ASSIGNMENT 9

1. **Aim** – Company maintains employees information as employee ID,name,designation and salary. Allow user to add,delete info. Of employee.Display info. Of particular employee.IF employee does not exist an appropriate message is displayed.Use index sequential data.

2. **Objective** – To implement the employee database and functions to add, delete student.

3. Theory –

File organization indicates how the records are organized in a file. There are different types of organizations for files so as to increase their efficiency of accessing the records. Following are the types of file organization schemes −

- Sequential file organization
- Indexed sequential file organization
- Relative file organization

Indexed Sequential File Organization

An indexed sequential file consists of records that can be accessed sequentially. Direct access is also possible. It consists of two parts −

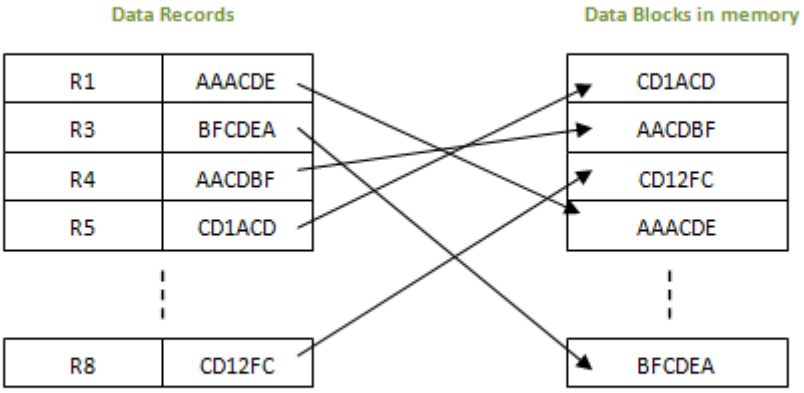
- **Data File** contains records in sequential scheme.
- **Index File** contains the primary key and its address in the data file.

Following are the key attributes of sequential file organization −

- Records can be read in sequential order just like in sequential file organization.
- Records can be accessed randomly if the primary key is known. Index file is used to get the address of a record and then the record is fetched from the data file.
- Sorted index is maintained in this file system which relates the key value to the position of the record in the file.
- Alternate index can also be created to fetch the records.

Indexed Sequential Access Method (ISAM)

This is an advanced sequential file organization method. Here records are stored in order of primary key in the file. Using the primary key, the records are sorted. For each primary key, an index value is generated and mapped with the record. This index is nothing but the address of record in the file.



In this method, if any record has to be retrieved, based on its index value, the data block address is fetched and the record is retrieved from memory.

4. **Algorithm** –

5. **Program code –**

```
#include<iostream>

#include<fstream>

#include<string.h>

using namespace std;


typedef struct EMP_REC

 {
```

```
 char name[10];
 int emp_id;
 int salary;
 char des[10];
 }Rec;

typedef struct INDEX_REC
 {
 int emp_id;
 int position;
 }Ind_Rec;

class Employee
{

 Rec Records;
 Ind_Rec Ind_Records;
 public:
 void Create();
 void Display();
 void Search();
 void deletion();
};

void Employee::Create()
{
```

```
char ch='y';
ofstream seqfile;
ofstream indexfile;
int i=0;
indexfile.open("IND.DAT",ios::out|ios::binary);
seqfile.open("EMP.DAT",ios::out|ios::binary);
do
{
 cout<<"\n Enter Name: ";
 cin>>Records.name;
 cout<<"\n Enter Emp_ID: ";
 cin>>Records.emp_id;
 cout<<"\n Designation";
 cin>>Records.des;
 cout<<"\n Enter Salary: ";
 cin>>Records.salary;
 cout<<Records.name<<" "<<Records.emp_id<<" "<<Records.salary;

 seqfile.write((char*)&Records,sizeof(Records));

 Ind_Records.emp_id=Records.emp_id;
 Ind_Records.position=i;
 indexfile.write((char*)&Ind_Records,sizeof(Ind_Records));
 i++;
 cout<<"\nDo you want to add more records?";
 cin>>ch;
```

```
  }while(ch=='y');
  seqfile.close();
  indexfile.close();
}

void Employee::Display()
{
 ifstream seqfile;
 ifstream indexfile;

 seqfile.open("EMP.DAT",ios::in|ios::binary);
 indexfile.open("IND.DAT",ios::in|ios::binary);
 cout<<"\n The Contents of file are ..."<<endl;
 int i=0;
 while(indexfile.read((char *)&Ind_Records,sizeof(Ind_Records)))
 {
  i=Ind_Records.position*sizeof(Rec);
  seqfile.seekg(i,ios::beg);
  seqfile.read((char *)&Records,sizeof(Records));
  if(Records.emp_id!=-1)
  {
  cout<<"\nName: "<<Records.name<<flush;
  cout<<"\nEmp_ID: "<<Records.emp_id;
  cout<<"\nDesignation :"<<Records.des;
  cout<<"\nSalary: "<<Records.salary;
  cout<<"\n";
```

```
    }


 }
 seqfile.close();
 indexfile.close();
}
void Employee::Search()
{
 fstream seqfile;
 fstream indexfile;
 int id,pos,offset;
 cout<<"\n Enter the Emp_ID for searching the record ";
 cin>>id;
 indexfile.open("IND.DAT",ios::in|ios::binary);
 pos=-1;
 while(indexfile.read((char *)&Ind_Records,sizeof(Ind_Records)))
 {
  if(id==Ind_Records.emp_id)
  {
   pos=Ind_Records.position;
   break;
  }
 }
  if(pos==-1)
  {
  cout<<"\n Record is not present in the file";
```

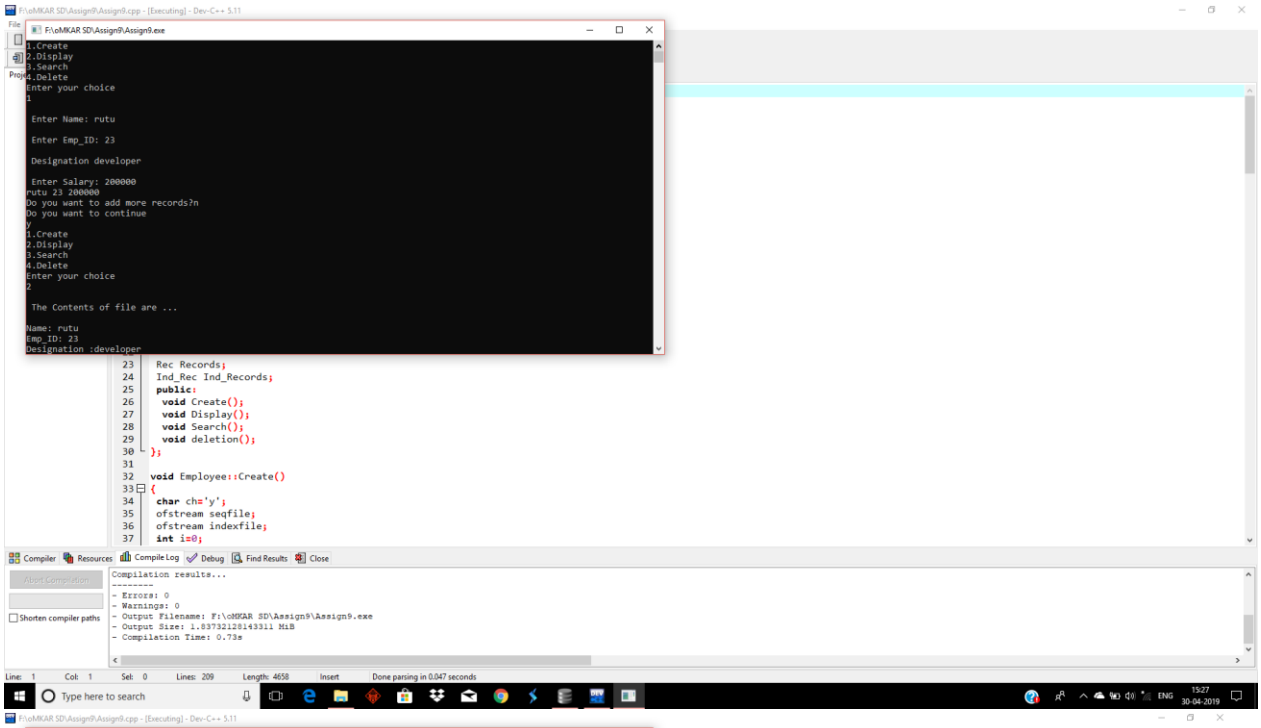
```
  return;
  }
  offset=pos*sizeof(Records);
  seqfile.open("EMP.DAT",ios::in|ios::binary);
  seqfile.seekg(offset,ios::beg);
  seqfile.read((char *)&Records,sizeof(Records));
  if(Records.emp_id==-1)
  {
  cout<<"\n Record is not present in the file";
  return;
  }
  else
  {
  cout<<"\n The Record is present in the file and it is...";
  cout<<"\n Name: "<<Records.name;
  cout<<"\n Emp_ID: "<<Records.emp_id;
  cout<<"\n Designation: "<<Records.des;
  cout<<"\n Salary: "<<Records.salary;
  }
  seqfile.close();
  indexfile.close();
}
void Employee::deletion()
{
    int id,pos;
    cout<<"For deletion"<<endl;
```

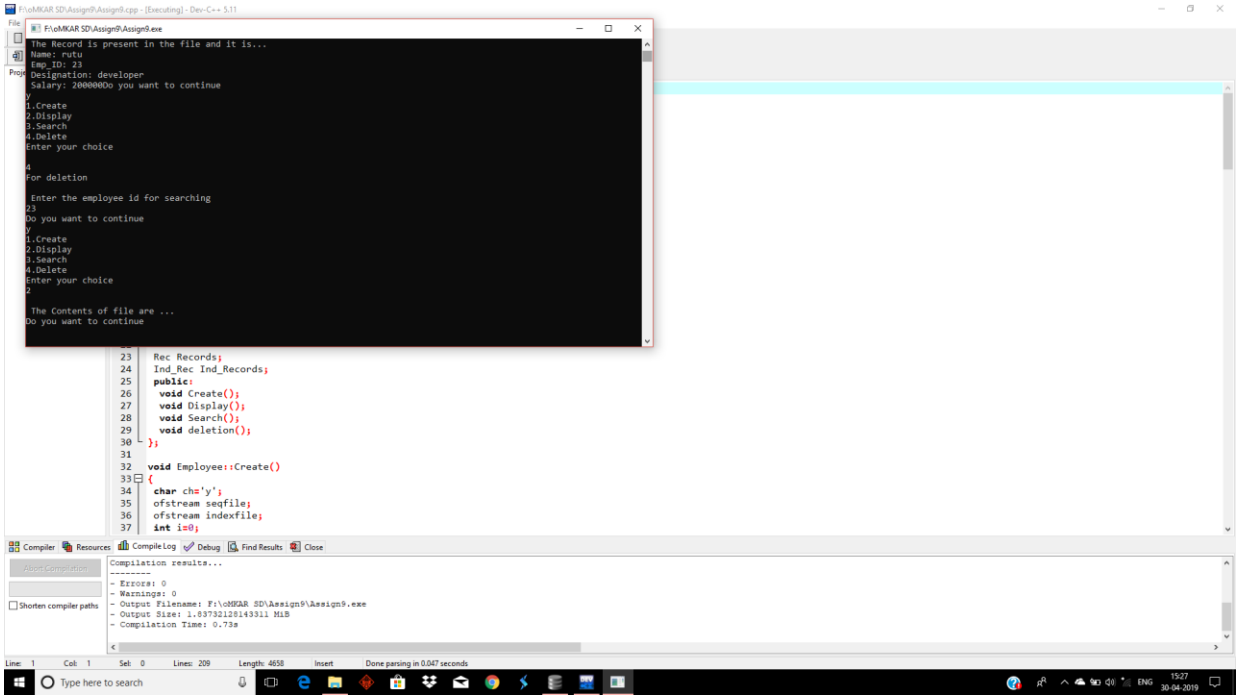
```
    cout<<"\n Enter the employee id for searching"<<endl;
    cin>>id;
    fstream seqfile;
    fstream indexfile;
  seqfile.open("EMP.DAT",ios::in|ios::binary|ios::out);
 indexfile.open("IND.DAT",ios::in|ios::binary|ios::out);
 seqfile.seekg(0,ios::beg);
 indexfile.seekg(0,ios::beg);
 pos=-1;
 while(indexfile.read((char *)&Ind_Records,sizeof(Ind_Records)))
 {
  if(id==Ind_Records.emp_id)
  {
   pos=Ind_Records.position;
   Ind_Records.emp_id=-1;
   break;
  }
 }
  if(pos==-1)
  {
  cout<<"\n Record is not present in the file";
  return;
  }
  int offset=pos*sizeof(Rec);
  seqfile.seekp(offset);
  strcpy(Records.name,"");
```

```
  Records.emp_id=-1;
  Records.salary=-1;
  strcpy(Records.des,"");
  seqfile.write((char *)&Records,sizeof(Records))<<flush;
  offset=pos*sizeof(Ind_Rec);
  indexfile.seekp(offset);
  Ind_Records.emp_id=-1;
  Ind_Records.position=pos;
  indexfile.write((char *)&Ind_Records,sizeof(Ind_Records));
  seqfile.seekg(0);
  indexfile.close();
  seqfile.close();
}
int main()
{
 Employee e;
 char ans='y';
 int choice,key;
 do
 {
    cout<<"1.Create"<<endl;
    cout<<"2.Display"<<endl;
    cout<<"3.Search"<<endl;
    cout<<"4.Delete"<<endl;
    cout<<"Enter your choice"<<endl;
    cin>>choice;
```

```
        switch(choice)
        {
        case 1:
          e.Create();
          break;
        case 2:
         e.Display();
         break;
        case 3:
           e.Search();
           break;
        case 4:
          e.deletion();
          break;
        }
        cout<<"Do you want to continue"<<endl;
        cin>>ans;
 }while (ans=='y');
return 0;
}
```

6. **Output screen shots/Copy Past From Terminal –**





7. **Conclusion –**

- Since each record has its data block address, searching for a record in larger database is easy and quick. There is no extra effort to search records. But proper primary key has to be selected to make ISAM efficient.
- This method gives flexibility of using any column as key field and index will be generated based on that. In addition to the primary key and its index, we can have index generated for other fields too. Hence searching becomes more efficient, if there is search based on columns other than primary key.