

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“Jnana Sangama”, Belagavi-590018



A File structures Mini Project Report on “Telephone Directory Management System”

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR
THE AWARD OF DEGREE OF

BACHELOR OF ENGINEERING

IN

INFORMATION SCIENCE AND ENGINEERING

SUBMITTED BY

HARSHITHA H SHETTY (1JB19IS035)

JYOTHI PS (1JB19IS037)

Under the Guidance of

Mrs. SRIDEVI G M

Assistant Professor,
Dept. of ISE, SJBIT
Bengaluru-60



DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

SJB INSTITUTE OF TECHNOLOGY

BGS HEALTH AND EDUCATION CITY,
Kengeri, Bengaluru-560060, KARNATAKA, INDIA.

2021 – 2022

|| Jai Sri Gurudev ||
Sri Adichunchanagiri Shikshana Trust ®
SJB INSTITUTE OF TECHNOLOGY
BGS Health & Education City, Kengeri, Bengaluru – 560 060

Department of Information Science & Engineering



CERTIFICATE

Certified that the Mini-project work entitled “**Telephone Directory Management System**”, is a bonafide work carried out by **HARSHITHA H SHETTY(1JB19IS035)** and **JYOTHI PS(1JB19IS037)**, a bonifide students of **SJB Institute of Technology**, in partial fulfilment for 6th semester in **INFORMATION SCIENCE AND ENGINEERING** of the **VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELAGAVI** during the academic year **2021-22**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The mini project report has been approved as it satisfies the academic requirements in respect of mini project prescribed for the said degree.

Mrs. SRIDEVI G M
Asst. Professor
Dept. of ISE, SJBIT

Dr. REKHA B
Professor & Head
Dept. of ISE, SJBIT

EXTERNAL VIVA

NAME OF THE EXAMINERS

SIGNATURE WITH DATE

1. _____

2. _____



ACKNOWLEDGEMENT

We would like to express our profound gratitude to His Divine Soul **Padmabhushan Sri Sri Sri Dr. Balagangadharanatha MahaSwamiji** and His Holiness **Jagadguru Sri Sri Sri Dr. Nirmalanandanatha MahaSwamiji** for providing us an opportunity to complete our academics in this esteemed institution.

We would also like to express our profound thanks to **Revered Sri Sri Dr. Prakashnath Swamiji, Managing Director**, SJB Institute of Technology, for his continuous support in providing amenities to carry out this project in this admired institution.

We express our gratitude to **Dr. K V Mahendra Prashanth, Principal**, SJB Institute of Technology, for providing us an excellent facilities and academic ambience; which have helped us in satisfactory completion of project work.

We extend our sincere thanks to **Dr. Rekha B, Professor & Head**, Department of Information Science and Engineering for providing us an invaluable support throughout the period of our project work.

We wish to express our heartfelt gratitude to the mini-project coordinator, **Mrs.Sridevi G M**, Assistant Professor, Department of Information Science and Engineering for her valuable guidance, suggestions and cheerful encouragement during the entire period of our project work.

Finally, We take this opportunity to extend our earnest gratitude and respect to our parents, Teaching & Non-teaching staff of the department, and all our friends, who have directly or indirectly supported us during the period of our project work.

HARSHITHA H SHETTY(1JB19IS035)
JYOTHI PS(1JB19IS037)

ABSTRACT

The main aim and objective was to plan and program system application and to get rid of manual entry and to store it in a file so that easily available . We have to apply the best software engineering practice for system application. We developed “Telephone Directory Management system” where the software Visual Studio for C++ is used and perform basic operations like insertion, deletion, display and search can be performed. The Telephone Directory Management System can handle all the details about a Person. The user can collect People information by adding, searching, removing and searching for details. This mini project contains limited features, but essential one. This details includes Telephone id, name of the person, Phone number and City. It tracks all the details of a person.

Table of Contents

Sl. No.	Chapters	Page No.
	Acknowledgement	i
	Abstract	ii
	Table of Contents	iii
	List of Figures	iv
1	Introduction	1
	1.1 File Structure	
	1.2 Overview of project	
2	System Requirements Specification	6
	2.1 Requirements	
	2.2 Development Environment	
3	System Design	8
	3.1 Operations performed on a file	
	3.2 File structure used in project	
4	Implementation	13
	4.1 Console window	
	4.2 Code snippets	
5	Results	21
	5.1 Main menu	
	5.2 Indexed File	
	Conclusion and Future Enhancements	28
	References	29

List of Figures

Sl.No.	Particular	Page No.
Figure 1.1	Trees	2
Figure 1.2	Example for AVL Trees	2
Figure 1.3	Example for B Tree	3
Figure 1.4	Example for B+ Tree	3
Figure 1.5	Extensible Hashing	4
Figure 3.1	Hashing	9
Figure 3.2	Inserting in Hash table	10
Figure 3.3	Searching in Hash table	11
Figure 3.4	Deleting in Hash table	11
Figure 5.1	Main Menu	21
Figure 5.2	Insertion of a record	22
Figure 5.3	Searching a record	23
Figure 5.4	Displaying records	24
Figure 5.5	Updating a record	25
Figure 5.6	Deletion of a record	26
Figure 5.7	Indexed file	27

Chapter 1

INTRODUCTION

We have developed Telephone Directory Management System to get rid from manual entry and store it in a file which can be easily available. Here we perform various operations like inserting a record, deleting a record, search and displaying a record. For this purpose, need the software which could easily and conveniently maintain the telephone directory details. The record of series can be stored in a single file.

1.1 File Structure

A file structure is a combination of representations for data in files and of operations for accessing the data. A file structure allows applications to read, write, and modify data. It might also support finding the data that matches some search criteria or reading through the data in some particular order. An improvement in file structure design may make an application hundreds of times faster. The details of the representation of the data and the implementation of the operations determine the efficiency of the file structure for particular applications.

1.1.1 Short History of File Structure Design

General goals of research and development in file structures:

- To find the target information with as few access as possible (i.e., 2 or 3 accesses).
- To get the information we need with one access to the disk.
- To group information to get everything with only one access.
- Trees -in the early 1960s, the idea of applying tree structures emerged as a potential solution. Unfortunately, trees can grow very unevenly as records are added and deleted, resulting in long searches requiring many disk accesses to find a record.

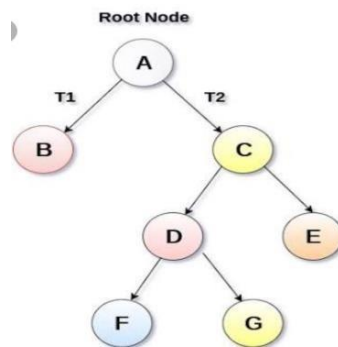


Figure 1.1 Trees

- Sequence access -> Direct access
Tape -> Disk.
Index : <key, pointer> in smaller file.
- As files grew intolerably large for unaided sequential access and as storage devices such as disk drives became available, indexes are added to files. Indexes made it possible to keep a list of keys and pointers in a smaller file that could be searched more quickly. With the key and pointer, user had direct access to large primary file.
- AVL tree is a self -adjusting binary tree for data in memory. Other researchers began to look for ways to apply AVL trees, or something like them to files.

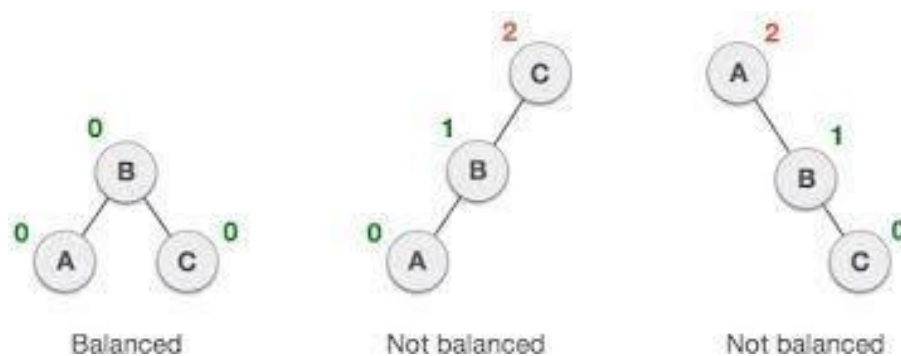


Figure 1.2 Examples for AVL tree

- It took nearly 10 years of design work before a solution emerged in the form of the B- tree. B-tree is a balanced tree structure and provide excellent access performance, but sequential access with a cost.

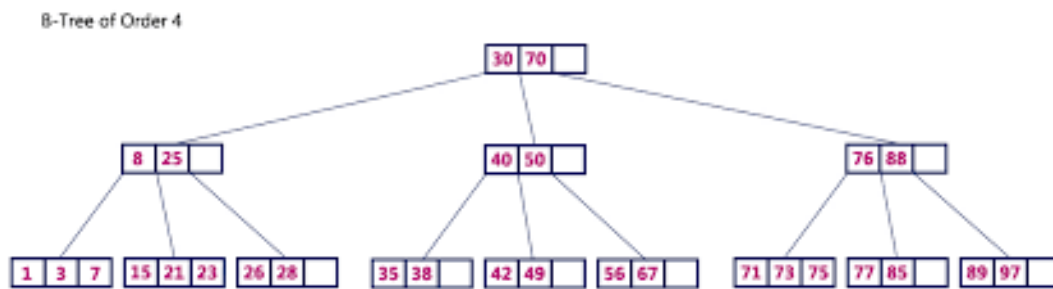


Figure 1.3 Examples for B tree

- The combination of a B-tree and sequential linked list is called a B+tree. Over the next 10 years, B-trees and B+ trees became the basis for many commercial file systems

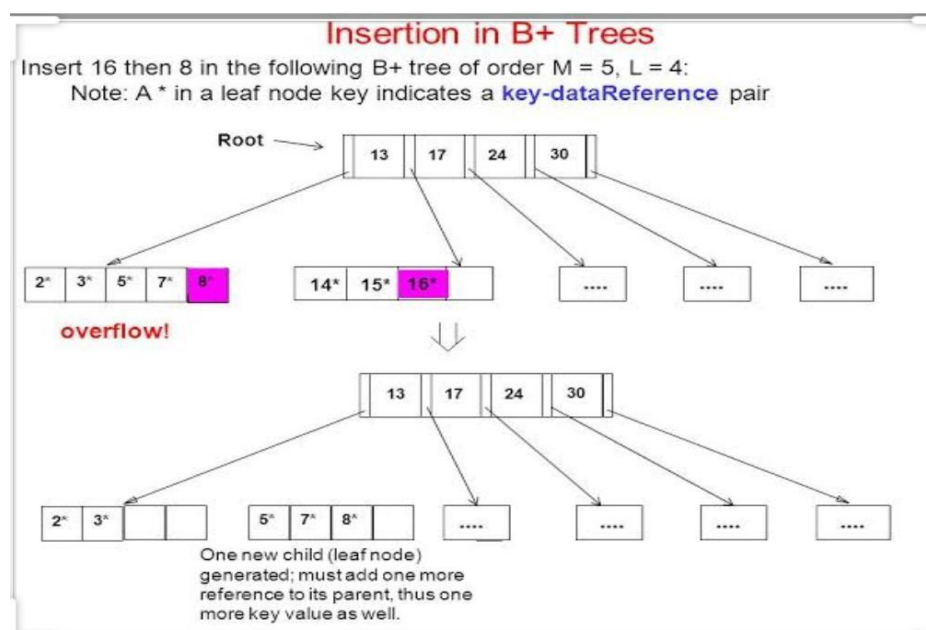


Figure 1.4 Example of B+ tree

- An approach called hashing is a good way to do that with the files that do not change size greatly over time. From early on, hashed indexes were used to provide fast access to files.

- After the development of B-trees, researchers turned to work on systems for extendible, dynamic hashing that could retrieve information with one ,or at most ,two disk access no matter how big the file became.

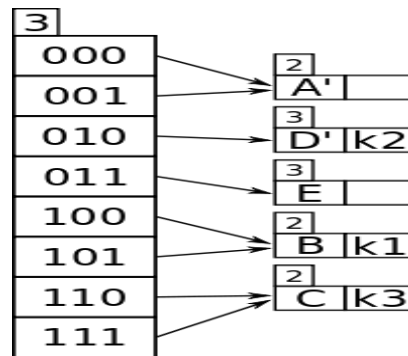


Figure 1.5 Extensible hashing

1.1.2 Application of File Structure

Relative to other parts of a computer, disks are slow. 1 can pack thousands of megabytes on a disk that fits into a notebook computer. The disk access is a quarter of a million times longer than a memory access. Hence, disks are very slow compared to memory. On the other hand, disks provide enormous capacity at much less cost than memory. They also keep the information stored on them when they are turned off.

Tension between a disk's relatively slow access time and its enormous, non-volatile capacity, is the driving force behind file structure design. Good file structure design will give us access to all the capacity without making our applications spend a lot of time waiting for the disk.

1.2 Overview of Project

A telephone directory is one of the most useful tools of information in communication. The traditional hard copy prints of the yellow and white pages have been in existence since 1878, and in late 20th century telephone address books went online.

1.2.1 Problem Statement

In present system all work is done on papers manually. The member's details are stored in a register. The validity has to be calculated manually. There is no means to effectively keep track of subscription validity of the members.

Disadvantages of present working system:

- The data storage is relatively difficult in papers, files and registers. The retrieval of any data
- like phone number is time consuming.
- Possibility of loss of data or any other circumstances.

Updating of new data like city is not possible.

1.2.2 Objective of Project

- To ease process of storing contact details.
- To improve Existing System.
- To develop scalable system
- To make process of updation easier

The software product “Telephone Directory management System” will be an application that will be Used for maintaining the records in an organized manner and to replace old paper work system. This project aims at automating the telephone management details for smooth working of the database by automating almost all the activities. Updations and modifications will be easily achievable.

Chapter 2

SYSTEM REQUIREMENTS SPECIFICATION

A computerized way of handling information about user details is efficient, organized and time saving, compared to a manual way of doing so. This is done through a menu driven console-based application whose requirements are mentioned in this section.

2.1 Requirements

The requirements include Hardware and software requirements for the project

2.1.1 Hardware Requirements

Minimum RAM : 2GB

Processor : Intel core processor 3 and above

Operating System : Windows 10 and above

2.1.2 Software Requirements

Language : C++

Software : Turbo C++

2.2 Development Environment

Turbo C++ is the software used for the development of Telephone Directory Management System. The source code is in the programming language C++.

2.2.1 C++

C++ is a multi-paradigm programming language that supports object-oriented programming (OOP), created by Bjarne Stroustrup in 1983 at Bell Labs, C++ is an extension(superset) of C programming and the programs are written in C language can run in C++ compilers. An interpreter is a computer program that directly executes, i.e. performs, instructions written in a programming or scripting language, without requiring them previously to have been compiled into a machine language program.

2.2.2 Features

- Simple : Every C++ program can be written in simple English language so that it is very easy
- Platform dependent : A language is said to be platform dependent whenever the program is executing in the same operating system where that was developed and compiled but not run and execute on another operating system. C++ is platform dependent language.
- Portability : It is the concept of carrying the instruction from one system to another system. In C++ Language. Cpp file contain source code, we can edit also this code. .exe file contain application, only we can execute this file. When we write and compile any C++ program on window operating system that program easily run on other window-based system.
- Powerful : C++ is a very powerful programming language, it has a wide variety of data types,
- Object oriented Programming language : This main advantage of C++ is; it is object-oriented programming language. It follows concept of oops like polymorphism, inheritance, encapsulation, abstraction.
- Case sensitive : C++ is a case sensitive programming language.
- Compiler based : Without compilation no C++ program can be executed.

2.2.3 Turbo C++

Turbo C++ is a discontinued C++ compiler and integrated development environment and computer language originally from Borland. Most recently it was distributed by Embarcadero Technologies, which acquired all of Borland's compiler tools with the purchase of its Code Gear division in 2008. The original Turbo C++ product line was put on hold after 1994 and was revived in 2006 as an introductory-level IDE, essentially a stripped down version of their flagship C++Builder. Turbo C++ 2006 was released on September 5, 2006 and was available in 'Explorer' and 'Professional' editions. The Explorer edition was free to download and distribute while the Professional edition was a commercial product. In October 2009 Embarcadero Technologies discontinued support of its 2006 C++ editions.

Chapter 3

SYSTEM DESIGN

The purpose of the design phase is to develop a clear understanding of what the developer wants people to gain from his/her project. As the developer works on the project, the test for every design decision should be

"Does this feature fulfil the ultimate purpose of the project?"

A purpose statement affects the design process by explaining what the developer wants the project to do, rather than describing the project itself. The Design Document will verify that the current design meets all of the explicit requirements contained in the system model as well as the implicit requirements desired by the customer.

3.1 Operations Performed on a File

3.1.1 Insertion

The system is initially used to add products containing product id, name and price into the file. Records with duplicate product id fields are not allowed to be inserted. The length of the product name is checked to see whether it contains less than 50 characters.

3.1.2 Display

The system can then be used to display existing records. The records are displayed based on the way we inserted. It contains product id, name and price of the product.

3.1.3 Search

The system can then be used to search for existing records. The user is prompted for a productID. If product ID is matched then entire details of the existing record will be displayed. If product ID does not exist it displays the message saying record not found.

3.1.4 Delete

The system can then be used to delete existing records. The user can delete specific item or all items in his cart. The user is prompted for a product ID, which is needed to be deleted. The requested record, if found is cleared, a "record deleted" message is displayed, and the record

of the respective product ID will also be deleted in the file (The key of the record will be replaced by *) . If absent record not found message will be displayed.

3.2 File Structure Used in Project

The file structure used in the Telephone Directory Management System is Hashing.

3.2.1 Hashing

Hashing is the process of transforming any given key or a string of [characters](#) into another value. This is usually represented by a shorter, fixed-length value or key that represents and makes it easier to find or employ original string.

- With Hashing addresses generated appear to be random there is no immediately obvious connections between key and location of record .Hence it is sometimes referred as randomizing.
- With Hashing, two different keys may be transformed to same address so two records may be sent to same place in file. This is called Collision.

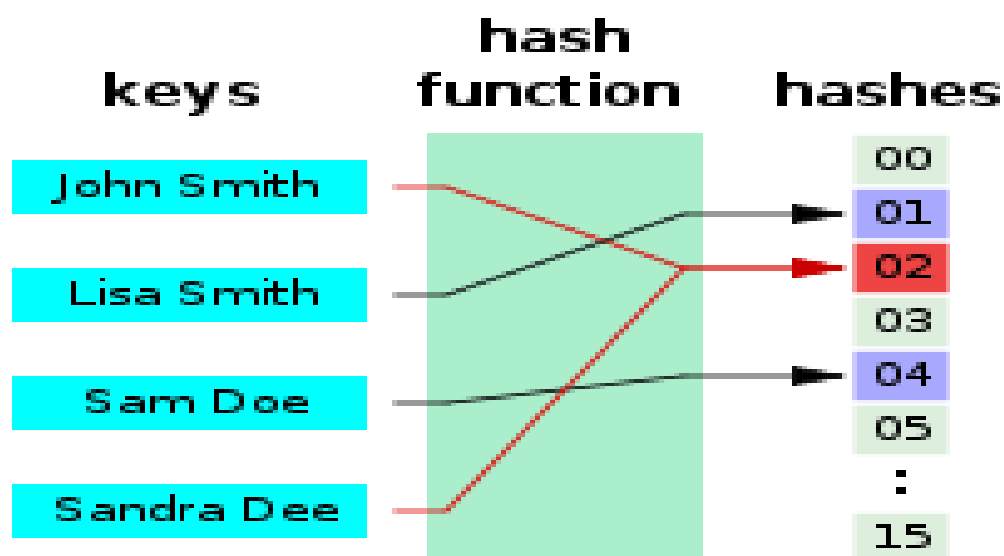


Figure 3.1 Hashing

How to calculate the hash key?

Let's take hash table size as 7.

size = 7

arr[size];

Formula to calculate key is,

key = element % size

Inserting elements in the hash table

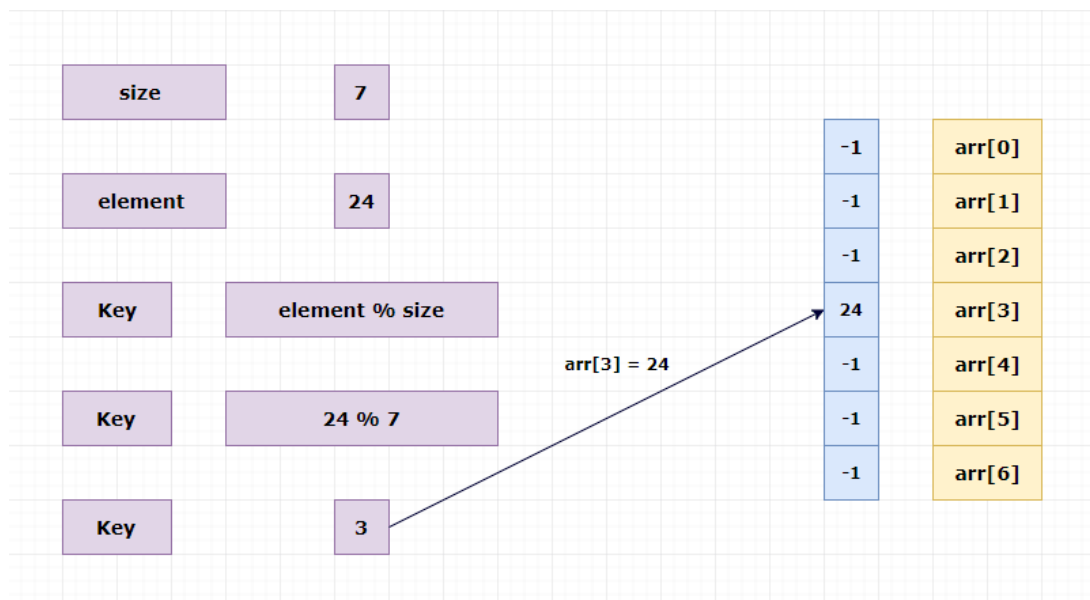


Figure 3.2 Inserting in Hash Table

Searching elements from the hash table

i)search 8

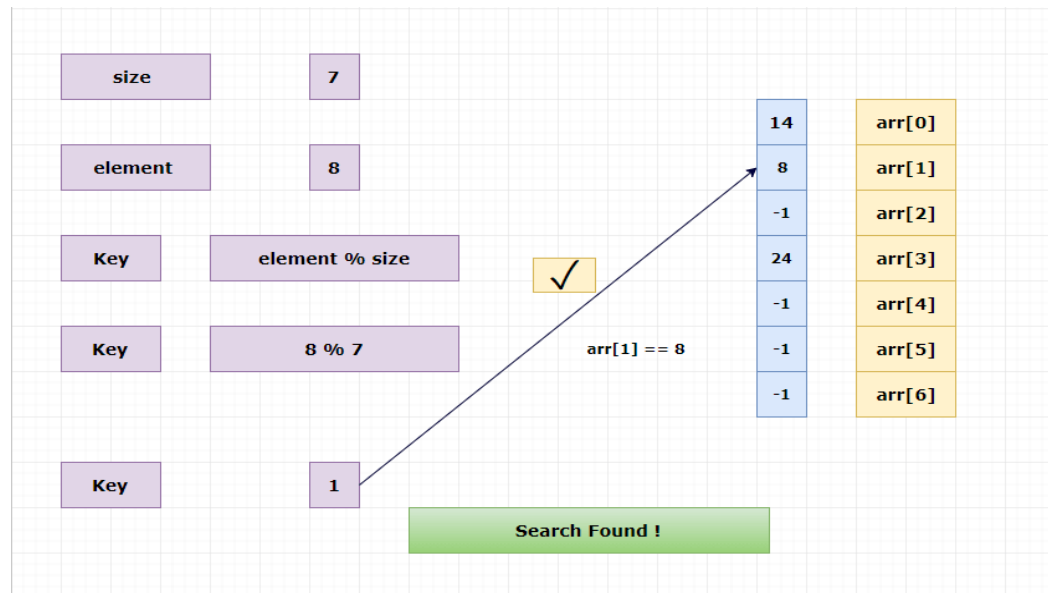


Figure 3.3 Searching in hash Table

Deleting an element from the hash table

Here we are not going to remove any element . we just mark the index as -1.

Example

Delete: 24

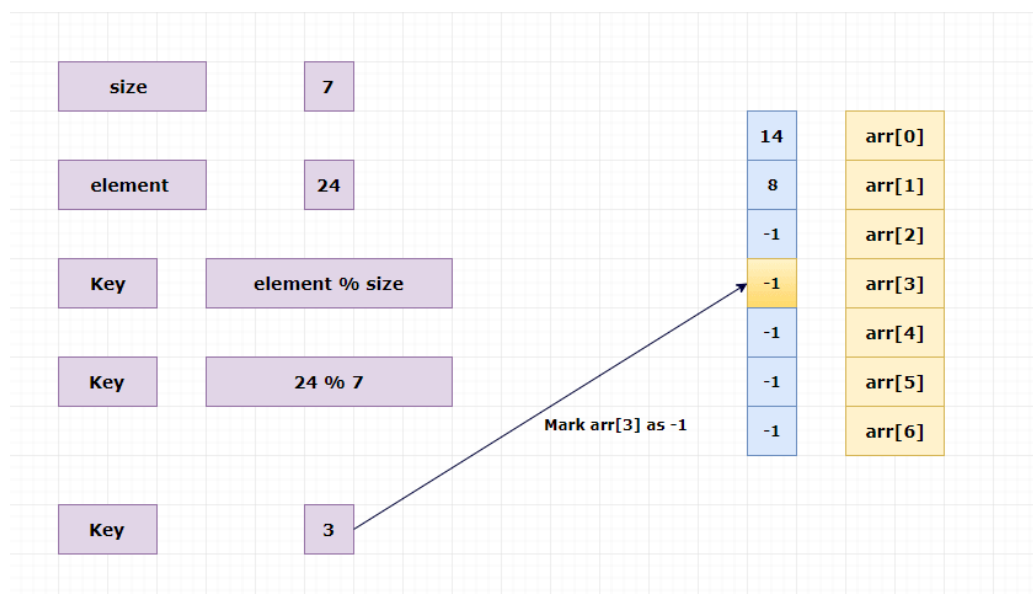


Figure 3.4 Deleting in Hash Table

3.2.2 Collisioin

In hashing technique, Collison is a situation when hash value of two key becomes similar.

Collision resolution techniques are

- 1) Open Addressing
 - a. Linear Probing
 - b. Quadratic Probing
 - c. Double Hashing Technique
- 2) Closed Addressing
 - a. Chaining

Chapter 4

IMPLEMENTATION

Implementation is the stage in the project where the theoretical design is turned into a working system and is giving confidence on the new system for the users that it will work efficiently and effectively. It involves careful planning, investigation of the current system and its constraints on implementation, design of methods to achieve the changeover, an evaluation of change over methods. Implementation is the most important phase. The most critical stage in achieving a successful new system is giving the users confidence that the new system will work and be effective. Any system developed should be secured and protected against possible hazards.

Component testing is a method where testing of each component in an application is done separately. Component testing is also known as module, unit or program testing. It finds the defects in the module and verifies the functioning of software. Our project is console-based, so all the implementation of our project is in console.

4.1 Console Window

The console window consists of 6 options which will direct to particular tasks.

The options in main screen

- Add the Telephone record
- Search in the Telephone record
- Display all the records
- Delete an record
- Modify an record
- Quit program

4.2 Code Snippets

4.2.1 Add a record

```
void student::read()
{
    Clrscr ();
    int i;
    gotoxy(20,2); for(i=0;i<50;i++)
    cout<<"-";
    gotoxy(35,4);
    cout<<"ADD THE RECORD";
    gotoxy(20,6); for(i=0;i<50;i++)
    cout<<"-";
    gotoxy(20,8);
    cout<<"ENTER TELEPHONE DETAILS:"<<endl;
    cout<<"\tEnter the TEL_id:(Tel_)"<<endl;
    gets(eid);
    cout<<"\tEnter the name of the Person:"<<endl;
    gets(ename);
    cout<<"\tEnter the Phone number:"<<endl;
    gets(toc);
    cout<<"\tEnter the City:"<<endl;
    gets(rfee);

    strcpy(buffer,eid);    strcat(buffer,"|");
    strcat(buffer,ename);  strcat(buffer,"|");
    strcat(buffer,toc);    strcat(buffer,"|");
    strcat(buffer,rfee);    strcat(buffer,"|");
    return;
}

int hash(char key[])
{
    int i=0,sum=0;
    while(key[i]!='\0')
```

```
{  
    sum=sum+(key[i]);  
    i++;  
}  
return sum % max;  
}
```

4.2.2 Display All Records

```
void student::display()  
{  
    clrscr();  
    chardummy[85];    file.open(studentfile,ios::in|ios::out);    cout<<setiosflags(ios::left);  
    cout<<"\t\t\t\t DIRECTORY DETAILS"<<endl;  
    cout<<" -----  
    ----- "<<endl;  
    cout<<"\t"<<setw(10)<<"TEL_ID"<<setw(20)<<"NAME"<<setw(15)<<"PHONE  
    NUMBER"<<setw(15)<<"CITY"<<endl;  
    cout<<" -----  
    ----- "<<endl;  
  
    while(1)  
    {  
        file.getline(eid,15,'|'); file.getline(ename,20,'|');file.getline(toc,15,'|');  
        file.getline(rfee,15,'|'); file.getline(dummy,85,'\n');if(file.eof())  
            break; if(eid[0]!='#')  
            {  
                cout<<"\t"<<setw(10)<<eid<<setw(20)<<ename<<setw(15)<<toc<<setw(15)<<rfee<<"\n  
                ";  
            } }  
  
    file.close();  
  
    getch(); }
```

4.2.3 Search a Record

```
void student::retrieve(int addr,char k[])
{
    int found=0,i;
    char dummy[10];i=addr;
    file.open(studentfile,ios::in|ios::out);

    do {
        file.seekg(i*recsize,ios::beg);
        file.getline(dummy,5,'\n');
        if(strcmp(dummy,"####")==0)
            break;
        file.seekg(i*recsize,ios::beg);
        file.getline(eid,15,'|');
        if(strcmp(eid,k)==0)
        {
            found=1; textcolor(RED);cout<<"\n";
            gotoxy(20,12);
            cout<<"RECORD FOUND!!\n";
            cout<<endl;
            file.getline(ename,20,'|');
            file.getline(toc,15,'|');
            file.getline(rfee,15,'|');
            gotoxy(20,14);
            cout<<"TEL_ID:"<<eid<<endl;gotoxy(20,15); cout<<"NAME:"<<ename; gotoxy(20,16);
            cout<<"PHONE NUMBER:"<<toc;
            gotoxy(20,17); cout<<"CITY:"<<rfee;
            break;
        }else
        { i++;
            if(i%max==0)
                i=0;
```

```
    } } while(i!=addr);if(found==0)
cout<<"\n\t\t\trecord does not exist in hash file\n";getch();
return;
}
```

4.2.4 Update a record

```
void student::modify(int addr,char k[])
{
int found=0,i,count,fn;
char dummy[10],temp[80]
;i=addr;
ifile.open(datafile,ios::in|ios::out);
while(!(ifile.eof()))
{
fn = ifile.tellg();
ifile.getline(eid,15,'|');
ifile.getline(temp,80,'\n');
if(strcmp(eid,k)==0)
{
ifile.seekg(fn,ios::beg);
ifile.put('$');
}}
ifile.close();
file.open(studentfile,ios::in|ios::out);
do
{
file.seekg(i*recsize,ios::beg);
file.getline(dummy,5,'\n');
if(strcmp(dummy,"####")==0)
break;
file.seekg(i*recsize,ios::beg);
```



```
file.getline(eid,15,'|');
if(strcmp(eid,k)==0)
{
found=1; textcolor(RED);
cout<<"\n"; gotoxy(20,12);
cout<<"RECORD FOUND\n";
cout<<endl;
file.getline(ename,20,'|');
gotoxy(20,13); cout<<"TEL_ID:"<<eid<<endl;
gotoxy(20,14);
cout<<"NAME:"<<ename<<"\n";
file.getline(ename,20,'|');
gotoxy(20,16);
cout<<"ENTER THE NAME OF THE PERSON:";
gets(ename);
gotoxy(20,17);
cout<<"Enter the PHONE NUMBER:";
gets(toc);
gotoxy(20,18);
cout<<"Enter the CITY:";
gets(rfee);
gotoxy(20,19);
strcpy(buffer,eid);    strcat(buffer,"|");
strcat(buffer,ename);  strcat(buffer,"|");
strcat(buffer,toc);    strcat(buffer,"|");
strcat(buffer,rfee);    strcat(buffer,"|");
file.seekp(addr*recsize,ios::beg);
file<<buffer;break;
}else
{
i++;
if(i%max==0)
i=0;
```



```
if(strcmp(eid,k)==0)
{
found=1;
cout<<"\n\t\t\t\t\tRECORD FOUND!!\n";
cout<<endl; file.getline(ename,20,'|');
cout<<"\t\t\t\t\tkey="<<eid<<endl<<"\n\t\t\t\t\ttname="<<ename<<"\n";
file.seekg(i*reccsize,ios::beg);
for(j=0;j<reccsize-2;j++)
file<<"#";file<<endl;
break;
}
else{
i++;
if(i%max==0)
i=0;
}}
while(i!=addr);
if(found==0)
cout<<"\n\t\t\t\t\trecord does not exist in hash file\n";
getch();
return;
}
```

4.2.6 Quit program

- Prompts the user to enter option 6 when he/she is in the console window.
- When the user enters 6 the program will be terminated.

Chapter 5

RESULTS

5.1 Main Menu

The first screen consists of Main menu which provides user choices to add the Telephone record , to search, delete or modify the record.

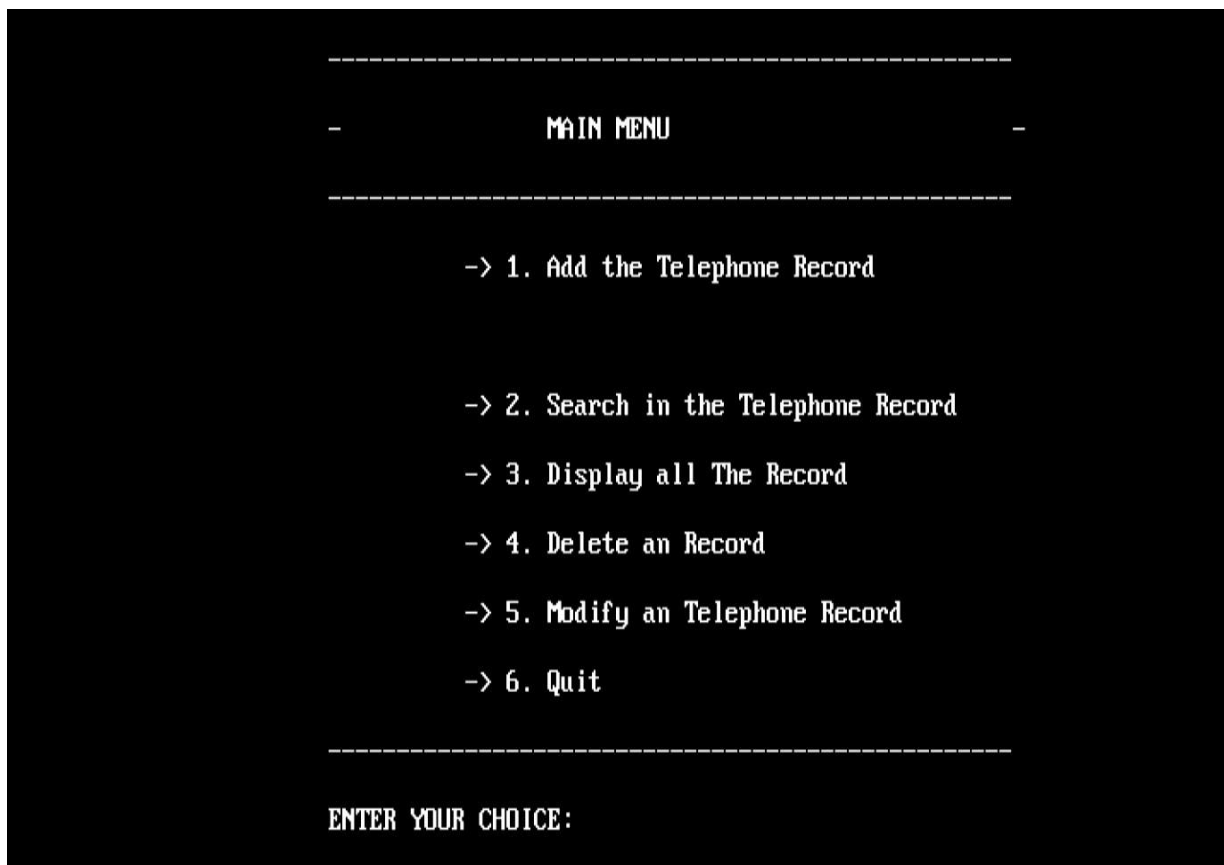


Figure 5.1 Main menu

5.1.1 Insertion Operation

User is asked to enter Telephone details which includes the attributes Tel_id, person name, phone number and city where he lives .

```
-----  
                        ADD THE RECORD  
-----  
  
      ENTER TELEPHONE DETAILS:  
Enter the TEL_id:(Tel__)1  
Enter the name of the Person:Uishwas  
Enter the Phone number:7986726374  
Enter the City:Bangalore_
```

Figure 5.2 Insertion of a record

5.1.2 Search Operation

Here the Telephone ID is entered and searched . If the given telephone id is found the search operation is successful and it displays the details of given telephone id. If the given telephone id is not found the search is unsuccessful with a alert “Record not found”.



Figure 5.3 Searching a record

5.1.3 Display Operation

Once the record is inserted, it can be displayed . It displays the record of previously inserted which includes all the attributes Tel_id , Name of the person, Phone number, City .

DIRECTORY DETAILS			
TEL_ID	NAME	PHONE NUMBER	CITY
1	jack	7898765456	Mysore
2	vasu	7897654323	banga lore
1	Uishwas	7986726374	Banga lore

Figure 5.4 Displaying records

5.1.4 Update Operation

If the entered value or if there is some modification required the administrator can modify the records. Record can be modified by entering the telephone id, if previous record is found the record can be updated or modified by entering new record details.

```
-----  
MODIFY AN Record  
-----  
ENTER TELEPHONE ID:2  
-----  
RECORD FOUND  
TEL_ID:2  
NAME:Usha  
  
ENTER THE NAME OF THE PERSON:Mary  
Enter the PHONE NUMBER:7876543453  
Enter the CITY:Mangalore  
-
```

Figure 5.5 Updating a record

5.1.5 Delete Operation

Delete operation includes deleting a record with the telephone id. If the given telephone id matches, then the record would be deleted otherwise it displays invalid record not found and delete operation is unsuccessful.

```
=====
                        DELETE AN TELEPHONE RECORD
=====

Enter the ID to delete:1

=====

                        RECORD FOUND!!

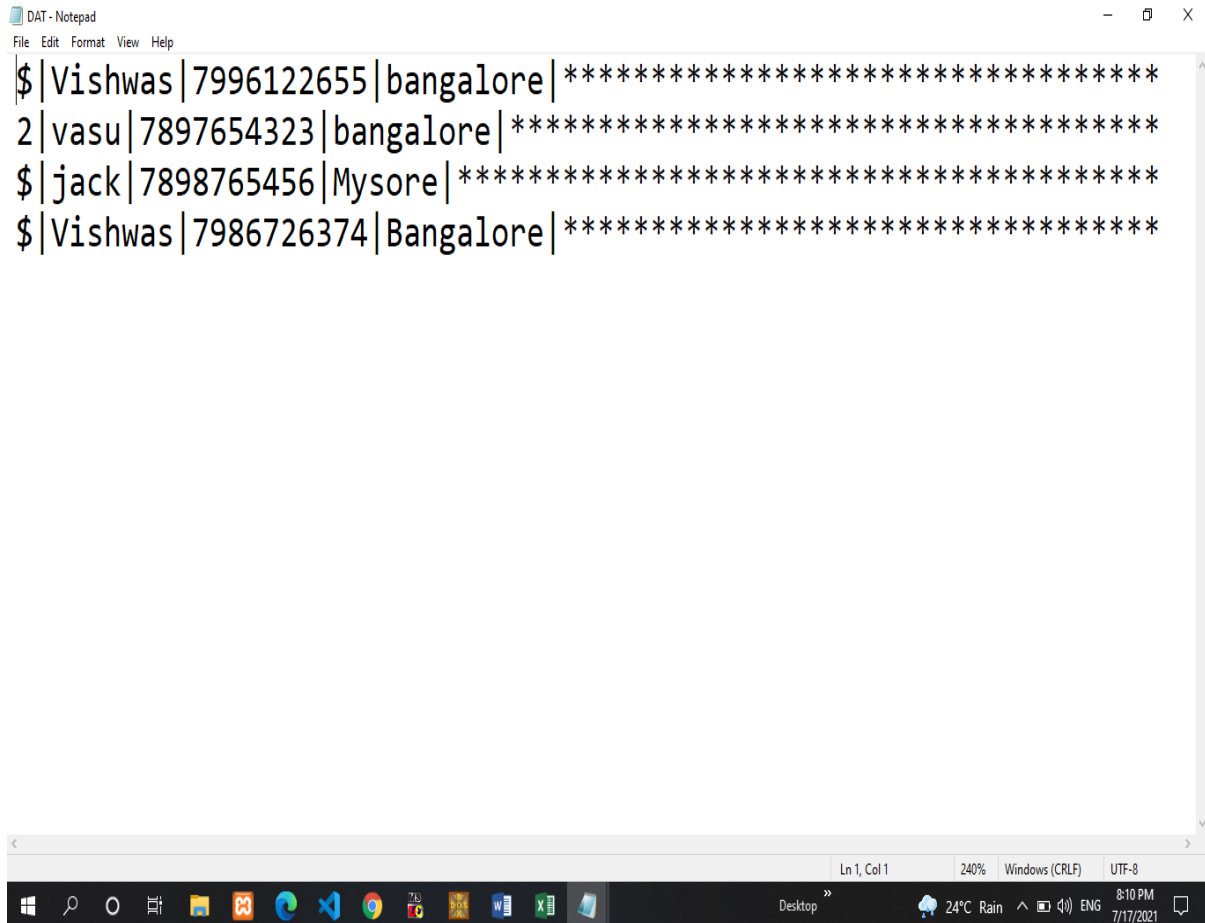
                        key=1

                        name= jack
```

Figure 5.6 Deletion of a record

5.2 Indexed Files

This contains index page where all the attributes of the project are included and stored in a DAT.txt file.



The screenshot shows a Notepad window titled 'DAT - Notepad' with a menu bar (File, Edit, Format, View, Help). The text content is as follows:

```
$|Vishwas|7996122655|bangalore|*****  
2|vasu|7897654323|bangalore|*****  
$|jack|7898765456|Mysore|*****  
$|Vishwas|7986726374|Bangalore|*****
```

The status bar at the bottom indicates 'Ln 1, Col 1', '240%', 'Windows (CRLF)', and 'UTF-8'. The Windows taskbar is visible at the bottom with various application icons and a system tray showing '24°C Rain' and the date '7/17/2021'.

Figure 5.7 Indexed File

CONCLUSION AND FUTURE ENHANCEMENT

Conclusion

The Mini Project “**Telephone Directory Management System**” is designed in order to reduce the burden of maintaining bulk records of all the details in which Inserting , Retrieving and deleting the details are easy when compared to the manual update and storing. This mini project helps in maintaining the user to store telephone directory details in an Organized manner, manipulate or manage the details and to replace old paper work system. This is to conclude that the project that I undertook was worked upon sincere effort. Most of the requirement are fulfilled up to the mark.

Future Enhancement:

For any system, present satisfaction is important, but it is also necessary to see and visualize the future scope. Future enhancement is necessary for any system as the limitations that cannot be denied by anybody. These limitations can be overcome by better technologies, Adding Feedback and suggestion option. The main goals of this mini-project are to learn accessing data from file system and displaying, fetching, retrieving, inserting, deleting to it using different techniques available. And to write Dataflow Diagram and Flow-Chart for the same file system.

REFERENCES

- [1] File Structures: An Object-Oriented Approach with C++ 3rd Edition by Michael J. Folk(Author), Bill Zoellick (Author), Greg Riccardi (Author).
- [2] The C++ Programming Language, 4th Edition by Bjarne Stroustrup (Author).
- [3] The Waite Group's C Programming Using Turbo C+/Book and Disk Subsequent Edition by Robert Lafore (Author).
- [4] <https://www.geeksforgeeks.org>
- [5] <https://www.log2base2.com/algorithms/searching/ hashing-in-c-data-structure.html>