KIET Group of Institutions, Ghaziabad

COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

PROJECT BASED LEARNING

on

PHONE DIRECTORY APPLICATION

SUBJECT: DESIGN AND ANALYSIS OF ALGORITHM LAB

(KCS-553)

Submitted By:

DHRUV RASTOGI	- 2100290110051 (CSIT 3A)
ADARSH KUMAR DUBEY	- 2100290110009 (CSIT 3A)
HARSH SINGH	- 2100290110061 (CSIT 3A)

ACKNOWLEDGEMENT

I’ve got this golden opportunity to express my kind gratitude and sincere thanks to my subject faculty **“Mr.Vinay Kumar”**, Computer Science and Information Technology Department, **KIET GROUP OF INSTITUTIONS** for their kind support and necessary counselling in the preparation of this project report. I’m also indebted to each and every person responsible for the making up of this project directly or indirectly.

I must also acknowledge or deep debt of gratitude each one of my colleague who led this project come out in the way it is. It’s my hard work and untiring sincere efforts and mutual cooperation to bring out the project work. Last but not the least, I would like to thank my parents for their sound counselling and cheerful support. They have always inspired us and kept our spirit up.

Aim: To make an algorithm which can search through a data as fast as possible.

ABSTRACT

The main purpose of a phone book app is to keep track of the people you call. The phone book app will have basic features like adding a new contact, searching for a contact, updating a contact, and deleting a contact. This small C Phonebook project lets you do simple Phonebook operations just like on your phone.

One can add, list, change, search, and delete records related to the phonebook. File management and data structure ideas have been used a lot in almost every part of this mini project. phonebook application in C is a console program that doesn’t have any pictures.

The source code is full and has no mistakes at all. It is put together with the GCC compiler in Code Blocks. Functions, file handling, and the structure of data are used. This program shows you how to add, list, change, edit, search for, and delete data from or from a file.

The main menu of this Phonebook app lets you add new records, list them, change and update them, search for saved contacts, and delete phonebook records. These are the basic functions.

When adding a record to the phonebook, you are asked for personal information like your name, gender, father’s name, phone number, citizenship number, email address, and address. Then, these records can be changed, put on a list, searched for, and taken away.

BRIEF DESCRIPTION

INTRODUCTION

This C Phonebook small project enables you to do basic Phonebook operations much as you would on your phone. Phonebook-related records may be added, listed, modified, searched, and deleted. Almost every function in this little project makes substantial use of file management and data structure ideas.

Phonebook is a simple C mini project that will teach you the fundamentals of functions, file handling, and data structure. This program will show you how to add, list, change, edit, search, andremove data from and into a file.

This mini project in C Phonebook allows you to perform simple Phonebook operations like in your mobile. You can add, list, modify, search and delete Phonebook-related records. File handling and data structure concepts has been extensively used for almost all functions in this mini project

EXPLAINATION

The primary features that make up the main menu of this Phonebook program include adding new records, listing them, altering and updating them, searching for stored contacts, and deleting phonebook data.

When adding a record to the Phonebook, personal information such as name, sex, father's name, phone number, citizenship number, email, and address is requested. Modifying, listing, searching for, and removing these records is then possible.

In this short project, we employed a variety of functions. These functions are simple to grasp since their names solely refer to the actions they perform.

One of the simplest mini projects built by the Code with C team is the Phonebook application. It is aimed mostly for novices who are just beginning to construct tiny projects in the C programming language. Personal diary management and contact management systems are two more initiatives that are extremely comparable.

ABOUT PHONEBOOK

Phonebook is a very simple mini project in C that can help you understand the basic concepts of functions, file handling and data structure. This application will teach you how to add, list, modify or edit, search and delete data to/from the file.

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, citizenship number, email and address are asked while adding a record into the Phonebook. These records can then be modified, listed, searched for and removed.

I have used many functions in this mini project. These functions are easy to understand as their name only signifies their respective operations.

- void menu() – This function is used to display the main menu.
- void start() – This functions calls the menu function mentionedabove.
- void back() – This function is used to go back to start.
- void addrecord() – It adds a new Phonebook record.
- void listrecord() – This function is used to view list of added recordsin file.
- void modifyrecord() – This function is used to modify added records.
- void deleterecord() – It deletes record from file.
- void searchrecord() – It searches for added record by name.

It is a very simple tool that helps you learn the basics of creating files, file extensions, and how data is organized. This software shows you how to add, view, edit, change, receive, and delete data from files.

One of the main things you can do with the main phonebook application (shown in the main menu below) is add new items, view them by logging in, edit and update them, search for saved contacts, and delete data.

To add a login to the phonebook, you need to know things about the person, like their name, type, identity, phone number, nationality, email address, and address. After that, you can change, look at, search for, and delete this text. It is thought that more than 600 million people use mobile phones around the world, and that number is growing.

The success of cell phones is easy to explain: people always have them with them. When a trader goes from one place to another, he is doing business without business. If the boy gets home late, he can let his parents know.

If your device is giving you trouble, you can ask for help along the way. People use cell phones to talk to each other and to other people. Most informal meetings, like going to the bar, are set up on an anonymous and up-to-date map through a cell phone.

A cell phone can be used in many different ways. There are, however, some problems. Technology wants to reach people everywhere and anywhere, but just because it exists doesn't mean it can reach all possible people.

The way out of this problem is for customers to share information about their real lives. But now, how mobile users interact with older versions depends on them. In particular, the connection between the user and the general user does not take the situation into account. This makes it hard to know when and why to call.

What is phonebook application?

The Phonebook app includes a basic set of functions for adding, searching for, updating, and deleting new contacts. This mini-C phonebook design enables you to perform basic tasks in your phonebook, such as dialling numbers. You can add text to the phonebook, as well as find, edit, search, and delete entries.

Why phonebook is important?

Because its goal is to find the phone number of a subscriber who has been identified by name and address.

Does the phonebook still exist?

Yes, White pages and phone books are as old as the rotary-dial phone. But both are still online in digital form.

TECHNOLOGY USED

PHONEBOOK PROJECT	**PROJECT DETAILS**
Project Name:	PHONE DIRECTORY APPLICATION
Project Platform:	C/C++
Programming Language Used:	C Programming Language
IDE Tool (Recommended):	Dev-C++/Code blocks
Project Type:	Desktop Application
Database:	Stores data in .DAT file

TOPICS RELATED TO DSUC ON WHICH PHONE DIRECTORY APPLICATION PROJECT IS MAPPED

1-D ARRAY

A One-Dimensional Array is the simplest form of an Array in which the elements are stored linearly and can be accessed individually by specifying the index value of each element stored in the array. This is one of the simplest forms of Array. They are very easy to define and use in the programs. The values stored in a One-Dimensional Array can be easily initialized and manipulated, making it a very feasible Data structure type.

STRUCTURES

Structures (also called structs) are a way to group several related variables into one place. Each variable in the structure is known as a member of the structure. Unlike an array, a structure can contain many different data types (int, float, char, etc.).You can create a structure by using the struct keyword and declare each of its members inside curly braces

To access the structure, you must create a variable of it. Use the struct keyword inside the main() method, followed by the name of the structure and then the name of the structure variable:

DATA TYPES (CHAR, INT, LONG INT)

Data types are the type of data stored in a C program. Data types are used while defining a variable or functions in C. It’s important for the compiler to understand the type of predefined data it is going to encounter in the program. In general terms, a data type is an attribute that tells a computer how to interpret the value.

Let’s say if variables are containers, then data type is the type of container. The type of container tells what kind of stuff it should contain. For example, you won’t put cookies in a bottle. Right! Similarly, you don’t store an integer value in a variable of data type String.

Modifiers are C keywords that modify the meaning of fundamental data types. It indicates how much memory will be allocated to a variable. To adjust the memory allocated for a variable, modifiers are prefixed with fundamental data types. Like long, short, signed, unsigned

STRINGS

String in C programming is a sequence of characters terminated with a null character ‘\0’. Strings are defined as an array of characters. The difference between a character array and a string is the string is terminated with a unique character ‘\0’. Declaring a string is as simple as declaring a one-dimensional array. Below is the basic syntax for declaring a string.

```
char str_name[size];
```

In the above syntax **str_name** is any name given to the string variable and size is used to define the length of the string, i.e the number of characters strings will store.
Note: There is an extra terminating character which is the ***Null character (‘\0’) used to indicate the termination of a string that differs strings from normal character arrays***. When a Sequence of characters enclosed in the double quotation marks is encountered by the compiler, a null character ‘\0’ is appended at the end of the string by default.

FILE HANDLING

The process of file handling refers to how we store the available data or info in a file with the help of a program. The C language stores all the data available in a program into a file with the help of file handling in C. This data can be fetched/extracted from these files to work again in any program.

File handling refers to the method of storing data in the C program in the form of an output or input that might have been generated while running a C program in a data file, i.e., a binary file or a text file for future analysis and reference in that very program.

file refers to a source in which a program stores the information/data in the form of bytes of sequence on a disk (permanently). The content available on a file isn’t volatile like the compiler memory in C. But the program can perform various operations, such as creating, opening, reading a file, or even manipulating the data present inside the file. This process is known as file handling in C.

We can use a variety of functions in order to open a file, read it, write more data, create a new file, close or delete a file, search for a file, etc. These are known as file handling operators in C.

Like : fopen, fread, fclose, fwrite, fputc, fgetc, etc

Note: It is important to know that we must declare a file-type pointer when we are working with various files in a program. This helps establish direct communication between a program and the files.

Here is how you can do it:

FILE *fpointer;

Out of all the operations/functions mentioned above, let us discuss some of the basic operations that we perform in the C language.

PROBLEM STATEMENT

Nowadays it is very difficult to hard core every details related to any phone number such as email address, father name , mother name, gender, citizen number etc on a paper and it is nearly impossible to modify previous details of a particular phone number and it also very difficult to search a particular number with details on a hard copy. We also cannot be able to delete any phone number with details

PROPOSED SOLUTION

This software is very useful nowadays to store full information under a single contact number. The software also has options for removing and changing the contact number entered. This C Phonebook small project enables you to do basic Phonebook operations much as you would on your phone. Phonebook-related records may be added, listed, modified, searched, and deleted. 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

CODING IMPLEMENTATION:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<stdlib.h>
#include<windows.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");
```

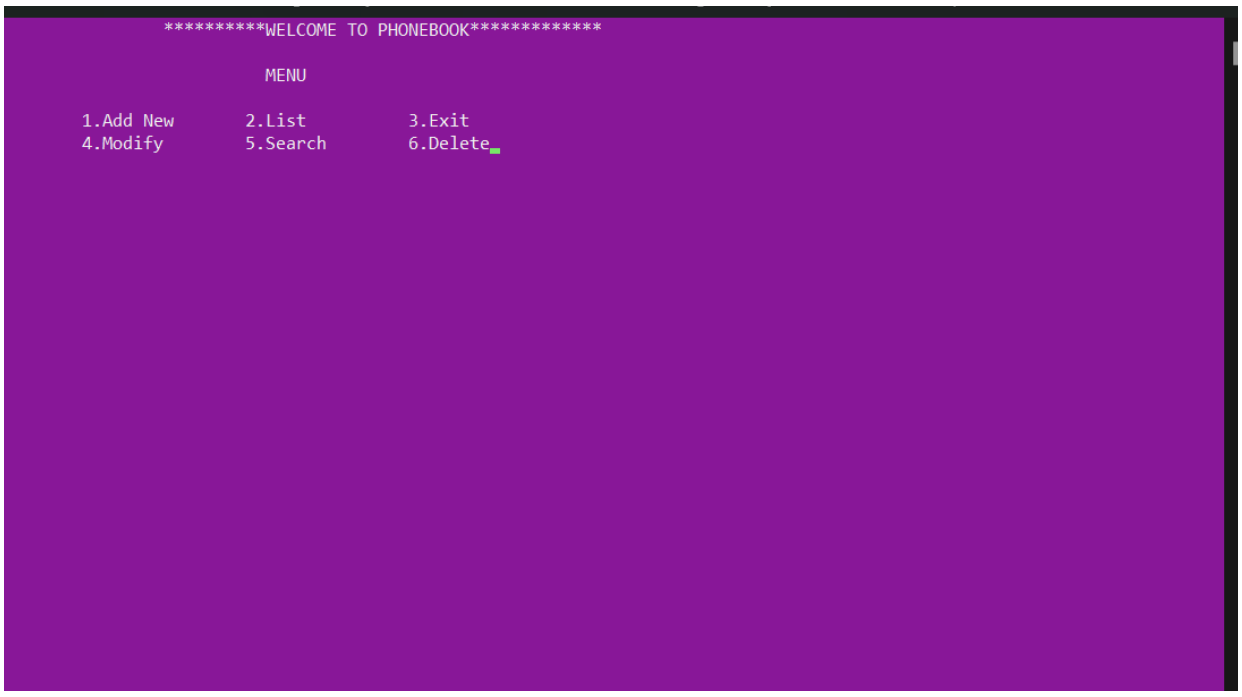
```
fclose(f);

    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\nMobile no=%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\nMobile
no:%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.ci
tision_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)
                {
```

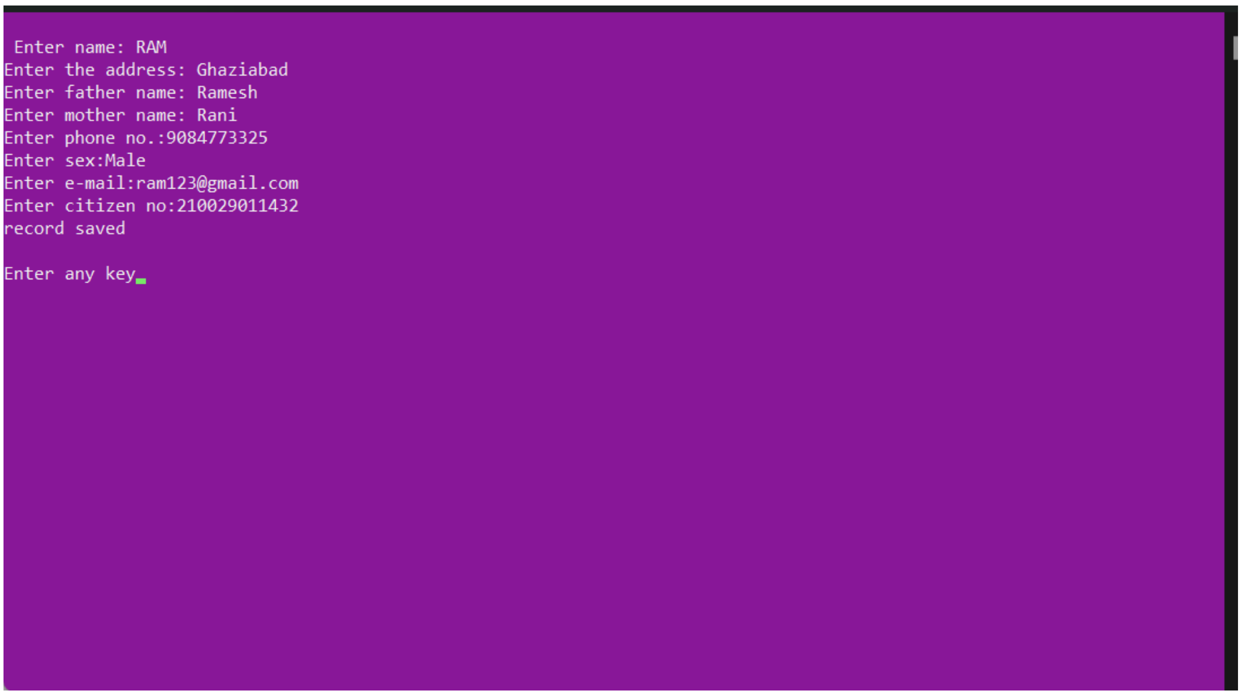
```c
                     printf("\n Enter name:");
                     got(s.name);
                     printf("\nEnter the address:");
                     got(s.address);
                     printf("\nEnter father name:");
                     got(s.father_name);
                     printf("\nEnter 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;
```

```c
    char c,ch;
    do
    {
        c=getch();
                if(c!=8&&c!=13)
                {
                    *(name+i)=c;
                        putch(c);
                        i++;
                }
                if(c==8)
                {
                    if(i>0)
                    {
                        i--;
                    }
                   // printf("h");
                    system("cls");
                    for(j=0;j<i;j++)
                    {
                        ch=*(name+j);
                        putch(ch);

                    }

                }
    }while(c!=13);
      *(name+i)='\0';
}
```

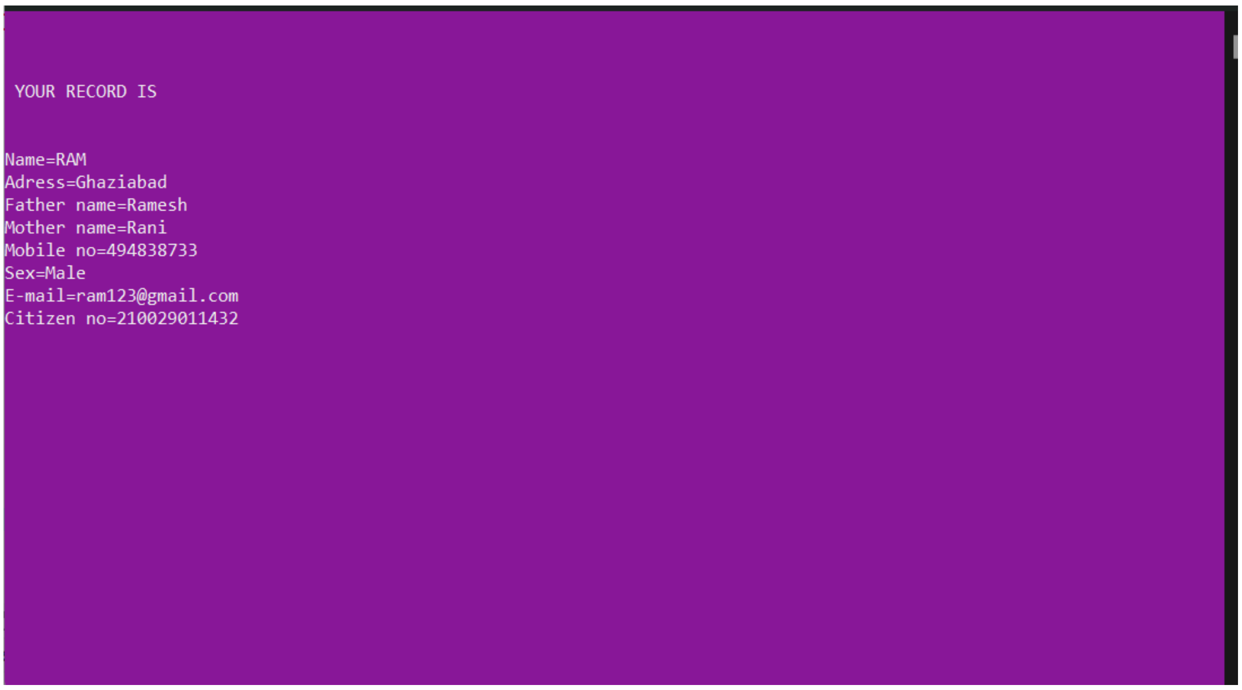
RESULT/OUTPUT



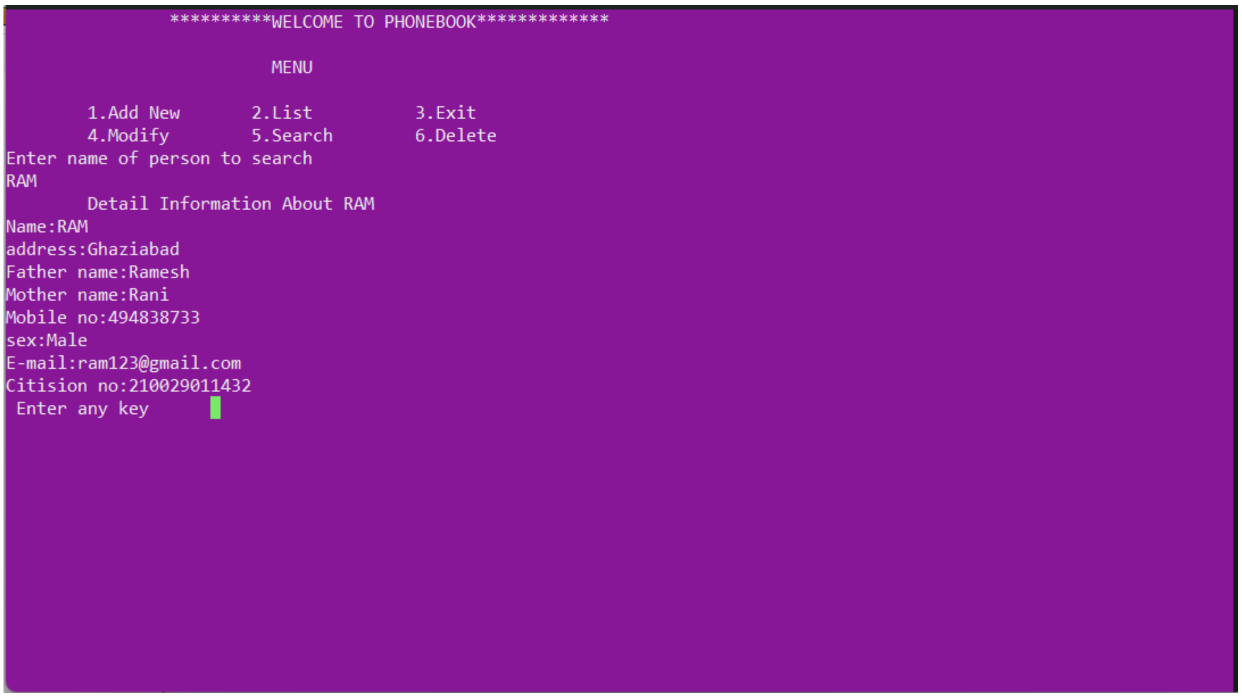
```
                **********WELCOME TO PHONEBOOK*************

                          MENU

        1.Add New        2.List          3.Exit
        4.Modify         5.Search        6.Delete_
```



```
 Enter name: RAM
Enter the address: Ghaziabad
Enter father name: Ramesh
Enter mother name: Rani
Enter phone no.:9084773325
Enter sex:Male
Enter e-mail:ram123@gmail.com
Enter citizen no:210029011432
record saved

Enter any key_
```



```
 YOUR RECORD IS


Name=RAM
Adress=Ghaziabad
Father name=Ramesh
Mother name=Rani
Mobile no=494838733
Sex=Male
E-mail=ram123@gmail.com
Citizen no=210029011432
```



```
                **********WELCOME TO PHONEBOOK*************

                          MENU

        1.Add New       2.List          3.Exit
        4.Modify        5.Search        6.Delete
Enter name of person to search
RAM
        Detail Information About RAM
Name:RAM
address:Ghaziabad
Father name:Ramesh
Mother name:Rani
Mobile no:494838733
sex:Male
E-mail:ram123@gmail.com
Citision no:210029011432
 Enter any key
```

OUTCOME OF PROJECT

The phonebook is a very simple C mini project that can help you understand the basic concepts of functions, file handling, and data structure. This program will teach you how to add, list, change or edit, search and remove data from/to a file. Adding new records, listing them, editing and updating them, looking for saved contacts, and removing phonebook records are simple Functions that make up the main menu of this Phonebook program.

Personal information, such as name, gender, phone number, e-mail, and address, is requested when you add a record to your phonebook. These records can then be updated, entered, searched, and deleted. I’ve used a lot of functions in this mini project. These functions are easy to understand since their name means just their respective operations.