ASSIGNMENT NO.9.

Aim :- Company maintains employee information as employee ID, name, designation and salary. Allow user to add, delete information of employee. Display information of particular employee. If employee does not exist an appropriate message is displayed. If it is, then the system displays the employee details. Use index sequential file to maintain the data.

Objective:- to study use of different data structure concepts in this program.

Theory:-

Input/output formatting

Writing to or reading from a file is similar to writing onto a terminal screen or reading from a keyboard. Differences are:

- File must be opened with an OPEN statement, in which the unit number and (optionally) the filename are given
- Subsequent writes (or reads) must refer to a known unit number (used for open)
- File should be closed at the end

File opening and closing

The syntax is:

```
OPEN([unit=]lunit,file='name' [,options])

CLOSE([unit=]lunit [,options])
```

For example:

```
OPEN(10, file='output.dat', status='new')

CLOSE(unit=10)
```

- The first parameter is the unit number and the keyword unit= can be omitted.
- The unit numbers 0,5 and 6 are predefined.
  - 0 is output for standard (system) error messages
  - 5 is for standard (user) input

- 6 is for standard (user) output
- These units are opened by default and should not be re-opened nor closed by users

Some options for opening a file:

- status: existence of the file ('old', 'new', 'replace', 'scratch', 'unknown')
- position: offset, where to start writing ('append')
- action: file operation mode ('write','read','readwrite')
- form: text or binary file ('formatted', 'unformatted')
- access: direct or sequential file access ('direct','sequential','stream')
- iostat: error indicator, (output) integer (non zero only upon an error)
- err: the label number to jump upon error
- recl: record length, (input) integer for direct access files only. Be careful, it can be in bytes or words...

Algorithm:-

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;
  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);
 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";
```

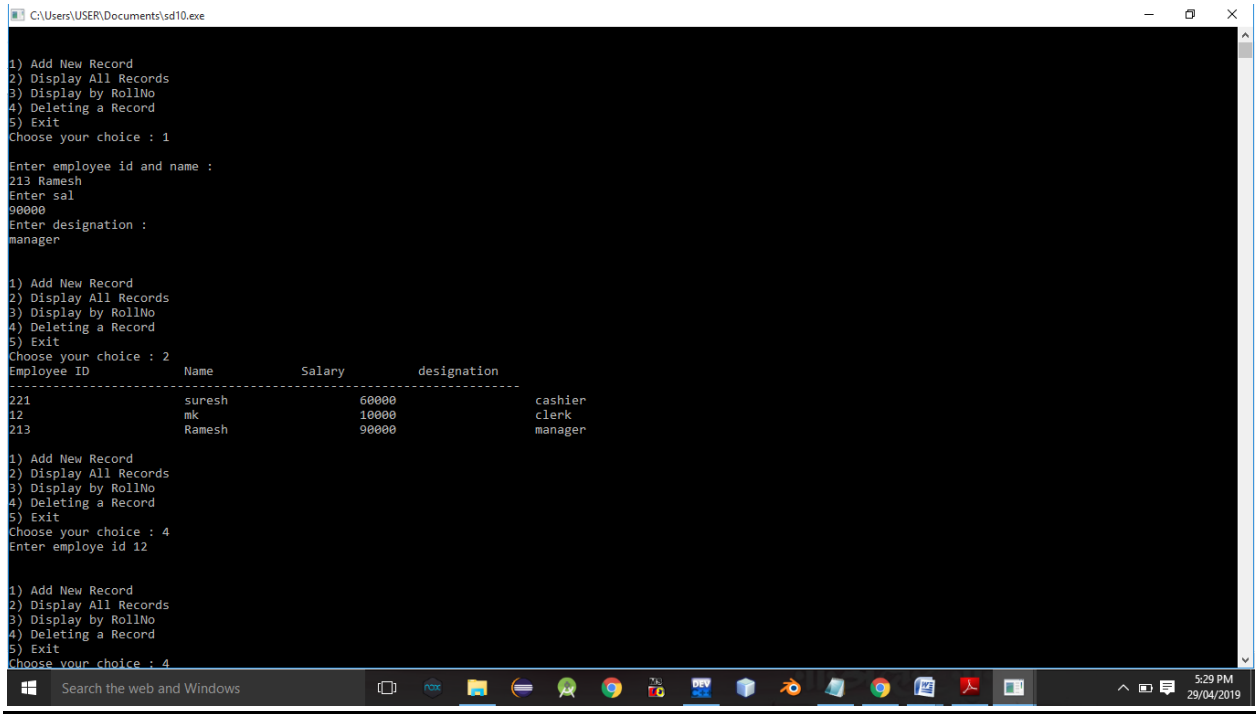
```
    }
 }
 seqfile.close();
 indexfile.close();
}
void Employee::Search()
{
 fstream seqfile;
 fstream indexfile;
 int id,pos,offset;
 cout<<"\n Enter the Emp_ID for searching the record ";
 cin>>id;
 indexfile.open("IND.DAT",ios::in|ios::binary);
 pos=-1;
 while(indexfile.read((char *)&Ind_Records,sizeof(Ind_Records)))
 {
  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";
  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 found";
  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<<"\tMENU";
     cout<<"1.Enter info"<<endl;
     cout<<"2.Display info"<<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;
}
```

Output Screenshots:-



Conclusion:- Thus, this assignment is completed successfully.