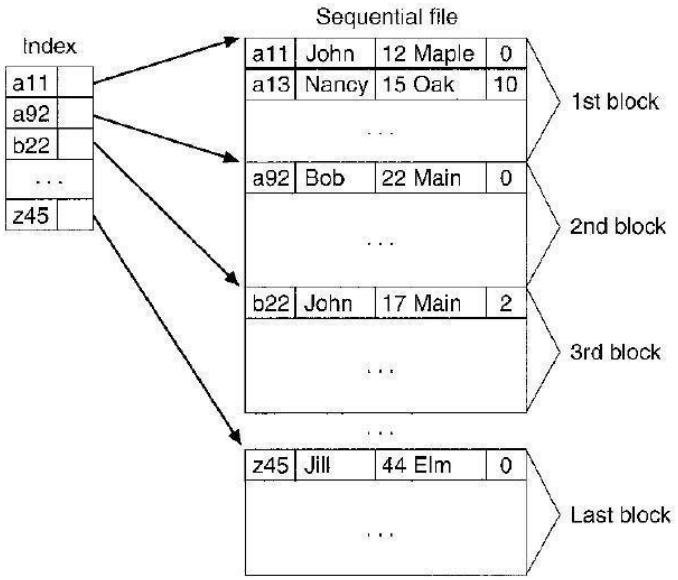


through an index. Thus, these files share the merits of both systems enabling sequential or direct access to the data. The index to these files operates by storing the highest record key in given cylinders and tracks. Note how this organization gives the index a tree structure. Obviously this type of file organization will require a direct access device, such as a hard disk. Indexed sequential file organization is very useful where records are often retrieved randomly and are also processed in (sequential) key order. Banks may use this organization 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.

DATA STRUCTURE LABORATORY



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 addit ion/deletion of records. Because frequent updating can be very inefficient, especially for large files, batch updates are often performed.

4. Multi-Indexed file:

In multi-indexed file, the data file is associated with one or more logically separated index files. Inverted files and multi-list files are examples of multi-indexed 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 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 form the keyboard and write the same to the file

S4: Close the file.

S5: Return to the main function

4. 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

5. 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.

6. 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.

Software Required: g++ / gcc compiler- / 64 bit Fedora, eclipse IDE

Program:

```
//============================================================================
==========
// Name        : SequentialFiles.cpp
// Author      :
// Version     :
// Copyright   : Your copyright notice
// Description : Hello World in C++, Ansi-style
//============================================================================
==========

#include <iostream>
#include <fstream>
#include <cstring>
#include <iomanip>
using namespace std;
const int MAX = 20;
class Student
{
        int rollno;
        char name[20], city[20];
        char div;
        int year;

  public:
        Student()
        {
                strcpy(name, "");
```

```
		strcpy(city, "");
		rollno = year = div = 0;
	}
	Student(int rollno, char name[MAX], int year, char div, char city[MAX])
	{
		strcpy(this->name, name);
		strcpy(this->city, city);
		this->rollno = rollno;
		this->year = year;
		this->div = div;
	}
	int getRollNo()
	{
		return rollno;
	}
	void displayRecord()
	{
		if (rollno != 0)
		{
			cout << '\n';
			cout << setw(5) << rollno << setw(20) << name << setw(5) << year <<
setw(5) << div << setw(10) << city;
		}
	}
};
//==========File Operations ==========
class FileOperations
{
	fstream file;

  public:
	FileOperations(char *filename)
	{
		file.open(filename, ios::in | ios::out | ios::ate | ios::binary | ios::trunc);
	}
	void insertRecord(int rollno, char name[MAX], int year, char div, char city[MAX])
	{
		Student s1(rollno, name, year, div, city);
		file.seekp(0, ios::end);
		file.write((char *)&s1, sizeof(Student));
		file.clear();
	}
	void displayAll()
	{
		Student s1;
		file.seekg(0, ios::beg);
		while (file.read((char *)&s1, sizeof(Student)))
		{
			s1.displayRecord();
		}
		cout<<'\n';
		file.clear();
	}
	void displayRecord(int rollNo)
	{
```

```
	Student s1;
	file.seekg(0, ios::beg);
	bool flag = false;
	while (file.read((char *)&s1, sizeof(Student)))
	{
		if (s1.getRollNo() == rollNo)
		{
			s1.displayRecord();
			flag = true;
			break;
		}
	}
	if (flag == false)
	{
		cout << "\nRecord of " << rollNo << " is not present.";
	}
	file.clear();
}
void updateRecord(int rollno, char name[MAX], int year, char div, char city[MAX])
{
	Student s1(rollno, name, year, div, city);
	file.seekp(-sizeof(s1), ios::cur);
	file.write((char *)&s1, sizeof(Student));
	file.clear();
}
void deleteRecord(int rollno)
{
	ofstream outFile("new.dat", ios::binary);
	file.seekg(0, ios::beg);
	bool flag = false;
	Student s1;

	while (file.read((char *)&s1, sizeof(Student)))
	{
		if (s1.getRollNo() == rollno)
		{
			flag = true;
			continue;
		}
		outFile.write((char *)&s1, sizeof(Student));
	}
	if (!flag)
	{
		cout << "\nRecord of " << rollno << " is not present.";
	}
	file.close();
	outFile.close();
	remove("student.dat");
	rename("new.dat", "student.dat");
	file.open("student.dat", ios::in | ios::out | ios::ate | ios::binary);
}
~FileOperations()
{
	file.close();
	cout << "\nFile Closed.\n";
```

```
        }
};
int main()
{
        cout << "\n----------SEQUENTIAL FILE----------\n";
        cout << "\n*****Student Database*****\n";
        ofstream newFile("student.dat", ios::app | ios::binary);
        newFile.close();
        FileOperations file((char *)"student.dat");
        int rollNo, year, choice = 0;
        char div;
        char name[MAX], address[MAX];
        while (choice != 6)
        {
                //clrscr();
                cout << "\n-----Menu-----\n";
                cout << "1) Add New Record\n";
                cout << "2) Display All Records\n";
                cout << "3) Display by RollNo\n";
                cout << "4) Updating a Record\n";
                cout << "5) Deleting a Record\n";
                cout << "6) Exit\n";
                cout << "\nChoose your choice : ";
                cin >> choice;
                switch (choice)
                {
                case 1: //New Record
                        cout << "\nEnter RollNo and Name : ";
                        cin >> rollNo >> name;
                        cout << "Enter Year and Division : ";
                        cin >> year >> div;
                        cout << "Enter Address : ";
                        cin >> address;
                        file.insertRecord(rollNo, name, year, div, address);
                        cout << "\nRecord Inserted.\n";
                        break;
                case 2:
                        cout << '\n';
                        cout << setw(5) << "ROLL" << setw(20) << "NAME" << setw(5) <<
"YEAR" << setw(5) << "DIV" << setw(10) << "CITY";
                        file.displayAll();
                        break;
                case 3:
                        cout << "\nEnter Roll Number : ";
                        cin >> rollNo;
                        cout << '\n';
                        cout << setw(5) << "ROLL" << setw(20) << "NAME" << setw(5) <<
"YEAR" << setw(5) << "DIV" << setw(10) << "CITY";
                        file.displayRecord(rollNo);
                        cout<<'\n';
                        break;
                case 4:
                        cout << "\nEnter Roll Number : ";
                        cin >> rollNo;
                        cout << '\n';
```

```
                        cout << setw(5) << "ROLL" << setw(20) << "NAME" << setw(5) <<
"YEAR" << setw(5) << "DIV" << setw(10) << "CITY";
                        file.displayRecord(rollNo);
                        cout<<'\n';
                        cout << "\nEnter new RollNo and Name : ";
                        cin >> rollNo >> name;
                        cout << "Enter new Year and Division : ";
                        cin >> year >> div;
                        cout << "Enter new Address : ";
                        cin >> address;
                        file.updateRecord(rollNo, name, year, div, address);
                        cout << "\nRecord Updated.\n";
                        break;
                case 5:
                        cout << "\nEnter Roll No : ";
                        cin >> rollNo;
                        file.deleteRecord(rollNo);
                        cout << "\nRecord Deleted Succesfully.\n";
                        break;
                case 6:
                        cout << "\nExitting Program!!!";
                        break;
                default:
                        cout << "\nWrong choice entered!!!\n";
                }
        }
        return 0;
}
```

Input : Enter the choice for the operation (e.g. : 1 or 2 ..etc)

Output:

```
----------SEQUENTIAL FILE----------

*****Student Database*****

-----Menu-----
1) Add New Record
2) Display All Records
3) Display by RollNo
4) Updating a Record
5) Deleting a Record
6) Exit

Choose your choice : 1

Enter RollNo and Name : 1 ABC
Enter Year and Division : 1 A
Enter Address : DEF

Record Inserted.

-----Menu-----
1) Add New Record
2) Display All Records
3) Display by RollNo
4) Updating a Record
5) Deleting a Record
6) Exit

Choose your choice : 1

Enter RollNo and Name : 2 GHI
Enter Year and Division : 2 B
Enter Address : JKL

Record Inserted.

-----Menu-----
1) Add New Record
2) Display All Records
3) Display by RollNo
4) Updating a Record
5) Deleting a Record
6) Exit

Choose your choice : 1

Enter RollNo and Name : 3 MNO
Enter Year and Division : 3 C
Enter Address : PQR

Record Inserted.

-----Menu-----
1) Add New Record
2) Display All Records
3) Display by RollNo
4) Updating a Record
5) Deleting a Record
6) Exit

Choose your choice : 2
```

```
 ROLL                NAME YEAR  DIV       CITY
    1                 ABC    1    A        DEF
    2                 GHI    2    B        JKL
    3                 MNO    3    C        PQR

-----Menu-----
1) Add New Record
2) Display All Records
3) Display by RollNo
4) Updating a Record
5) Deleting a Record
6) Exit

Choose your choice : 3

Enter Roll Number : 2

 ROLL                NAME YEAR  DIV       CITY
    2                 GHI    2    B        JKL

-----Menu-----
1) Add New Record
2) Display All Records
3) Display by RollNo
4) Updating a Record
5) Deleting a Record
6) Exit

Choose your choice : 4

Enter Roll Number : 3

 ROLL                NAME YEAR  DIV       CITY
    3                 MNO    3    C        PQR

Enter new RollNo and Name : 4 MNO
Enter new Year and Division : 4 A
Enter new Address : STU

Record Updated.

-----Menu-----
1) Add New Record
2) Display All Records
3) Display by RollNo
4) Updating a Record
5) Deleting a Record
6) Exit

Choose your choice : 2

 ROLL                NAME YEAR  DIV       CITY
    1                 ABC    1    A        DEF
    2                 GHI    2    B        JKL
    4                 MNO    4    A        STU

-----Menu-----
1) Add New Record
2) Display All Records
3) Display by RollNo
4) Updating a Record
5) Deleting a Record
6) Exit
```

```
Choose your choice : 5

Enter Roll No : 2

Record Deleted Succesfully.

-----Menu-----
1) Add New Record
2) Display All Records
3) Display by RollNo
4) Updating a Record
5) Deleting a Record
6) Exit

Choose your choice : 2

 ROLL                NAME YEAR  DIV      CITY
    1                 ABC    1    A       DEF
    4                 MNO    4    A       STU

-----Menu-----
1) Add New Record
2) Display All Records
3) Display by RollNo
4) Updating a Record
5) Deleting a Record
6) Exit

Choose your choice : 7

Wrong choice entered!!!

-----Menu-----
1) Add New Record
2) Display All Records
3) Display by RollNo
4) Updating a Record
5) Deleting a Record
6) Exit

Choose your choice : 6

Exitting Program!!!
File Closed.

[Program finished]
```

Conclusion: This program implements and performed different operations on student database using sequential file