Department of Electrical Engineering

Group Members:

Zaeem Muzammil (bsee2135@pieas.edu.pk)

Muhammad Ali Sajid (bsee2117@pieas.edu.pk)x

Section: B

Dated: 06/01/2022

CIS-101L: COMPUTER FUNDAMENTALS & PROGRAMMING

SEMESTER PROJECT

PHONEBOOK DIRECTORY

ABSTRACT:

The phonebook application works specifically for tracking people. The Phonebook application contains a set of basic functions for adding, searching, updating, and deleting new contacts. This Mini-C++ phonebook design allows you to perform simple tasks in your phonebook, such as mobile phones. You can add text to the phonebook, find, edit, search, and delete. The concept of file management and data structure is often used in almost all functions in this project. It uses functions, file management, and data structure. This application provides information on adding, viewing, modifying, receiving, and deleting data from/to files. Adding new entries, browsing them, editing and updating, searching for saved contacts, and deleting contacts in the phonebook is one of the most important services that become the main menu in the phonebook application. When you add anything to your phone book, you will be asked for personal information such as name, gender, first name, phone number, nationality, email address, and address. You can then edit, view, search, and delete this text.

INTRODUCTION:

This is a user-friendly program. This helps maintain a direct connection between the computer and the user. Its purpose is to create a simple but efficient phonebook that can be used by the user to add, display, edit, search, delete and exit icon. For this instance, we will use different aspects of C++ Language which are as follows:

1. To familiarize with different libraries in C++ Language which includes “<iostream>”, “<fstream>”, “<string>”, “<conio.h” and “iomanip”.
2. Use of “if, else” statements
3. Use of loops
4. Use of File Handling for Opening and Saving data
5. Storing of Data in Array
6. Use of Modular Programming to Make Program Efficient
7. Use of Structures for Storing Data

THEORETICAL TREATEMENT:

1. **LIBRARIES**

To form a phonebook management system following libraries are used in the coding:

i) **<iostream>**
   - Iostream is the basic library used for declaring output statements and inputting data from the user. For output we use "cout" and for input we use "cin".
   - Another abbreviation for standard is "using namespace std" that is used to link cout and cin statements. Its main function is to skip getline code and just use cout and cin statements.

ii) **<fstream>**
   This library is the main library used in our program that will perform all main tasks of project from reading, writing and editing number. It includes following divisions:
   (1) **OF Stream**
   Of Stream is used to for writing purpose. Append mode will be used to write in the file. Append mode not only writes in the file but also saves data which can further be used for editing and reading purpose.
   (2) **IF Stream**
   If Stream is only used to open the saved file in reading mode. In our program, if stream will display all the details of the program inputted by the user. If the user has not entered any details of the contact no details will be displayed. If stream uses open function to display all the details of the contact.
   (3) **F Stream**
   F Stream can basically perform both tasks simultaneously. It will open the file in append mode as well as allow the user to read the input file.

iii) **<string>**
   This library is used to manipulate string function in the program. String function can save values from integer to character which is defined in this library.

2. **IF ELSE STATEMENTS**
   If Else statements play a key role in writing a simple but effective program. It gives a choice to a program to either go with first situation or number of situations entered.
3. **LOOPS**
   A loop is used for executing a block of statements repeatedly until a particular condition is satisfied. It continues until the given statement falsifies. Following are some type of loops:
   i) for loop
   ii) while loop
   iii) do while loop
4. **FILE HANDLING**
   - To save and open the file, file handling will be used. File handling is used to save large amount of input data so that no loss of data may occur. It can also open the file in append mode and allow the user to input data and then also open the input data in read mode.

5. **ARRAYS**
    - Using arrays is a very unique approach for doing the code. Arrays are data storing hideouts that can save large amount of data. It can save a number whether it is integer, float, double or long.
    - Arrays can not only be used to save the data but multi-dimensional arrays can also be used to manipulate matrices, finding inverses and as a result solve difficult problems and mathematical equations.
6. **MODULAR PROGRAMMING**
    - Modular programming is the process of subdividing a computer program into separate sub-programs. A module is the separate component formed during modular programming.
    - It can often be used in a variety of applications and functions with other components of the system.
    - Modular programming usually makes your code easier to read because it separates it into functions that reads only one aspect of the program. It can make your files a lot smaller and easier to understand compared to monolithic code that defines all the program in a single function usually the main function.
    - Due to the use of modular programming, our program becomes more efficient and its readability increases. Moreover, a logical error is also easier to locate in a sub-divided program.
    - Modular programming is used to divide all the program in sub-parts by forming modules and using functions that makes the program simpler and easier to understand.
7. **STRUCTURES**
    - Structures are the user-defined data type that can store large amount of data. By using structures, the data storing capacity of a program increases without any risk of losing it.
    - Structures can be used in all the places where the programmer wants to save large amounts of data, store and then reuse them in the further program.

CONSTRUCTION:

1) **LIBRARIES**

   To form a phonebook management system following libraries are used in the coding:

   i) <iostream>

   Iostream is used to form a direct relationship between cout and cin statements. All the data entered in this program from phone number to contact details will be done by this library.

   ii) <fstream>

   1) OF Stream
   - First, the file will be opened in append mode so that no loss of data may occur. Then, all the details of the contact will be written.

- Editing the details of contact from number to name will also be done in this mode. Append mode will save the previous data and allow the user to edit the details in after the previous details are saved so.
- To save the file append mode is closed and all the input details are saved until the user deletes them himself.

2) **IF Stream**
   If stream will display all the details of the program inputted by the user. If the user has not entered any details of the contact no details will be displayed. If stream uses open function to display all the details of the contact.

3) **F Stream**
   F stream will not only allow the user to input the specifics of the contact but it will also allow the user to read the details of after inputting them.

iii) **<string>**

- A string will save not only phone number but all contact details will also be of the string form.
- For this instance, length function will be used that will compare lengths by using ASCII Codes.
- Moreover, we will use "if else" statements to get a number of 11 digits. If the number is greater than or less than this limit, a prompt statement will tell the user to revise his number. As for as, contact name is concerned, it will be given enough space so that user can enter full name of the contact.
- The strings will be used to declare contact name and contact number in the start and will further be used in the body.

2) **IF ELSE STATEMENTS**
   In this program, if else statements are used at following places:

- First of all, the phone number inputted by the user should be of 11 digits. It means that if the user enters less or greater numbers than 11, a prompt statement will be displayed that will allow the user to get aware of his error.
- Secondly, the phone number must consist of integer values while contact details must be in character notation. If user performs any mistake in entering his number or contact details, the user will be asked to revise the inputted values again. As for as, details of contact are considered, it will be given enough space so that full name of the contact can be inputted without any error.
- If else statements are also used in the to make the user aware if his input contact already exists or not. If the user will enter contact details of a previously saved contact, a prompt statement will be displayed that will let him know.

3) **LOOPS**

In this program, loops are used to repeat the function until it falsifies. In this program loops are used in the following places:

i) Getline function to read the program if the number of digits are according to the limit.
ii) Save contact details in a file.
iii) Increment or decrement the number of contacts added after saving the number.
iv) To count the number of contacts entered by the user.
v) Edit the contact on user’s command.
vi) Check if the given number is already added or not.
vii) Search contacts by entering name or number from user.
viii) Display all contact list.
ix) Delete any contact.
x) Continue bringing menu list after the user has performed one task.

In all these tasks loops are used to display arrays in which the number is saved. This also increases our knowledge of using unidimensional loops.

4) **FILE HANDLING**

In this program, file handling will be used at following places:

- At first, to write something in the file, append mode will be used that will open the file in append mode. In this mode, writing of contact details are possible and so save them append mode is the preference. After the file is closed, the append mode will automatically save all the details of the contact inputted by the user.
- Secondly, to read what the user has written, open function will be used that will allow the user to read all the contact details.

5) **ARRAYS**

In this program, arrays are used to store input number and then display them in the output. In string form, arrays are declared to store input contact from the user and then at the end display them by writing in the file.

6) **MODULAR PROGRAMMING**

In this program, modular programming is used to divide whole program in sub-parts and separate functions are declared which work independently to make the program easier and simpler. Following functions are declared in this program:

a. Adding Contact
b. Save to File
c. Show all Contacts
d. Edit Contact
e. Append to File
f. Reading Contact
g. Check Name
h. Check Number
i. Delete Contact

7) **STRUCTURES**
As far as this program is considered structures are used to store all the details of the contact not only in write mode but read mode is also in its scope.

PROGRAMMING:

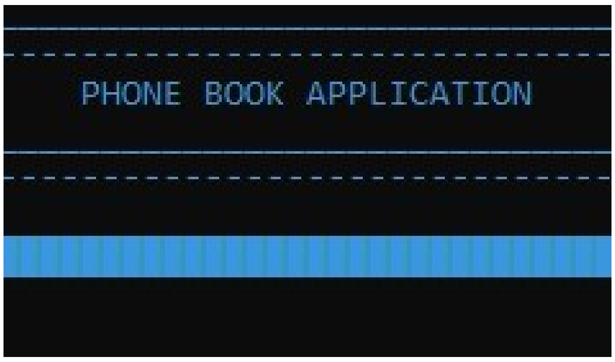
```
#include <iostream>
#include <fstream>
#include <string>
#include<windows.h>

using namespace std;

const int MAX_NAMES = 100;
const string PHONEBOOK_FILENAME = "phonebook.txt";

struct Person
{
    string name;
    string phone;

    Person()
    {
        name = "";
        phone = "";
    }

    bool Matches(string x)
    {
        return (name.find(x) != string::npos);
    }
};

void ReadAllPeople(Person people[], int& num_people)
{
    ifstream fin;
    fin.open(PHONEBOOK_FILENAME.c_str(), ios::app);
    if (fin.fail())
    {
        cout << "Unable to open file " << PHONEBOOK_FILENAME << endl;
        return;
    }

    int i = 0;
    getline(fin, people[i].name);
    while (!fin.eof() && i < MAX_NAMES)
    {
        getline(fin, people[i].phone);

        i++;

        if (i < MAX_NAMES)
            getline(fin, people[i].name);
    }

    num_people = i;

    fin.close();
```

```
}

void SaveToFile(const Person people[], int num_people)
{
    ofstream fout;

    fout.open(PHONEBOOK_FILENAME.c_str());

    for (int i = 0; i < num_people; i++)
    {
        fout << people[i].name << endl;
        fout << people[i].phone << endl;
    }
}

void AppendToFile(Person person)
{
    ofstream fin;
    fin.open(PHONEBOOK_FILENAME.c_str(), ios::app);

    fin << person.name << endl;
    fin << person.phone << endl;

    fin.close();
}

void add_contact(Person people[], int& num_people)
{
    Person person;

    cout << "\nEnter a name to be added: ";
    getline(cin, person.name);

    cout << "Enter the person's number: ";
    getline(cin, person.phone);

    for (int i = 0; i < num_people; i++)
    {
        if (i + 1 == num_people)
            people[num_people] = person;
    }
    num_people++;

    AppendToFile(person);
    cout << "\n\nName has been added. " << endl;
}

void edit_contact(Person people[], int num_people)
{
    Person person;
    int count;

    cout << "Enter name to change: ";
    getline(cin, person.name);

    for (count = 0; count < num_people; count++)
    {
        if (people[count].Matches(person.name))
        {
```

```cpp
            cout << endl
                << people[count].name << endl;

            cout << "Current number: " << people[count].phone;

            cout << "\nEnter New number: ";
            getline(cin, people[count].phone);

            SaveToFile(people, num_people);

            cout << "\n\nNew number Saved!";

            return;
        }
    }

    if (count = num_people)
        cout << "\nName not found.";
}

int check_name(Person people[], int num_people, string x)
{
    for (int i = 0; i < num_people; i++)
    {
        if (x == people[i].name)
        {
            return i;
        }
    }
    return -1;
}

int check_number(Person people[], int num_people, string x)
{
    for (int i = 0; i < num_people; i++)
    {
        if (x == people[i].phone)
        {
            return i;
        }
    }
    return -1;
}

void find_contact(Person people[], int num_people)
{
    int ans, count = 0;
    Person person;

    cout << "1. Find by contact name" << endl;
    cout << "2. Find by contact number" << endl;
    cin >> ans;

    cin.ignore();
    if (ans != 1 && ans != 2)
    {
        cout << "Invalid Choice";
        return;
    }
    else if (ans == 1)
    {
        cout << "\nEnter name to search : [Enter '0' to exit]\n";
        getline(cin, person.name);

        if (person.name.length() == 1 && person.name == "0")
            return;

        for (int i = 0; i < num_people; i++)
        {
            if (person.name == people[i].name)
            {
                cout << "Contact Found!\n";
                cout << people[i].name << " - " << people[i].phone << endl;
                count++;
            }
        }
        if (count == 0)
            cout << "Contact Not Found!\n";
    }
    else
    {
        cout << "\nEnter number to search : [Enter '0' to exit]\n";
        getline(cin, person.phone);

        if (person.phone.length() == 1 && person.phone == "0")
            return;

        for (int i = 0; i < num_people; i++)
        {
            if (person.phone == people[i].phone)
            {
                cout << "Contact Found!\n";
                cout << people[i].name << " - " << people[i].phone << endl;
                count++;
            }
        }
        if (count == 0)
            cout << "Contact Not Found!\n";
    }
}

void delete_contact(Person people[], int& num_people)
{
    Person person;

    int ans, check;
    cout << "1. Delete by contact name" << endl;
    cout << "2. Delete by contact number" << endl;
    cin >> ans;

    cin.ignore();
    if (ans != 1 && ans != 2)
    {
        cout << "Invalid Choice";
        return;
    }
    else if (ans == 1)
    {
        cout << "Enter contact name to delete: ";
        getline(cin, person.name);
        check = check_name(people, num_people, person.name);
    }
    else if (ans == 2)
    {
        cout << "Enter contact number to delete: ";
        getline(cin, person.phone);
        check = check_number(people, num_people, person.phone);
    }
    if (check == -1)
    {
        cout << "Contact Not Found!\n";
        return;
    }
    for (int i = check; i < num_people - 1; i++)
    {
        people[i].name = people[i + 1].name;
        people[i].phone = people[i + 1].phone;
    }
    num_people--;
    SaveToFile(people, num_people);
    cout << "Contact Deleted!\n";
}

void show_all_contacts(const Person people[], int num_people)
{
    cout << "\n\nAll records:\n";
    for (int i = 0; i < num_people; i++)
        cout << people[i].name << " - " << people[i].phone << endl;
}
    void start()
    {
        system("Color 03");
        cout << "\n\n\n\n\n\n";
        cout << "\t\t\t\t ____________________________________\n";
        cout << "\t\t\t\t ------------------------------------\n";
        cout << "\t\t\t\t        PHONE BOOK APPLICATION\n";
        cout << "\t\t\t\t ____________________________________\n";
        cout << "\t\t\t\t ------------------------------------\n\n";

        cout << "\t\t\t\t ";
        char x = 219;
        for (int i = 0; i < 35; i++)
        {
            cout << x;
            if (i < 10)
                Sleep(300);
            if (i >= 10 && i < 20)
                Sleep(150);
            if (i >= 10)
                Sleep(25);
        }
        system("cls");
    }

int main()
{
    start();
    Person people[MAX_NAMES];
    int choice, num_people;

    ReadAllPeople(people, num_people);

    do
    {

        system("color 03");

        cout << endl;
        cout << " ===================================================\n";
        cout << " --------------- PHONEBOOK DIRECTORY ---------------\n";
        cout << " ===================================================\n";
        cout << endl;
        cout << " :::::::::::::::::::: MAIN MENU ::::::::::::::::::::\n";
        cout << endl;
        cout << " 1. View Contact List.\n";
        cout << " 2. Add a Contact Number.\n";
        cout << " 3. Modify a Contact Number.\n";
        cout << " 4. Delete a Contact Number.\n";
        cout << " 5. Search a Contact Number.\n";
        cout << " 6. Exit Phonebook\n";
        cout << endl;
        cout << " ==================================================\n";
        cout << " Enter Your Choice: ";
        cin >> choice;

        switch (choice)
        {
        case 1:
            system("CLS");
            show_all_contacts(people, num_people);
            break;
        case 2:
            cin.ignore();
            system("CLS");
            add_contact(people, num_people);
            break;
        case 3:
            cin.ignore();
            edit_contact(people, num_people);
            break;
        case 4:
            delete_contact(people, num_people);
            break;
        case 5:
            cin.ignore();
            find_contact(people, num_people);
            break;
        }
    } while (choice != 6);
}
```

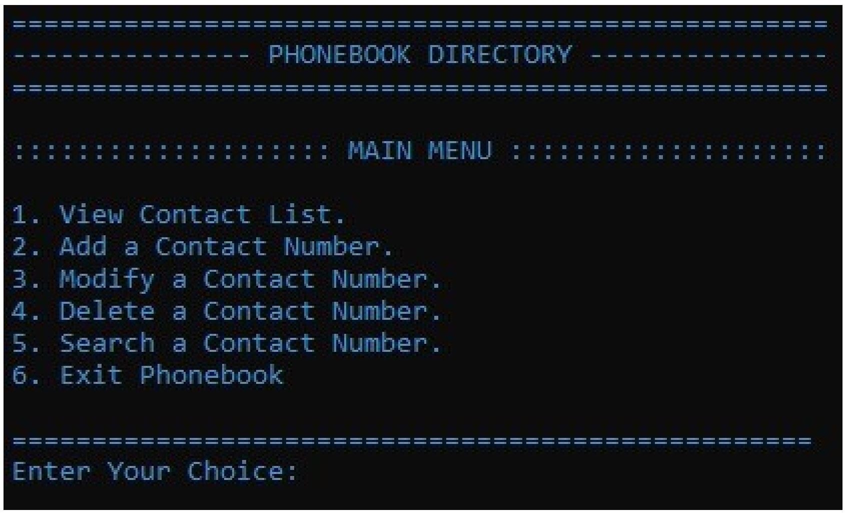
OUTPUT:

Loading Menu:



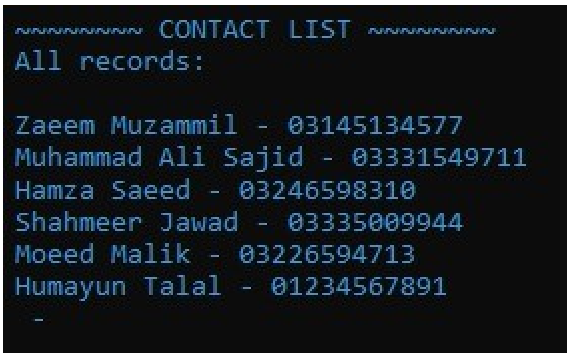
```
------------------------------------
    PHONE BOOK APPLICATION
------------------------------------
```

MAIN MENU:



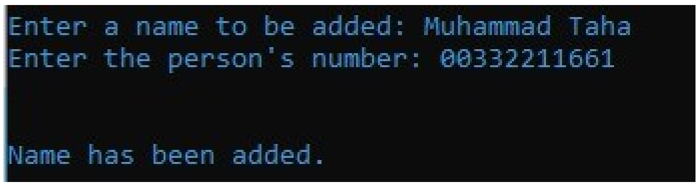
```
===================================================
--------------- PHONEBOOK DIRECTORY ---------------
===================================================

:::::::::::::::::::: MAIN MENU ::::::::::::::::::::

1. View Contact List.
2. Add a Contact Number.
3. Modify a Contact Number.
4. Delete a Contact Number.
5. Search a Contact Number.
6. Exit Phonebook

==================================================
Enter Your Choice:
```

(1) VIEW CONTACT LIST



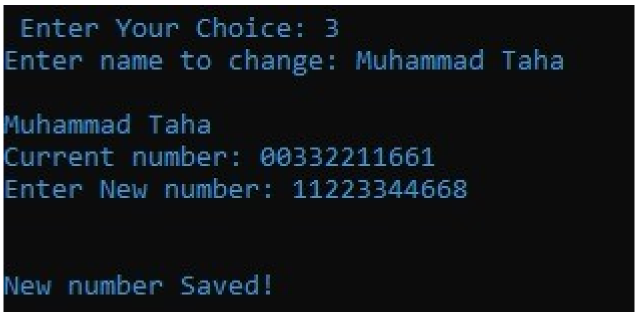
```
~~~~~~~~ CONTACT LIST ~~~~~~~~
All records:

Zaeem Muzammil - 03145134577
Muhammad Ali Sajid - 03331549711
Hamza Saeed - 03246598310
Shahmeer Jawad - 03335009944
Moeed Malik - 03226594713
Humayun Talal - 01234567891
 -
```

(2) ADD CONTACT

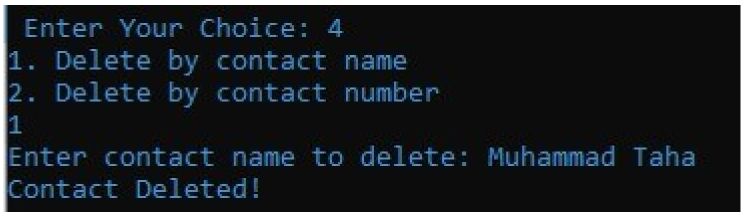


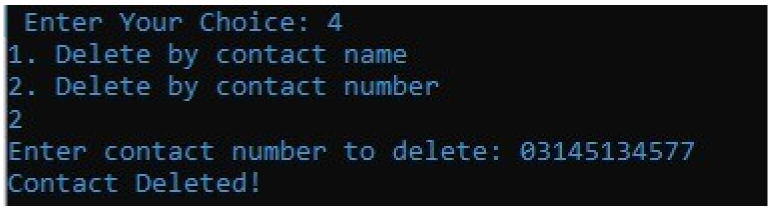
```
Enter a name to be added: Muhammad Taha
Enter the person's number: 00332211661


Name has been added.
```

(3) MODIFY A CONTACT NUMBER

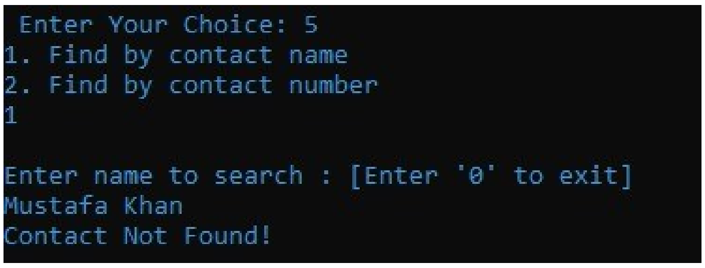

```
 Enter Your Choice: 3
Enter name to change: Muhammad Taha

Muhammad Taha
Current number: 00332211661
Enter New number: 11223344668


New number Saved!
```

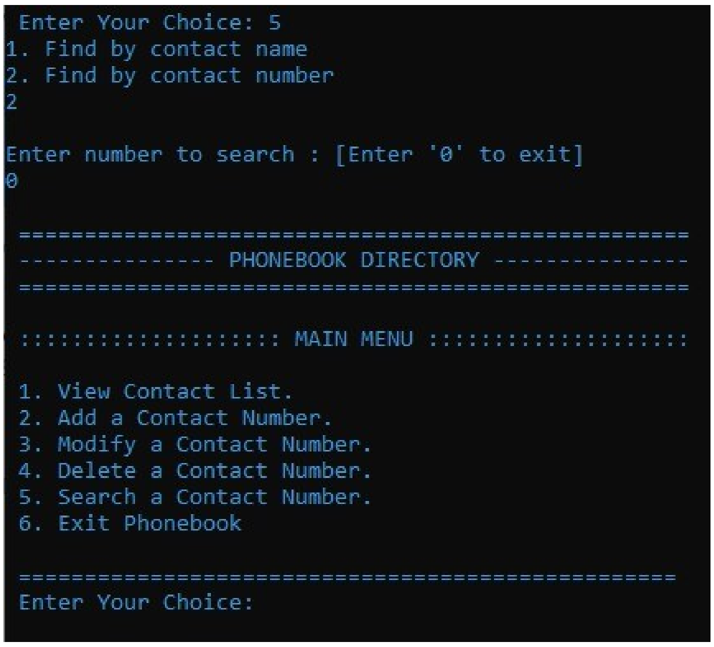
(4) DELETE A CONTACT NUMBER


```
 Enter Your Choice: 4
1. Delete by contact name
2. Delete by contact number
1
Enter contact name to delete: Muhammad Taha
Contact Deleted!
```

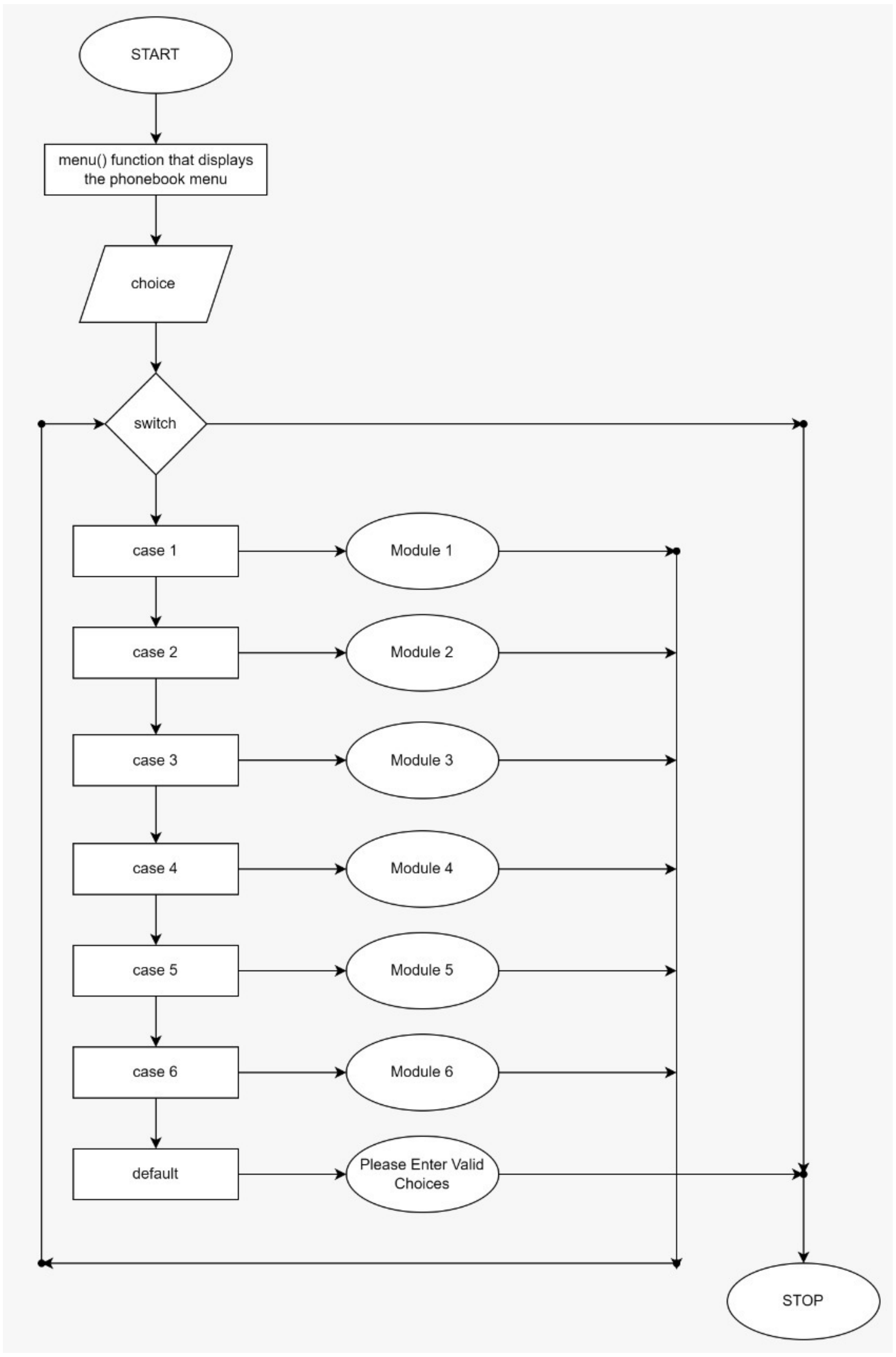

```
 Enter Your Choice: 4
1. Delete by contact name
2. Delete by contact number
2
Enter contact number to delete: 03145134577
Contact Deleted!
```

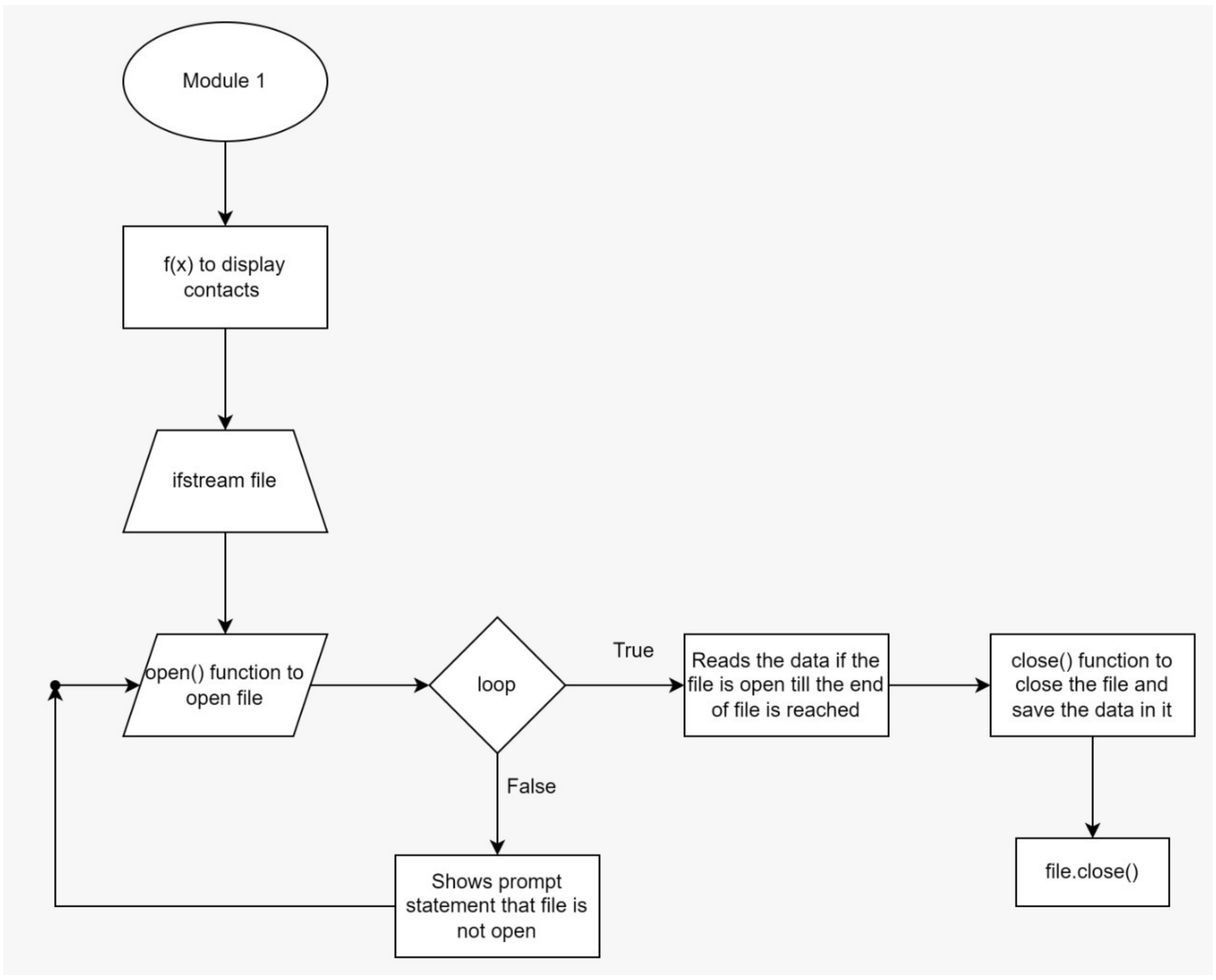
(5) SEARCH A CONTACT NUMBER

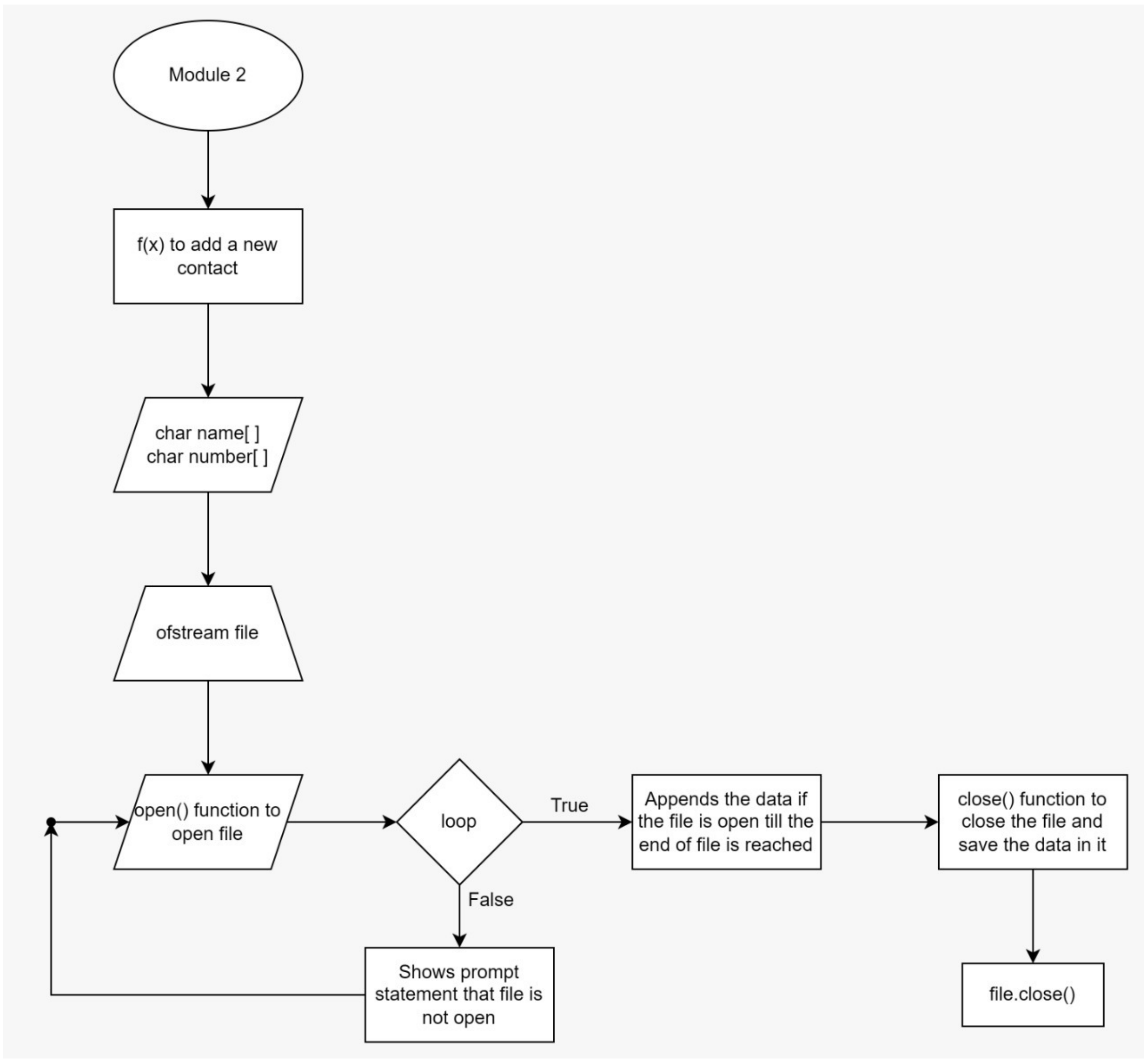

```
 Enter Your Choice: 5
1. Find by contact name
2. Find by contact number
1

Enter name to search : [Enter '0' to exit]
Mustafa Khan
Contact Not Found!
```

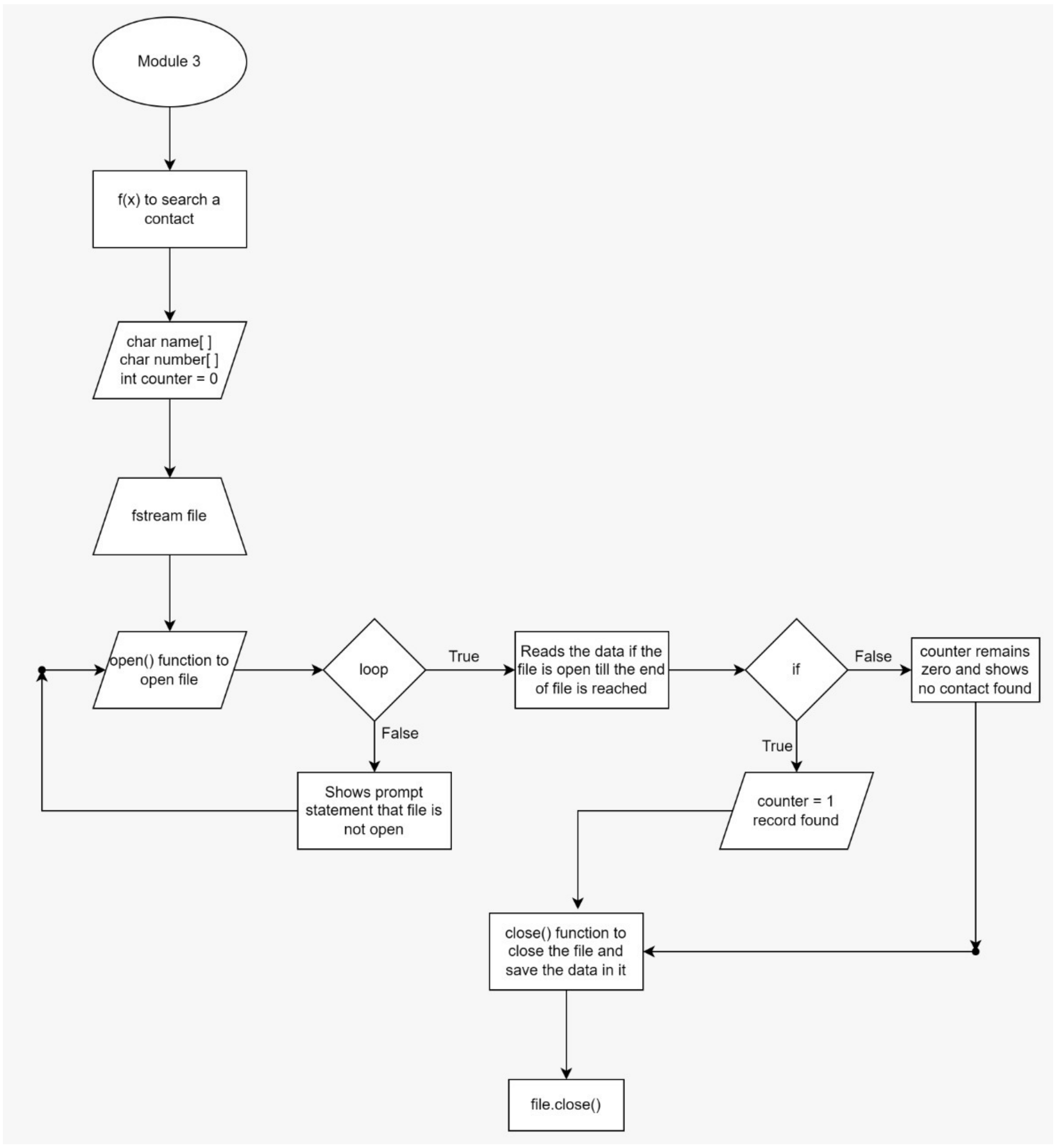

```
 Enter Your Choice: 5
1. Find by contact name
2. Find by contact number
2

Enter number to search : [Enter '0' to exit]
0

 ===================================================
 --------------- PHONEBOOK DIRECTORY ---------------
 ===================================================

 :::::::::::::::::::: MAIN MENU ::::::::::::::::::::

 1. View Contact List.
 2. Add a Contact Number.
 3. Modify a Contact Number.
 4. Delete a Contact Number.
 5. Search a Contact Number.
 6. Exit Phonebook

 ==================================================
 Enter Your Choice:
```

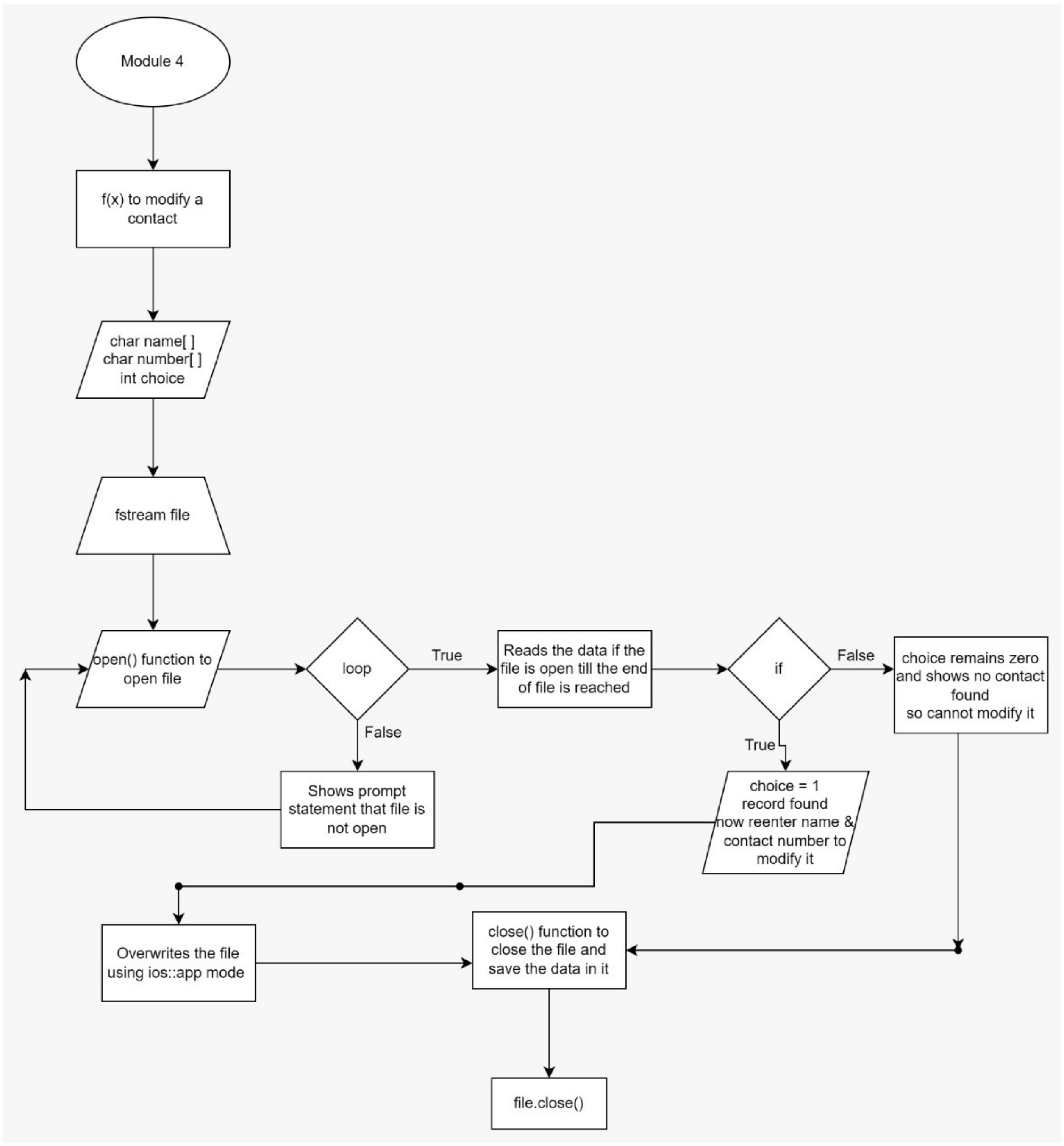
FLOW CHARTS:

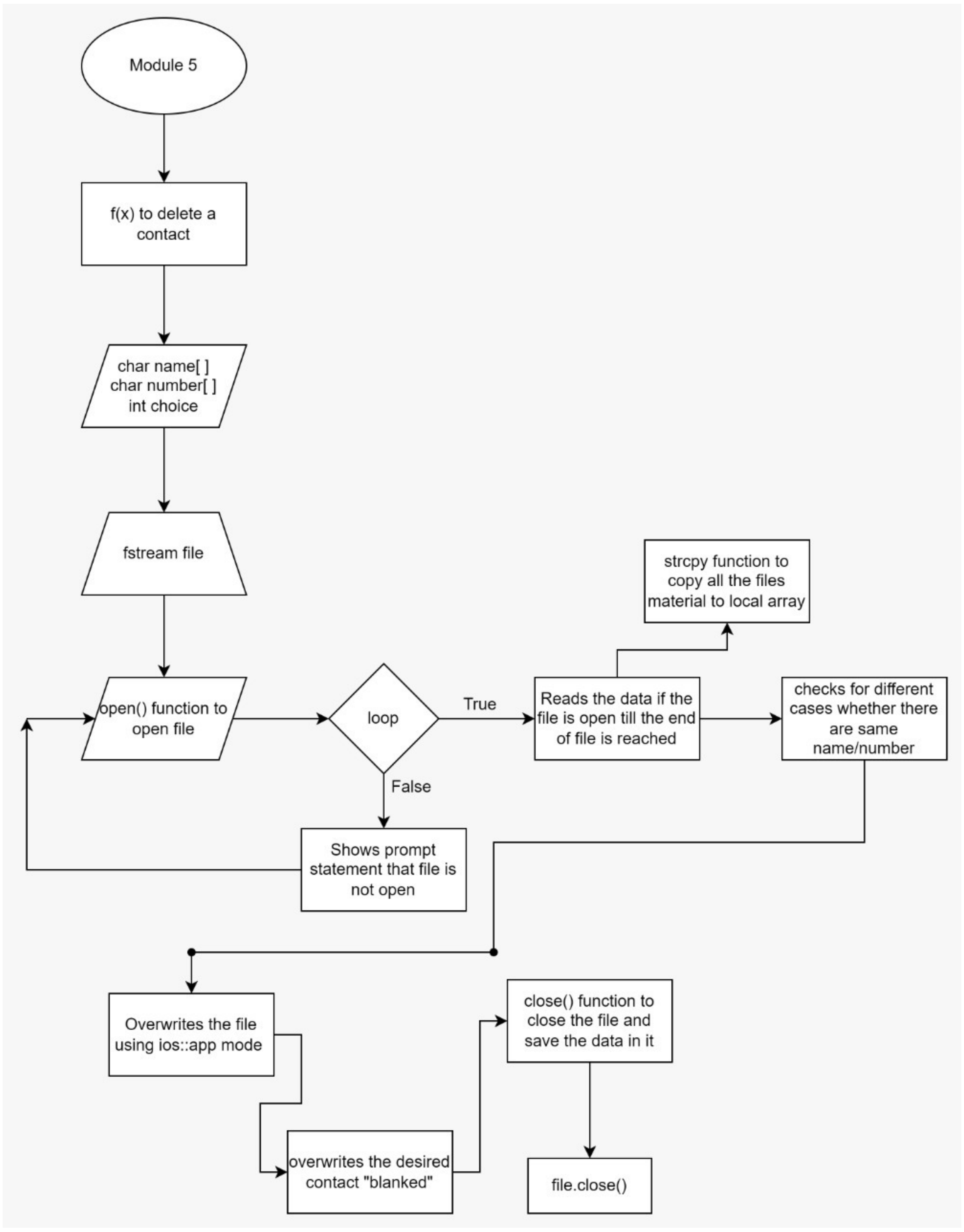

START
menu() function that displays the phonebook menu
choice
switch
case 1
Module 1
case 2
Module 2
case 3
Module 3
case 4
Module 4
case 5
Module 5
case 6
Module 6
default
Please Enter Valid Choices
STOP


Module 1
f(x) to display contacts
ifstream file
open() function to open file
loop
True
Reads the data if the file is open till the end of file is reached
close() function to close the file and save the data in it
file.close()
False
Shows prompt statement that file is not open


Module 2
f(x) to add a new contact
char name[ ]
char number[ ]
ofstream file
open() function to open file
loop
True
False
Appends the data if the file is open till the end of file is reached
close() function to close the file and save the data in it
Shows prompt statement that file is not open
file.close()


Module 3
f(x) to search a contact
char name[ ]
char number[ ]
int counter = 0
fstream file
open() function to open file
loop
True
False
Shows prompt statement that file is not open
Reads the data if the file is open till the end of file is reached
if
False
True
counter remains zero and shows no contact found
counter = 1 record found
close() function to close the file and save the data in it
file.close()


Module 4
f(x) to modify a contact
char name[ ]
char number[ ]
int choice
fstream file
open() function to open file
loop
True
False
Shows prompt statement that file is not open
Reads the data if the file is open till the end of file is reached
if
False
True
choice remains zero and shows no contact found so cannot modify it
choice = 1 record found now reenter name & contact number to modify it
Overwrites the file using ios::app mode
close() function to close the file and save the data in it
file.close()


Module 5
f(x) to delete a contact
char name[ ]
char number[ ]
int choice
fstream file
open() function to open file
loop
True
False
strcpy function to copy all the files material to local array
Reads the data if the file is open till the end of file is reached
checks for different cases whether there are same name/number
Shows prompt statement that file is not open
Overwrites the file using ios::app mode
overwrites the desired contact "blanked"
close() function to close the file and save the data in it
file.close()

REFERENCES:

- https://youtu.be/xNJLe3m03Y4
- https://www.tutorialspoint.com/cplusplus/cpp_files_streams.htm
- https://www.programiz.com/cpp-programming/structure