**J-Component Report**

**PHONEBOOK APPLICATION**

*Submitted by*

**S.Pranav - 17BEC1073**

**G.Praveen - 17BEC1105**

**N.Ganesh - 17BEC1050**

**CSE2003**

**DATA STRUCTURES AND ALGORITHMS**

**BACHELOR OF TECHNOLOGY**

in

**Electronics and Communication Engineering**

****

JUNE-2020

**School of Electronics Engineering**

**DECLARATION BY THE CANDIDATE**

I hereby declare that the Report entitled “**PhoneBook Application”** submitted by me to VIT Chennai is a record of bonafide work undertaken by me under the supervision of **Dr. R. Karthik, Senior Assistant Professor, SENSE, VIT Chennai.**

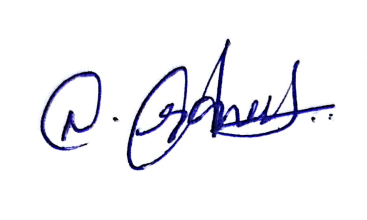
Chennai

Date:05-06-2020

Signature of the Candidate

 --- S.PRANAV (17BEC1073)

****--- G.PRAVEEN (17BEC1105)

****

--- N.GANESH (17BEC1050)

**ACKNOWLEDGEMENT**

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Dr.R.Karthik,** Professor, School of Electronics Engineering, for his consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

We are very grateful to **Dr.Siva subramaniahan,** Dean of the School of Electronics Engineering, VIT Chennai, for extending the facilities of the School towards our project and for her unstinting support.

We express our thanks to our Programme Chair **Dr.Vetrivelan.P** for his support throughout the course of this project.

We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the course. We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

S.PRANAV

G.PRAVEEN

N.GANESH

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Chapter** | Title | Page no |
| 1 | **ABSTRACT** | **1** |
| 2 | **INTRODUCTION** | **2** |
| 3 | **IDEALOGY** | **3** |
| 4 | **CODE** | **4** |
| 5 | **OUTPUT** | **14** |
| **6** | **USING JAVA** | **16** |
| **7** | **OUTPUTS** | **21** |
| **8** | | **DATA STRUCTURES USED** | **22** |
| **9** | | **CONCLUSION** | **23** |

**ABSTRACT**

This **Phonebook Application** allows you to perform simple Phonebook operations like in your mobile. You can add, list, modify, search and delete [Phonebook-related](https://www.codewithc.com/phonebook-management-system-project-in-c/) records. File handling and data structure concepts has been extensively used for almost all functions in this project.

Adding new records, listing them, modifying them and updating, search for contacts saved, and deleting the phonebook records are the basic functions which make up the main menu of this Phonebook application.

Personal information such as name, sex, father’s name, phone number, email and address are asked while adding a record into the Phonebook. These records can then be modified, listed, searched for and removed.

The program uses many functions. These functions are easy to understand as their name only signifies their respective operations. These functions make the program to divide into individual modules and hence it makes the program simpler to understand.

**Introduction**

The PhoneBook is an Application for storing contacts using data structures and algorithms so that the contact can be accessed very effectively. We have various functions associated with it, like Adding new contacts, Modifying the contacts, Search the contacts and Deleting contacts.

**Data Structures used:**

We are going to use Hash Tables with an open-chaining mechanism. Each Alphabet has a cell allocated to it and all the rest starting with the same alphabet and any additional starting with the same alphabets will be chained using a Doubly Linked List.

#### **Features:**

* This project helps to understand the basic concepts of the functions, file handling, and data structure.
* This application will teach beginners how to add, list, modify or edit, search, and delete data to/from the file.
* The basic functions are to add new records, listing them, modifying them and updating, search for contacts, and deleting the phonebook records.
* Developed using the C language.
* Easy to operate and understandable.

BASIC IDEA:

The phonebook books application is a very useful application which has multiple functional modules implemented in it.

Adding a new number.

Listing out all the contacts.

Modifying a contact.

Searching for a contact.

Deleting a contact.

Initially coded in Java, it was not very difficult to implement in C language as per the course requirements.

Adding a new number:

Allows us to add a new number with all the relevant information.

Listing out all the contacts.:

Lists out all the contacts with all the information stored.

Modifying a contact.: modifies the contact information.

Searching for a contact:

If a number is required, the person's name is given as input which will produce the phone number.

Deleting a contact.

A functionality for even deleting the contact with its entire information is also provided.

Finally we also included an additional exit function to check out from the app. And upon exit the previously saved information is not erased. It is stored in a seperate file.

This file can be accessed one the program is restarted.

The following images of the code in execution is very much self-explanatory and shows the versatality and the simplicity of as I would insist a moderately complex code.

**Code Used:**

**In C Language:**

#include<stdio.h>

#include<conio.h>

#include<string.h>

#include<stdlib.h>

struct person

{

char name[35];

char address[50];

char father\_name[35];

char mother\_name[30];

long int mble\_no;

char sex[8];

char mail[100];

char citision\_no[20];

};

void menu();

void got();

void start();

void back();

void addrecord();

void listrecord();

void modifyrecord();

void deleterecord();

void searchrecord();

int main()

{

system("color 5f");

start();

return 0;

}

void back()

{

start();

}

void start()

{

menu();

}

void menu()

{

system("cls");

printf("\t\t\*\*\*\*\*WELCOME TO PHONEBOOK\*\*\*\*");

printf("\n\n\t\t\t MENU\t\t\n\n");

printf("\t1.Add New \t2.List \t3.Exit \n\t4.Modify \t5.Search \t6.Delete");

switch(getch())

{

case '1':

addrecord();

break;

case '2': listrecord();

break;

case '3': exit(0);

break;

case '4': modifyrecord();

break;

case '5': searchrecord();

break;

case '6': deleterecord();

break;

default:

system("cls");

printf("\nEnter 1 to 6 only");

printf("\n Enter any key");

getch();

menu();

}

}

void addrecord()

{

system("cls");

FILE \*f;

struct person p;

f=fopen("project","ab+");

printf("\n Enter name: ");

got(p.name);

printf("\nEnter the address: ");

got(p.address);

printf("\nEnter father name: ");

got(p.father\_name);

printf("\nEnter mother name: ");

got(p.mother\_name);

printf("\nEnter phone no.:");

scanf("%ld",&p.mble\_no);

printf("Enter sex:");

got(p.sex);

printf("\nEnter e-mail:");

got(p.mail);

printf("\nEnter citizen no:");

got(p.citision\_no);

fwrite(&p,sizeof(p),1,f);

fflush(stdin);

printf("\nrecord saved");

fclose(f);

printf("\n\nEnter any key");

getch();

system("cls");

menu();

}

void listrecord()

{

struct person p;

FILE \*f;

f=fopen("project","rb");

if(f==NULL)

{

printf("\nfile opening error in listing :");

exit(1);

}

while(fread(&p,sizeof(p),1,f)==1)

{

printf("\n\n\n YOUR RECORD IS\n\n ");

printf("\nName=%s\nAdress=%s\nFather name=%s\nMother name=%s\nMobileno=%ld\nSex=%s\nE-mail=%s\nCitizen no=%s",p.name,p.address,p.father\_name,p.mother\_name,p.mble\_no,p.sex,p.mail,p.citision\_no);

getch();

system("cls");

}

fclose(f);

printf("\n Enter any key");

getch();

system("cls");

menu();

}

void searchrecord()

{

struct person p;

FILE \*f;

char name[100];

f=fopen("project","rb");

if(f==NULL)

{

printf("\n error in opening\a\a\a\a");

exit(1);

}

printf("\nEnter name of person to search\n");

got(name);

while(fread(&p,sizeof(p),1,f)==1)

{

if(strcmp(p.name,name)==0)

{

printf("\n\tDetail Information About %s",name);

printf("\nName:%s\naddress:%s\nFather name:%s\nMother name:%s\nMobileno:%ld\nsex:%s\nE-mail:%s\nCitision no:%s",p.name,p.address,p.father\_name,p.mother\_name,p.mble\_no,p.sex,p.mail,p.citision\_no);

}

else

printf("file not found");

}

fclose(f);

printf("\n Enter any key");

getch();

system("cls");

menu();

}

void deleterecord()

{

struct person p;

FILE \*f,\*ft;

int flag;

char name[100];

f=fopen("project","rb");

if(f==NULL)

{

printf("CONTACT'S DATA NOT ADDED YET.");

}

else

{

ft=fopen("temp","wb+");

if(ft==NULL)

printf("file opaning error");

else

{

printf("Enter CONTACT'S NAME:");

got(name);

fflush(stdin);

while(fread(&p,sizeof(p),1,f)==1)

{

if(strcmp(p.name,name)!=0)

fwrite(&p,sizeof(p),1,ft);

if(strcmp(p.name,name)==0)

flag=1;

}

fclose(f);

fclose(ft);

if(flag!=1)

{

printf("NO CONACT'S RECORD TO DELETE.");

remove("temp.txt");

}

else

{

remove("project");

rename("temp.txt","project");

printf("RECORD DELETED SUCCESSFULLY.");

}

}

}

printf("\n Enter any key");

getch();

system("cls");

menu();

}

void modifyrecord()

{

int c;

FILE \*f;

int flag=0;

struct person p,s;

char name[50];

f=fopen("project","rb+");

if(f==NULL)

{

printf("CONTACT'S DATA NOT ADDED YET.");

exit(1);

}

else

{

system("cls");

printf("\nEnter CONTACT'S NAME TO MODIFY:\n");

got(name);

while(fread(&p,sizeof(p),1,f)==1)

{

if(strcmp(name,p.name)==0)

{

printf("\n Enter name:");

got(s.name);

printf("\nEnter the address:");

got(s.address);

printf("\nEnter father name:");

got(s.father\_name);

printf("\nEnter mother name:");

got(s.mother\_name);

printf("\nEnter phone no:");

scanf("%ld",&s.mble\_no);

printf("\nEnter sex:");

got(s.sex);

printf("\nEnter e-mail:");

got(s.mail);

printf("\nEnter citizen no\n");

got(s.citision\_no);

fseek(f,-sizeof(p),SEEK\_CUR);

fwrite(&s,sizeof(p),1,f);

flag=1;

break;

}

fflush(stdin);

}

if(flag==1)

{

printf("\n your data id modified");

}

else

{

printf(" \n data is not found");

}

fclose(f);

}

printf("\n Enter any key");

getch();

system("cls");

menu();

}

void got(char \*name)

{

int i=0,j;

char c,ch;

do

{

c=getch();

if(c!=8&&c!=13)

{

\*(name+i)=c;

putchar(c);

i++;

}

if(c==8)

{

if(i>0)

{

i--;

}

// printf("h");

system("cls");

for(j=0;j<i;j++)

{

ch=\*(name+j);

putchar(ch);

}

}

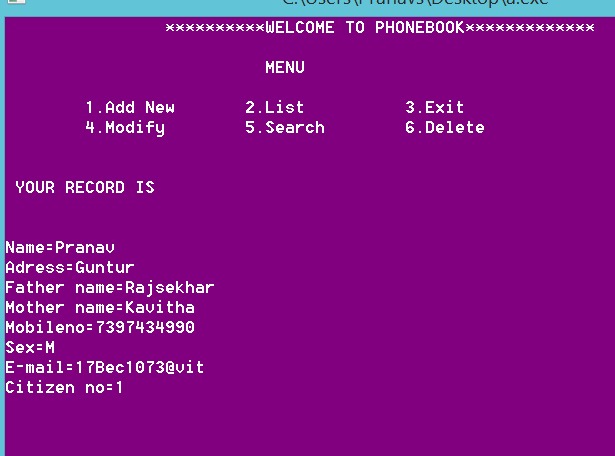
}while(c!=13);

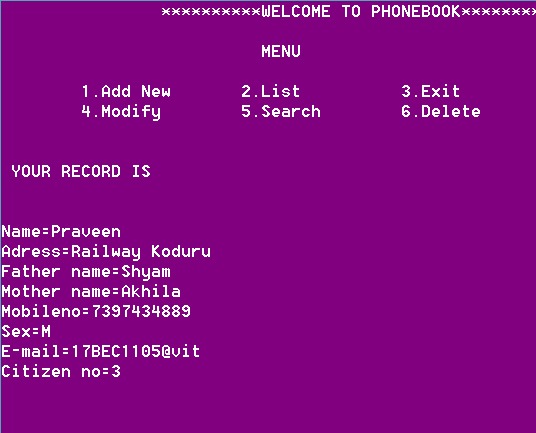
\*(name+i)='\0';

}

**OUTPUT:**

****

****

****

****

**In Java:**

import java.util.Collection;

import java.util.Collections;

import java.util.HashMap;

import java.util.Iterator;

import java.util.LinkedList;

import java.util.Map;

import java.util.Scanner;

import java.util.Set;

public class PhoneBook extends App{

public static void main(String[] args) {

String fname,lname;

Scanner input=new Scanner(System.in);

PhoneBook ph=new PhoneBook();

int acode,ch=0;

long number;

do{

System.out.println("Select a choice\n1:Add a new phonebook entry\n2:Find number\n3:Find name for a number\n4:Display phonebook\n5:Exit\n");

try{

ch=input.nextInt();

}catch (Exception e) {

System.out.println("Sorry you entered string .....program is terminated...");

break;

}

switch(ch){

case 1 :

System.out.println("Enter the first name: ");

fname=input.next();

System.out.println("Enter the last name: ");

lname=input.next();

System.out.println("Enter the areacode: ");

acode=input.nextInt();

System.out.println("Enter phone number: ");

number=input.nextLong();

if(ph.doEntry(fname, lname, acode, number)==true)

System.out.println("Record successfully saved.\n");

else

System.out.println("Phone number repeated.\n");

break;

case 2:

System.out.println("Enter the name");

fname=input.next();

boolean bb=true;

long a[]=ph.findNumber(fname);

for(long i : a){

if(i!=0){

System.out.println("Firstname :"+fname+"\nNumber: "+i+"\n");

bb=false;

}

}

if(bb)

System.out.println("No record found...");

break;

case 3:

System.out.println("Enter the number for name : ");

number=input.nextInt();

String ss=ph.findName(number);

if(ss.length()==0){

System.out.println("No record found"+"\n");

}

else

System.out.println("Name is : "+ph.findName(number)+"\n");

break;

case 4:

ph.display();

break;

case 5 :

break;

default:

System.out.println("Invalid input!!");

}

}while(ch!=9);

}

}

class Person{

String fname;

String lname;

Person(String fname,String lname){

this.fname=fname;

this.lname=lname;

}

}

class PhoneNumber{

int acode;

long number;

public PhoneNumber(int acode,long number2) {

this.acode=acode;

this.number=number2;

}

}

class BookEntry{

Person p;

PhoneNumber pn;

BookEntry(Person p,PhoneNumber pn){

this.p=p;

this.pn=pn;

}

}

class App{

HashMap map=new HashMap();

Map.Entry map1;

Collection c;

Iterator it,it1;

Person p;

Set s;

BookEntry b,b1;

boolean check;

PhoneNumber pn;

public boolean doEntry(String fname,String lname,int acode,long number){

p=new Person(fname, lname);

pn=new PhoneNumber(acode, number);

b=new BookEntry(p, pn);

check=true;

s=map.entrySet();

it=s.iterator();

while (it.hasNext() ) {

map1=(Map.Entry)it.next();

b1=(BookEntry)map1.getValue();

if(b1.pn.number==number){

check=false;

break;

}

}

if(check){

map.put(p,b);

}

return check;

}

public void display(){

s=map.entrySet();

it=s.iterator();

System.out.println("\n");

System.out.printf("%-15s%-15s%-15s%-15s\n","First name","Last name", "Areacode","Phone number");

while(it.hasNext()){

map1=(Map.Entry)it.next();

b=(BookEntry)map1.getValue();

System.out.printf("%-15s%-15s%-15s%-15s",b.p.fname,b.p.lname,b.pn.acode,b.pn.number);

System.out.println();

}

System.out.println("\n");

}

public long[] findNumber(String fname) {

long a[]=new long[map.size()];

int i=0;

s=map.entrySet();

it=s.iterator();

while ( it.hasNext() ) {

map1=(Map.Entry)it.next();

b=(BookEntry)map1.getValue();

if(b.p.fname.equals(fname)){

a[i]=b.pn.number;

i++;

}

}

return a;

}

public String findName(long number){

String st="";

s=map.entrySet();

it=s.iterator();

while ( it.hasNext() ) {

map1=(Map.Entry)it.next();

b=(BookEntry)map1.getValue();

if(b.pn.number==number){

st=b.p.fname;

}

}

return st;

}

public void sort(){

s=map.entrySet();

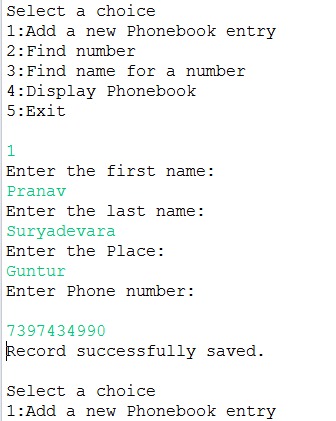
LinkedList lt=(LinkedList)s;

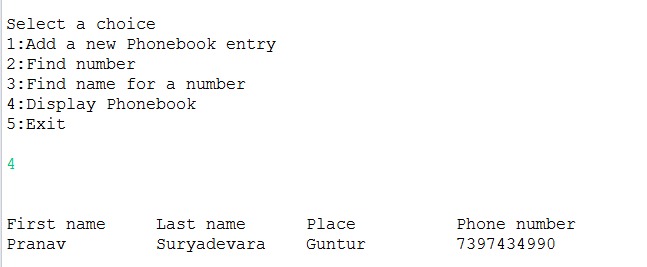
Collections.sort(lt);

}

}

**OUTPUT:**





**Data Structures Used:**

The main things we used in the above java code are Hash map, Collections, Lexicographical sorting

Lexicographical Sorting: In mathematics, the lexicographic or lexicographical order (also known as lexical order, dictionary order, alphabetical order or lexicographic(al) product) is a generalization of the way words are alphabetically ordered based on the alphabetical order of their component letters. This generalization consists primarily in defining a total order on the sequences (often called strings in computer science) of elements of a finite totally ordered set, often called an alphabet.

There are several variants and generalizations of the lexicographical ordering. One variant widely used in combinatorics orders subsets of a given finite set by assigning a total order to the finite set, and converting subsets into increasing sequences, to which the lexicographical order is applied. Another generalization defines an order on a Cartesian product of partially ordered sets; this order is a total order if and only if the factors of the Cartesian product are totally ordered. This kind of the sorting techniques are usually used on input is automatically in Java.

HashMap:

HashMap is a Map based collection class that is used for storing Key & value pairs, it is denoted as HashMap<Key, Value> or HashMap<K, V>. This class makes no guarantees as to the order of the map. It is similar to the Hashtable class except that it is unsynchronized and permits nulls(null values and null key).

It is not an ordered collection which means it does not return the keys and values in the same order in which they have been inserted into the HashMap. It does not sort the stored keys and Values. You must need to import java.util.HashMap or its super class in order to use the HashMap class and methods.

Collections:The **Collection in Java** is a framework that provides an architecture to store and manipulate the group of objects.

Java Collections can achieve all the operations that you perform on a data such as searching, sorting, insertion, manipulation, and deletion.

Java Collection means a single unit of objects. Java Collection framework provides many interfaces (Set, List, Queue, Deque) and classes ([ArrayList](https://www.javatpoint.com/java-arraylist), Vector, [LinkedList](https://www.javatpoint.com/java-linkedlist), [PriorityQueue](https://www.javatpoint.com/java-priorityqueue), HashSet, LinkedHashSet, TreeSet).

**CONCLUSION**

Hence the prototype has been well devised and was implemented successfully deployed. The algorithm used and its implementation were very effective.

FUTURE WORK

1. Can be developed further more by using hashing which makes the searching even more faster.
2. A website can be made which can be used to store the files from the program.
3. An application for smartphones can be developed by using platform specific software by using the code as it is.

**REFERENCES**

* <https://www.codewithc.com/mini-project-in-c-phonebook/>
* <https://clangfundamentals.blogspot.com/2018/06/phonebook-using-c-program.html>
* <https://projectnotes.org/it-projects/phonebook-system-in-c-c-with-source-code/>