



NAME : ANKIT MISHRA

CLASS : XII-A1

SCHOOL : MPVM

YEAR : 2014-15

Certified to be the bonafide work done by

Mr. **ANKIT MISHRA** of class **XII – A1**

during the year **2014-2015**

Date **\_\_\_\_\_\_** in the subject of **COMPUTER SCIENCE**.

Date:

Registration No. :

Signature of Internal Signature of External

Examiner: Examiner:

I would like to express my sincere gratitude to my Computer

science mentor **Mr. Shwetank Kakkar ,** for his vital

support, guidance and encouragement - without which this

project would not have come forth.

**TOPIC PAGE NO.**

* **Certificate 2**
* **Acknowledgement 3**
* **Introduction To C++ 6-8**
* **Telephone Directory 10-12**

**management**

* **Resource Requirements 13**
* **Coding 15-48**
* **Screen shots 50-54**
* **Bibliography 55**

**INTRODUCTION**

**TO C++**

**INTRODUCTION TO C++**

**Data File Handling:**

**File.** The information / data stored under a specific name on a storage device, is called a file.

**Stream.** It refers to a sequence of bytes.

**Text file.** It is a file that stores information in ASCII characters. In text files, each line of text is terminated with a special character known as EOL (End of Line) character or delimiter character. When this EOL character is read or written, certain internal translations take place.

**Binary file.** It is a file that contains information in the same format as it is held in memory. In binary files, no delimiters are used for a line and no translations occur here.

## Classes For File Stream Operation:

**ofstream**: Stream class to write on files  
**ifstream**: Stream class to read from files  
**fstream:** Stream class to both read and write from/to files.

### Input And Output Operation:

**put() and get() function**  
the function put() writes a single character to the associated stream. Similarly, the function get() reads a single character form the associated stream.  
example :  
file.get(ch);  
file.put(ch);

**write() and read() function**  
write() and read() functions write and read blocks of binary data.

internal stream pointers that point to the reading or writing locations within a stream can be manipulated using the following member functions:

|  |  |
| --- | --- |
| seekg() | moves get pointer(input) to a specified location |
| seekp() | moves put pointer (output) to a specified location |
| tellg() | gives the current position of the get pointer |
| tellp() | gives the current position of the put pointer. |

|  |  |
| --- | --- |
| **File mode parameter** | **Meaning** |
| ios::app | Append to end of file |
| ios::ate | go to end of file on opening |
| ios::binary | file open in binary mode |
| ios::in | open file for reading only |
| ios::out | open file for writing only |
| ios::nocreate | open fails if the file does not exist |
| ios::noreplace | open fails if the file already exist |
| ios::trunc | delete the contents of the file if it exist. |
|  |  |

**Linked Lists:**

A linked list is a technique of creating a list with the ability to add, delete, or retrieve items. Additional operations can also be provided to a more elaborate list such as finding an item, deleting an item, etc.

**The Beginning of a Linked List:**

If you create an [array-based list](http://www.functionx.com/cpp/articles/arraybasedlist.htm), you can start by declaring an array member variable that would hold the items and each item can be located by an index that is assigned to it when the item is added to the list. When  you do this, you must provide an estimate of the maximum number of items that will be allowed in the list. Without good planning, the dimension you specify could be too high or too low but C++ doesn't allow you to declare an array without a dimension if you are not initializing the array. This means that, when you create an array-based list, you must also specify the maximum number of items that the list can hold.

A linked list is a list that can grow or shrink as the [user wishes](http://www.functionx.com/cpp/articles/linkedlist.htm). This means that, when creating the list, you don't need to predict the maximum number of items that will be added to the list. To use this flexibility, the items must be managed through pointers. Because the list would use a member that defines its item, you can declare a member variable that is conform to the intended items.

When a list starts, it is empty or at least it must be considered like that, before any item is added to it. To specify this, you should declare a primary member variable. Although you can call it anything, it is usually called Head. This member can be made private if you don't intend to access it outside of the class. If you want clients of the class to access it, you can make it public. Although this member is declared as a pointer and it marks the beginning of the list, you should not allocate memory for it in the constructor. Its memory would be managed when it is accessed. Therefore, you can simply initialize is as NULL.

Once a list exists, the user can explorer it. One of the operations performed on items is to locate and retrieve one. To do this, you can declare a method that takes as argument as index. The method would examine the argument with regards to the number of items in the list to make sure the argument's value is in the range of [current items](http://www.functionx.com/cpp/articles/linkedlist.htm) of the list. If the number is too low or too high, the method can return null or 0. If the number is in the range, the method can return the item at that position.

One of the operations hardly performed on a list is to find one. This is because if you ask a list to locate a particular item, you must provide as much information as possible. Probably the most expedient way you can do this is to completely define an item and pass it to the list. Only if the item is found in the list would it be recognized.

**TELEPHONE**

**DIRECTORY**

**MANAGEMENT**

**TELEPHONE DIRECTORY MANAGEMENT**

The telephone directory maintenance system is designed to insure a smooth allotment of telephone to the employee of a particular department and query the telephone directory whenever required.

**CURRENT SYSTEM**

Currently telephones are been allotted to employees at random on a first come first served basis. There is no proper tracking of who is allotted which telephone number. Any number of telephones gets allotted to the employees.

LIMITATION OF THE CURRENT SYSTEM

Because of the current system of allotment of telephone numbers to the employees at random, the same telephone number at a time gets allotted to more than one employee. There is no proper record of how many phones are allotted to each department which results in tracing of employees being difficult. There is no mechanism to find out which department a particular employee belongs to.

**PROPOSED SYSTEM**

The proposed system is being designed to maintain the details of employees address,pincode,telephone numbers and all their details through an application in a file. Unique record id are generated for any new employee/new company and new telephone number.

**OBJECTIVES:**

The system design will meet the below objectives

* Unique number generation for the new employee/new telephone allotment.
* Easy retrieval of telephone details based on employee names.
* Easy retrieval of telephone details based on the record id number.
* Deletion of the record id deletes all the information belonging to that employee.
* Employee location details can be modified whenever the employee gets transferred.

**SCOPE AND BOUNDARY:**

The project is designed for

* Record maintenance
  + Add records
  + Delete records
* Employee Maintenance
  + Employee addition
  + Employee modification
* Telephone Directory maintenance
  + Add telephone number
  + Enquiry by record id
  + Enquiry by Employee name
  + Enquiry by telephone no.

**LIMITATIONS OF THE PROJECT**

There are several limitations of the project. The project is subdivided into three sub-modules.

1. Record maintenance system.
2. Employee maintenance system.
3. Telephone directory maintenance system.

There are several limitations in the module. Some of they are:

* User can store maximum of 2500 employee records.
* All the constraints are not checked properly.

**Future Scope:**

The future scope of this project is immense. If the small difficulties and problems can be eliminated, the software will be of great help in the real life scenario.

**Resource Requirement** :

* **Software Used** : 1. Windows XP

2. Turbo C++

3. Notepad

4. MS Word

* **Hardware Used**: 1. Pentium III onwards

2. 128 MB RAM

3. 40GB HDD

**CODING**

CODING :

#include<fstream.h>

#include<conio.h>

#include<stdio.h>

#include<string.h>

#include<iomanip.h>

#include<dos.h>

struct node

{

unsigned long recno;

long int pin;

char name[20];

node \*link; // Dynamic memory allocation

};

// Nodes are declared for sorting the phone.dat data file according

to recno,pin and name respectively.

node \*first,\*ptr1,\*ptr2,\*ptr,\*temp;

//Declaration of class for maintaining the data of customer

class person

{

private:

unsigned long record\_id;

char person\_name[20];

char res\_address[30];

long int res\_pincode;

char res\_phone\_no1[20];

char occu\_pation[10];

char company\_name[20];

char off\_address[20];

long int off\_pincode;

char off\_phone\_no1[20];

public:

person()

{

record\_id=0;

res\_pincode=off\_pincode=0;

res\_phone\_no1;

off\_phone\_no1;

}

// Generating automatic record number for customer

unsigned long get\_rec\_no();

// Function to input data records for customers

void get\_data();

// Displaying individual customers telephone and address information

void show\_data();

// Appends new customer information into data file

void add\_object();

// Extracts the data records from file to linked list

void make\_index();

// Arranges database acc. to name

void sort\_name();

// Arranges database acc. to recno

void sort\_recno();

// Arranges database acc. to pin no.

void sort\_pin();

void show\_object();

// Prints the sorted record on screen

void sort\_print();

void del\_object();

// Modify the customers information in database

void modify\_object();

void search\_object();

void sort\_object();

void del\_all();

void reports();

}pers; // Declaring global person variable

// Declaration for the get\_rec\_no()

unsigned long person::get\_rec\_no()

{

int found=0;

unsigned long rec\_no,temp\_recno;

struct node1

{

unsigned long recno;

node1 \*link;

};

node1 \*start,\*ptr,\*ptr1,\*ptr2;

fstream infile;

infile.open("phoneno.dat",ios::in);

infile.seekg(0,ios::end);

//tellg() returns the current stream position

int n=infile.tellg();

infile.close();

if(n==0)

rec\_no=1;

else

{

infile.open("phoneno.dat",ios::in);

start=ptr=new node1;

infile.seekg(0,ios::beg);

infile.read((char\*)&pers,sizeof(pers));

while(!infile.eof())

{

ptr->recno=record\_id;

ptr->link=new node1;

ptr=ptr->link;

infile.read((char\*)&pers,sizeof(pers));

}

ptr->link=NULL;

ptr1=start;

while(ptr1->link!=NULL)

{

ptr2=ptr1->link;

while(ptr2!=NULL)

{

if(ptr2->recno<ptr1 ->recno)

{

temp\_recno=ptr2->recno;

ptr2->recno=ptr1->recno;

ptr1->recno=temp\_recno;

}

ptr2=ptr2->link;

}

ptr2=ptr1->link;

ptr1=ptr2;

}

ptr1=start;

while(ptr1!=NULL && found!=1)

{

ptr2=ptr1;

ptr1=ptr1->link;

if((ptr2->recno)+1!=ptr1->recno)

{

rec\_no=(ptr2->recno)+1;

found=1;

}

}

if(found!=1)rec\_no=(ptr2->recno)+1;

ptr=start;

while(start!=NULL)

{

start=start->link;

delete ptr;

}

}

return rec\_no;

}

void person::del\_all()

{

clrscr();

fstream file1,file2;

file1.open("phoneno.dat",ios::out||ios::trunc);

remove("phoneno.dat");

file2.open("temp.dat",ios::in);

file1.close();

file2.close();

rename("temp.dat","phone.dat");

}

// Function definition of get\_data()

void person::get\_data()

{

clrscr();

// Calling function to get record no. automatically

record\_id=pers.get\_rec\_no();

gotoxy(5,7);

cout<<"RECORD\_ID :"<<pers.record\_id;

cin.get();

gotoxy(5,8);

cout<<"ENTER THE NAME--------------------->";

cin.getline(person\_name,20);

cin.get();

gotoxy(5,9);

cout<<"RESIDENTIAL ADDRESS---------------->";

cin.getline(res\_address,30);

cin.get();

gotoxy(5,10);

cout<<"ENTER RESIDENTIAL PINCODE---------->";

cin>>res\_pincode;

cin.get();

gotoxy(5,11);

cout<<"ENTER RESIDENTIAL PHONE NO--------->";

cin.getline(res\_phone\_no1,20);

cin.get();

gotoxy(5,12);

cout<<"ENTER THE OCCUPATION--------------->";

cin.getline(occu\_pation,10);

cin.get();

gotoxy(5,13);

cout<<"ENTER COMPANY'S NAME--------------->";

cin.getline(company\_name,20);

cin.get();

gotoxy(5,14);

cout<<"ENTER COMPANY'S ADDRESS------------>";

cin.getline(off\_address,20);

cin.get();

gotoxy(5,15);

cout<<"ENTER OFFICE PINCODE--------------->";

cin>>off\_pincode;

cin.get();

gotoxy(5,16);

cout<<"ENTER OFFICE PHONE NO-------------->";

cin.getline(off\_phone\_no1,20);

cin.get();

gotoxy(5,17);

clrscr();

}

void person::show\_data()

{

clrscr();

gotoxy(5,2);

cout<<"RECORD NUMBER----------------------->"<<record\_id;

gotoxy(5,3);

cout<<"NAME-------------------------------->"<<person\_name;

gotoxy(5,4);

cout<<"RESIDENTIAL ADDERSS----------------->"<<res\_address;

gotoxy(5,5);

cout<<"RESIDENTIAL PINCODE----------------->"<<res\_pincode;

gotoxy(5,6);

cout<<"RESIDENTIAL PHONE NO---------------->"<<res\_phone\_no1;

gotoxy(5,7);

cout<<"OCCUPATION-------------------------->"<<occu\_pation;

gotoxy(5,8);

cout<<"COMPANY's NAME---------------------->"<<company\_name;

gotoxy(5,9);

cout<<"COMPANY's ADDRESS------------------->"<<off\_address;

gotoxy(5,10);

cout<<"OFFICE PINCODE---------------------->"<<off\_pincode;

gotoxy(5,11);

cout<<"OFFICE PHONE NO--------------------->"<<off\_phone\_no1;

gotoxy(10,20);

getch();

}

//This function creates a master file .

void person::add\_object()

{

fstream file;

char choice='y';

while(choice=='y')

{

file.open("phoneno.dat",ios::app);

get\_data();

file.write((char\*)&pers,sizeof(pers));

file.flush();

file.close();

clrscr();

gotoxy(10,11);

cout<<"ANY MORE RECORDS TO BE READ<<Y/N>>-------------->";

cin>>choice;

}

}

void person::del\_object()

{

fstream infile,outfile;

unsigned long rec\_no;

clrscr();

cout<<"ENTER THE RECORD ID.NO.TO BE DELETED--------->";

cin>>rec\_no;

infile.open("phoneno.dat",ios::in);

infile.seekg(0);

outfile.open("tempno.dat",ios::app);

infile.read((char\*)&pers,sizeof(pers));

while(!infile.eof())

{

if(pers.record\_id!=rec\_no)

outfile.write((char\*)&pers,sizeof(pers));

else

{

show\_data();

}

outfile.flush();

infile.read((char\*)&pers,sizeof(pers));

}

gotoxy(20,20);

cout<<"Want to Delete the record(y/n)..."<<endl;

gotoxy(55,20);

char ans;

cin>>ans;

if(ans=='n'|| ans=='N')

{

return;

}

infile.close();

outfile.close();

remove("phoneno.dat");

rename("tempno.dat","phoneno.dat");

}

void person::search\_object()

{

fstream infile;

int search\_choice;

unsigned long rec\_no;

char phno[20];

char name[20];

do

{

clrscr();

int counter=0;

gotoxy(22,7);

cout<<" SEARCH MENU ";

gotoxy(22,9);

cout<<"------------------";

gotoxy(22,11);

cout<<"RECORD ID NO. ..1";

gotoxy(22,12);

cout<<"NAME ..2";

gotoxy(22,13);

cout<<"PHONE NO. ..3";

gotoxy(22,14);

cout<<"EXIT SEARCH MENU ..4";

gotoxy(1,20);

cout<<"ENTER THE FIELD CHOICE ACCORDING TO WHICH A RECORD";

cout<<"IS TO BE SEARCHED--------->";

cin>>search\_choice;

switch(search\_choice)

{

case 1:clrscr();

cout<<"\n ENTER THE RECORD ID.NO. TO BE SEARCHED----->";

cin>>rec\_no;

infile.open("phoneno.dat",ios::in);

infile.seekg(0,ios::beg);

infile.read((char\*)&pers,sizeof(pers));

while(!infile.eof())

{

if(pers.record\_id==rec\_no)

{

counter++;

pers.show\_data();

}

infile.read((char\*)&pers,sizeof(pers));

}

infile.close();

gotoxy(20,24);

cout<<"RECORDS FOUND="<<counter;

getch();

break;

case 2:

clrscr();

cout<<"\n ENTER THE NAME TO BE SEARCHED---------->";

gets(name);

infile.open("phoneno.dat",ios::in);

infile.seekg(0,ios::beg);

infile.read((char\*)&pers,sizeof(pers));

while(!infile.eof())

{

if(strcmpi(pers.person\_name,name)==0)

{

counter++;

pers.show\_data();

}

infile.read((char\*)&pers,sizeof(pers));

}

infile.close();

gotoxy(20,24);

cout<<"RECORDS FOUND="<<counter;

getch();

break;

case 3:

clrscr();

cout<<"\nENTER THE PHONE NO.TO BE SEARCHED-------->";

cin.getline(phno,20);

cin.get();

infile.open("phoneno.dat",ios::in);

infile.seekg(0,ios::beg);

infile.read((char\*)&pers,sizeof(pers));

while(!infile.eof())

{

if(pers.res\_phone\_no1==phno||pers.off\_phone\_no1==phno)

{

counter++;

pers.show\_data();

}

infile.read((char\*)&pers,sizeof(pers));

}

infile.close();

gotoxy(20,24);

cout<<"RECORDS FOUND="<<counter;

getch();

break;

case 4:

clrscr();

gotoxy(22,15);

cout<<"YOU HAVE ENDED THE SEARCH SESSION\n";

gotoxy(27,18);

cout<<"\tTHANK YOU!";

delay(700);

break;

}

}

while(search\_choice!=4);

}

void person::modify\_object()

{

fstream file;

unsigned long code;

int modify\_choice;

do

{

clrscr();

gotoxy(22,7);

cout<<" MODIFY MENU ";

gotoxy(22,8);

cout<<" ---------------";

gotoxy(22,10);

cout<<"RESIDENTIAL INFORMATON ..1";

gotoxy(22,11);

cout<<"OFFICIAL INFORMATION ..2";

gotoxy(22,12);

cout<<"EXIT ..3";

gotoxy(22,13);

cout<<"SELECT THE INFORMATION TO BE CHANGED : ";

gotoxy(22,14);

cout<<"ENTER YOUR CHOICE----->";

cin>>modify\_choice;

if(modify\_choice!=3)

{

clrscr();

gotoxy(10,15);

cout<<"ENTER THE RECORD NO. OF THE PERSON---->";

cin>>code;

file.open("phoneno.dat",ios::in|ios::out);

file.seekg(0,ios::beg);

file.read((char\*)&pers,sizeof(pers));

int n=file.tellg();

while(!file.eof())

{

if(pers.record\_id==code)

{

switch(modify\_choice)

{

case 1:

clrscr();

cout<<"\*\*\*\*\* The Details of "<<pers.person\_name;

cin.get();

cout<<"\n\nThe old Address is : "<<pers.res\_address;

cin.get();

cout<<"\nENTER NEW RESIDENTIAL ADDRESS-------->";

cin.getline(pers.res\_address,30);

cin.get();

cout<<"The old Pin code is : "<<pers.res\_pincode;

cout<<"\nENTER NEW PINCODE-------------------->";

cin>>pers.res\_pincode;

cout<<"The old Phone No.is : "<<pers.res\_phone\_no1;

cout<<"\nENTER NEW RESIDENTIAL PHONE NO------->";

cin>>pers.res\_phone\_no1;

file.seekg(n-sizeof(pers));

file.write((char\*)&pers,sizeof(pers));

file.flush();

break;

case 2:

clrscr();

cin.get();

cout<<"\nENTER THE OCCUPATION----------->";

cin.getline(pers.occu\_pation,10);

cin.get();

cout<<"\nENTER NEW COMPANY'S NAME------->";

cin.getline(pers.company\_name,20);

cin.get();

cout<<"\nENTER NEW OFFICE ADDRESS------->";

cin.getline(pers.off\_address,20);

cin.get();

cout<<"\nENTER NEW OFFICE PIN CODE------>";

cin>>pers.off\_pincode;

cin.get();

cout<<"\nENTER NEW OFFICE PHONE NO------>";

cin>>pers.off\_phone\_no1;

file.seekg(n-sizeof(pers));

file.write((char\*)&pers,sizeof(pers));

file.flush();

break;

}

}

file.read((char\*)&pers,sizeof(pers));

n=file.tellg();

}

file.close();

}

}

while(modify\_choice!=3);

clrscr();

gotoxy(22,10);

cout<<"YOU ENDED MODIFY INFORMATION SESSION\n";

gotoxy(30,13);

cout<<"\tTHANK YOU";

delay(700);

}

void ex\_change()

{

temp=new node;

temp->link=NULL;

temp->recno=ptr2->recno;

temp->pin=ptr2->pin;

strcpy(temp->name,ptr2->name);

ptr2->recno=ptr1->recno;

ptr2->pin=ptr1->pin;

strcpy(ptr2->name,ptr1->name);

ptr1->recno=temp->recno;

ptr1->pin=temp->pin;

strcpy(ptr1->name,temp->name);

delete temp;

}

void person::make\_index()

{

fstream infile;

first=new node;

ptr=first;

infile.open("phoneno.dat",ios::in);

infile.seekg(0,ios::beg);

infile.read((char\*)&pers,sizeof(pers));

while(!infile.eof())

{

ptr->recno=record\_id;

ptr->pin=res\_pincode;

strcpy(ptr->name,pers.person\_name);

ptr->link=new node;

ptr=ptr->link;

infile.read((char\*)&pers,sizeof(pers));

}

ptr->link=NULL;

infile.close();

}

void person::sort\_name()

{

pers.make\_index();

ptr1=first;

while(ptr1->link!=NULL)

{

ptr2=ptr1->link;

while(ptr2->link!=NULL)

{

if(strcmpi(ptr2->name,ptr1->name)<0)

ex\_change();

ptr2=ptr2->link;

}

ptr2=ptr1->link;

ptr1=ptr2;

}

}

void person::sort\_recno()

{

pers.make\_index();

ptr1=first;

while(ptr1->link!=NULL)

{

ptr2=ptr->link;

while(ptr2!=NULL)

{

if(ptr2->recno<ptr1 ->recno)

ex\_change();

ptr2=ptr2->link;

}

ptr2=ptr1->link;

ptr1=ptr2;

}

}

void person::sort\_pin()

{

pers.make\_index();

ptr1=first;

while(ptr1->link!=NULL)

{

ptr2=ptr1->link;

while(ptr2!=NULL)

{

if(ptr2->pin<ptr1 ->pin)

ex\_change();

ptr2=ptr2->link;

}

ptr2=ptr1->link;

ptr1=ptr2;

}

}

void person::sort\_print()

{

fstream infile;

ptr=first;

while(ptr!=NULL)

{

infile.open("phoneno.dat",ios::in);

infile.seekg(0,ios::beg);

infile.read((char\*)&pers,sizeof(pers));

while(!infile.eof())

{

if(ptr->recno==pers.record\_id)

{

pers.show\_data();

infile.seekg(0,ios::end);

}

infile.read((char\*)&pers,sizeof(pers));

}

infile.close();

ptr=ptr->link;

}

}

void del\_index()

{

while(first!=NULL)

{

ptr=first;

first=first->link;

delete ptr;

}

}

void person::sort\_object()

{

int sort\_choice;

do

{

clrscr();

gotoxy(22,7);

cout<<" SORT MENU ";

gotoxy(22,8);

cout<<"-------------------";

gotoxy(22,10);

cout<<"SORTED RECORDS NOS. ..1";

gotoxy(22,11);

cout<<"SORTED NAMES ..2";

gotoxy(22,12);

cout<<"SORTED PINCODES NOS. ..3";

gotoxy(22,13);

cout<<"EXIT SORT MENU ..4";

gotoxy(22,20);

cout<<"ENTER YOUR CHOICE NO.----->";

cin>>sort\_choice;

switch(sort\_choice)

{

case 1:

clrscr();

pers.sort\_recno();

pers.sort\_print();

del\_index();

delay(200);

break;

case 2:

clrscr();

pers.sort\_name();

pers.sort\_print();

del\_index();

delay(200);

break;

case 3:

clrscr();

pers.sort\_pin();

pers.sort\_print();

del\_index();

delay(200);

break;

case 4:

clrscr();

gotoxy(22,10);

cout<<"YOU ENDED THE SORTED SESSION\n";

gotoxy(27,13);

cout<<"\tTHANK YOU!";

delay(700);

break;

}

}

while(sort\_choice!=4);

}

void person::reports()

{

fstream infile;

int report\_choice;

do

{

clrscr();

gotoxy(22,7);

cout<<" REPORT MENU ";

gotoxy(22,8);

cout<<"------------------";

gotoxy(5,10);

cout<<"SORTED LIST OF NAMES WITH RESIDENCE AND OFFICE PHONE NOS. ..1";

gotoxy(5,11);

cout<<"LIST OF NAMES WITH THIER OFFICE DETAILS. ..2";

gotoxy(5,12);

cout<<"LIST OF NAMES WITH THEIR RECORD NOS.AND RESIDENTIAL ADDRESS ..3";

gotoxy(5,13);

cout<<"EXIT REPORT SESSION ..4";

gotoxy(5,14);

cout<<"ENTER YOUR CHOICE NO.------>";

cin>>report\_choice;

switch(report\_choice)

{

case 1:

clrscr();

{

cout<<"\n NAME RESIDENCE PHONE NO. OFFICE PHONE NO.";

cout<<"\n---------------------------------------------------------------------";

pers.sort\_name();

ptr=first;

while(ptr!=NULL)

{

infile.open("phoneno.dat",ios::in);

infile.seekg(0,ios::beg);

infile.read((char\*)&pers,sizeof(pers));

while(!infile.eof())

{

if(ptr->recno==pers.record\_id)

{

cout<<"\n";

cout<<setiosflags(ios::left)

<<setw(28)<<pers.person\_name<<setw(30)

<<pers.res\_phone\_no1<<setw(30)

<<pers.off\_phone\_no1;

infile.seekg(0,ios::end);

}

infile.read((char\*)&pers,sizeof(pers));

}

infile.close();

ptr=ptr->link;

}

del\_index();

gotoxy(20,20);

cout<<"PRESS ANY KEY TO CONTINUE";

getch();

break;

}

case 2:

clrscr();

infile.open("phoneno.dat",ios::in);

infile.seekg(0,ios::beg);

infile.read((char \*)&pers,sizeof(pers));

while(!infile.eof())

{

gotoxy(10,5);

cout<<"NAME------------------->"<<pers.person\_name;

gotoxy(10,6);

cout<<"COMPANY'S NAME--------->"<<pers.company\_name;

gotoxy(10,7);

cout<<"COMPANY'S ADDRESS------>"<<pers.off\_address;

gotoxy(10,8);

cout<<"OFFICE PHONE NO.------->"<<pers.off\_phone\_no1;

gotoxy(10,10);

cout<<"PRESS ANY KEY TO CONTINUE";

getch();

infile.read((char \*)&pers,sizeof(pers));

clrscr();

}

infile.close();

break;

case 3:

clrscr();

cout<<"\nREC NO. NAME RES.ADDRESS RES PHONE NO.";

cout<<"\n-------------------------------------------------------------";

infile.open("phoneno.dat",ios::in);

infile.seekg(0,ios::beg);

infile.read((char\*)&pers,sizeof(pers));

while(!infile.eof())

{

cout<<"\n";

cout<<setiosflags(ios::left)

<<setw(15)<<pers.record\_id

<<setw(15)<<pers.person\_name<<

setw(18)<<pers.res\_address<<

setw(35)<<res\_phone\_no1;

cout<<setw(55)<<" ";

infile.read((char\*)&pers,sizeof(pers));

}

infile.close();

gotoxy(20,40);

cout<<"\n PRESS ANY KEY TO CONTINUE";

getch();

break;

case 4:

clrscr();

gotoxy(22,10);

cout<<"YOU HAVE ENDED REPORT SESSION\n";

gotoxy(27,13);

cout<<"\tTHANK YOU";

delay(700);

break;

}

}

while(report\_choice!=4);

}

void main()

{

int main\_choice;

do

{

clrscr();

gotoxy(15,3);

cout<<" WELCOME TO TELEPHONE DIRECTORY ";

gotoxy(16,4);

cout<<" ================================== ";

gotoxy(19,6);

cout<<" MAIN MENU ";

gotoxy(21,7);

cout<<" ============================= ";

gotoxy(25,11);

cout<<" ADD A NEW RECORD ..1";

gotoxy(25,12);

cout<<" DELETE A RECORD ..2";

gotoxy(25,13);

cout<<" MODIFY A RECORD ..3";

gotoxy(25,14);

cout<<" SEARCH A RECORD ..4";

gotoxy(25,15);

cout<<" SORTED INFORMATION ..5";

gotoxy(25,16);

cout<<" REOPORTS ..6";

gotoxy(25,17);

cout<<" DELETE ALL RECORDS ..7";

gotoxy(25,18);

cout<<" EXIT ..8";

gotoxy(25,20);

cout<<"ENTER YOUR CHOICE NO.--------->";

cin>>main\_choice;

switch(main\_choice)

{

case 1:

pers.add\_object();

break;

case 2:

pers.del\_object();

break;

case 3:

pers.modify\_object();

break;

case 4:

pers.search\_object();

break;

case 5:

pers.sort\_object();

break;

case 6:

pers.reports();

break;

case 7:

pers.del\_all();

cout<<"\n\n\t\tALL RECORDS DELETED";

cout<<"\n\n\t\tPress Any Key To Continue.....";

getch();

break;

case 8:

clrscr();

gotoxy(22,10);

cout<<"YOU HAVE ENDED THE SESSION\n";

gotoxy(27,13);

cout<<"\tTHANK YOU";

delay(1000);

break;

}

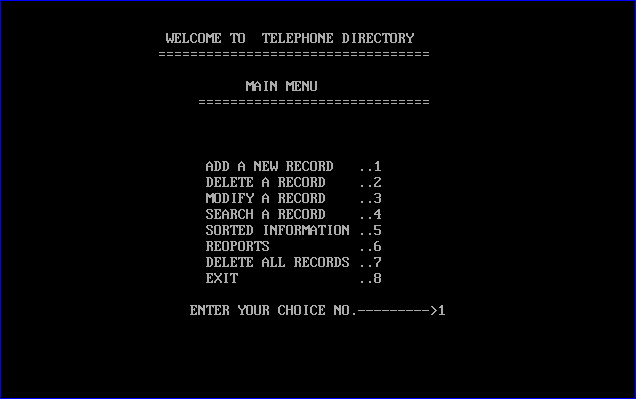
}

while(main\_choice!=8);

}

**SCREEN SHOTS**

Screen Shots:



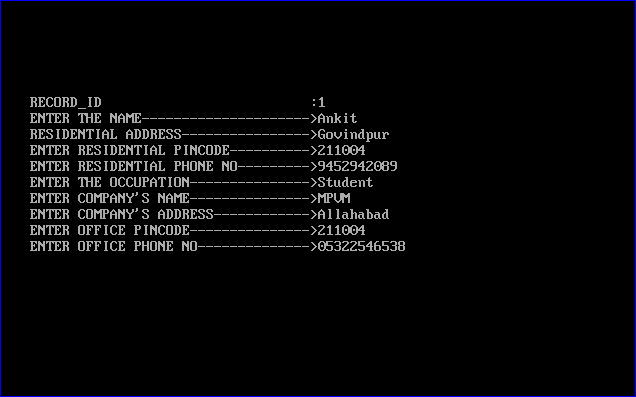
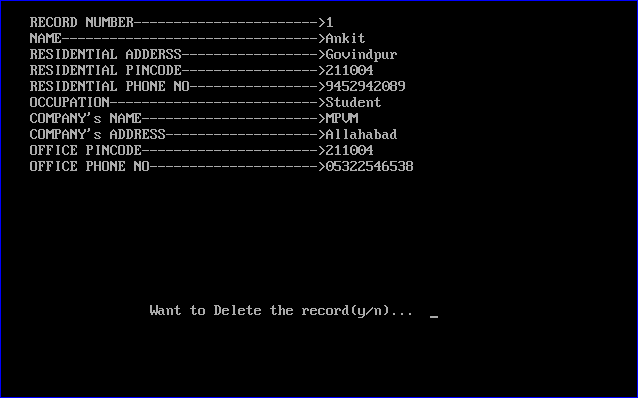
 Fig.1 Main Menu

Fig.2 Fig.2 Adding Record



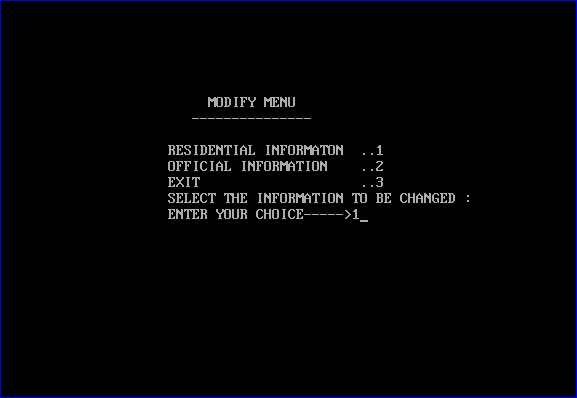
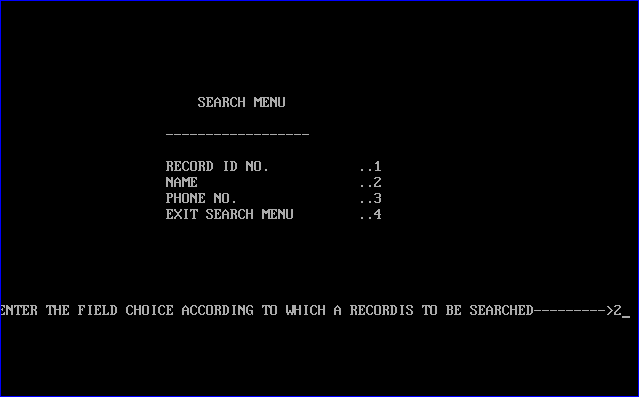
 Fig.3 Deleting a record

Fig.4 Modify Menu



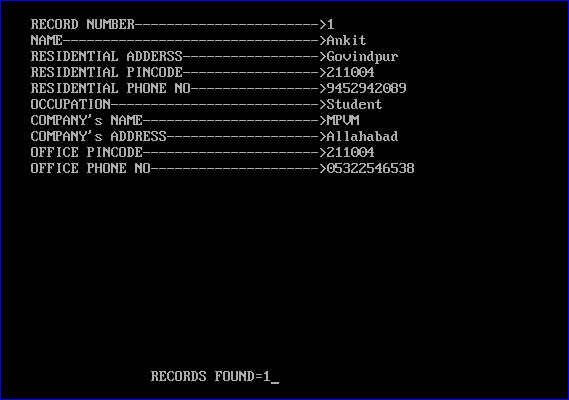
 Fig.5 Search Menu

Fig.6 Fig.6 Record Found

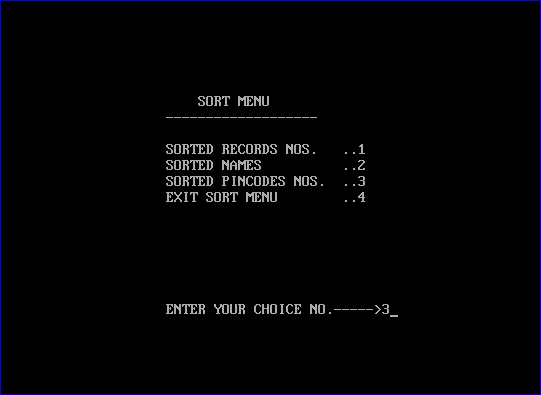


Fig.7 Fig.7 Sort Menu

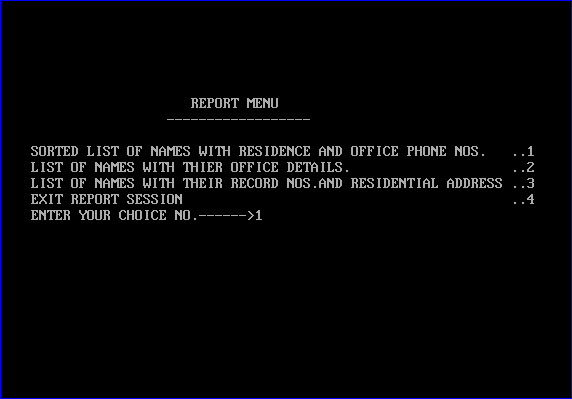


Fig.8 Report Menu

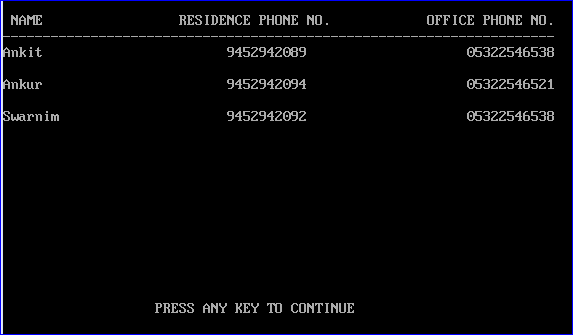


 Fig.9 Report of Name, Res. Phno. And Office Phno

Fig.10 Report of Recno.,Name,Res. Add. and Res. Phno