ADDRESS BOOK
PROJECT REPORT

Submitted by

ABINAYA VINA [RA2211031010145]
RUKMA RAO [RA2211031010090]

Under the Guidance of

Mr. K. Manikandan

Assistant Professor, NWC

In partial satisfaction of the requirements for the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE ENGINEERING

with specialization in INFORMATION TECHNOLOGY

SCHOOL OF COMPUTING

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR - 603203

MAY 2023

SRM INSTITUTION OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR-603203

BONAFIDE CERTIFICATE

Certified that this Project Report titled **"SIMPLE ADDRESS BOOK"** is the bonafide work done by **ABINAYA VINA, [RA2211010010145]** and **RUKMA JYOTI PRAKASH RAO [RA2211031010090]**, who completed the project under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other work.

SIGNATURE Mr. K. Manikandan **OODP – Course Faculty** **Assistant Professor** Department of NWC SRMIST	**SIGNATURE** Dr. Annapurani. K **Head of the Department** Department of NWC SRMIST

TABLE OF CONTENTS

PROBLEM STATEMENT

As part of this mini project, my team-mate and I have decided to create a ‘Simple Address Book’ in C++.
The problem statement is as follows:
Q) To make an Address Book List Program in C++ that allows us to store the details of a person. Use various functions to insert features such as ‘Adding A New Contact’, ‘Deleting A Contact’, ’Updating Details Of Pre-Existing Contact’, ‘View All Contacts’, or, ‘Search Entry’.

The purpose and aim of the project are to learn the core concepts of C++.

The first stage in writing an address book programme in C++ is to specify the data structure for the contacts. A class that has variables for each piece of data, including first name, last name, address and phone number, can be used to accomplish this.

When the programme launches, it can read the contents of the file and add the saved data to the address book data structure. The programme should also have a search feature that enables users to locate individual contacts by name, or contact entry number. A linear search method that iterates through the list of contacts and compares the search criteria with each contact's information can be used to do this.

All things considered, developing an address book programme in C++ is a terrific approach to practise object-oriented programming and file I/O ideas while developing a practical utility for organising contacts.

MODULES OF THE PROJECT

Data structure: This module defines the structure of the address book, such as the fields for name, address and phone number. It could use classes, structures, or arrays to store and organize the data.

Input/output: This module handles the user input and output, such as displaying a menu, prompting for data, and printing the results. It could use functions like cout, cin, and getline to interact with the console.

Operations: This module performs the operations or actions that the user requests, such as adding a new contact, searching for a contact, editing a contact, or deleting a contact. It could use functions like insert, find, update, and erase to modify the data.

File handling: This module saves and loads the data from a file, so that the address book can be persistent across multiple executions. It could use functions like ofstream and ifstream to write and read from a text file.

The main() function: In the main function, there is the usage of a switch-case in order to select the necessary choice.

The addContacts() function: We store all the information in a file that is created when we add the first contact. The next time, the file is just appended with the previous data. The maximum number of contacts is 100.

The viewContacts() function: In the view contacts function, we first give the names of the address book columns and use a while loop to print the contacts information till the last available one and stop if it is at 100.

The searchContact() function: Here, first, we give the user a choice or filter for the search and based on the choice we use an if condition. The if condition, based on a choice, searches the file and if a match is found with the given and existing record, that match is returned to us.

The editContact() function: In this function, we take the user input of the entry number, and using it we search the file for that record.

If found, we print it on screen for confirmation. When users type “y” we proceed to give them the option to enter new information.

The deleteContact() function: It is similar to the editContact function till the confirmation part. If the entry is valid we delete the record and set the entry numbers accordingly. If the entry is not valid we don’t modify the data.

UML DIAGRAMS

Unified Modelling Language (UML) diagrams are visual representations used for modelling object-oriented software systems. UML diagrams provide a standardized way to visualize and communicate software designs, making them easier to understand and implement. There are two main types of UML diagrams: behavioral and structural. Structural UML diagrams describe the static structure of a system, including its components, classes, interfaces, and relationships between them. Behavioral UML diagrams describe the dynamic behavior of a system, including the interactions between objects and the changes in their states.

Structure Diagrams:

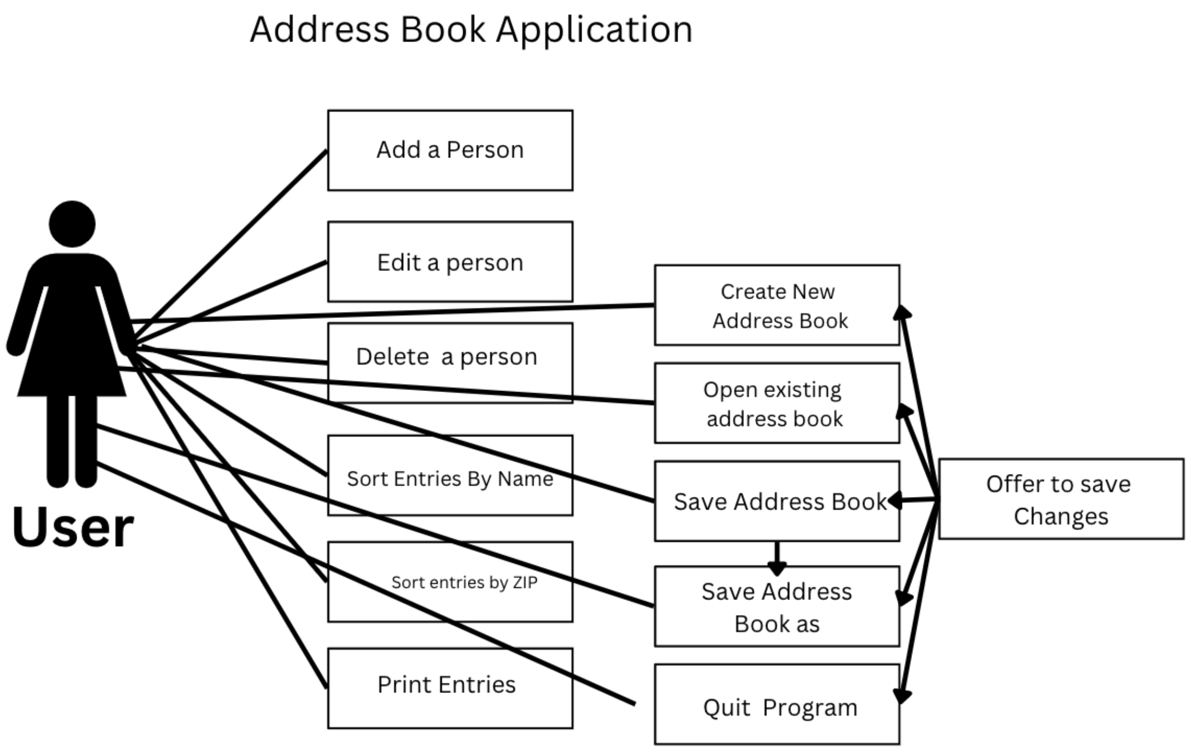
- Class Diagram
- Component Diagram
- Deployment Diagram
- Object Diagram
- Package Diagram

Behavioral Diagrams:

- Use Case Diagram
- Activity Diagram
- State Chart Diagram
- Sequence Diagram

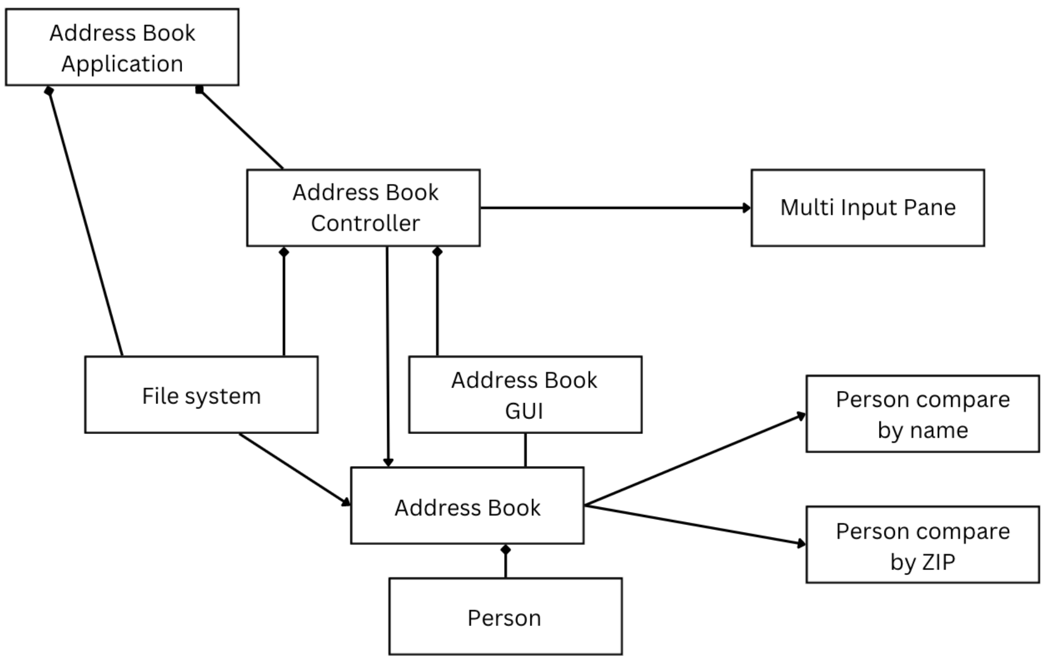
3.1) USE CASE DIAGRAM:

- A use case diagram for an address book in C++ shows actors such as users, the system, and possibly an external database. The diagram includes adding a contact, editing a contact, deleting a contact, searching for a contact, and displaying a list of contacts.



3.2)CLASS DIAGRAM

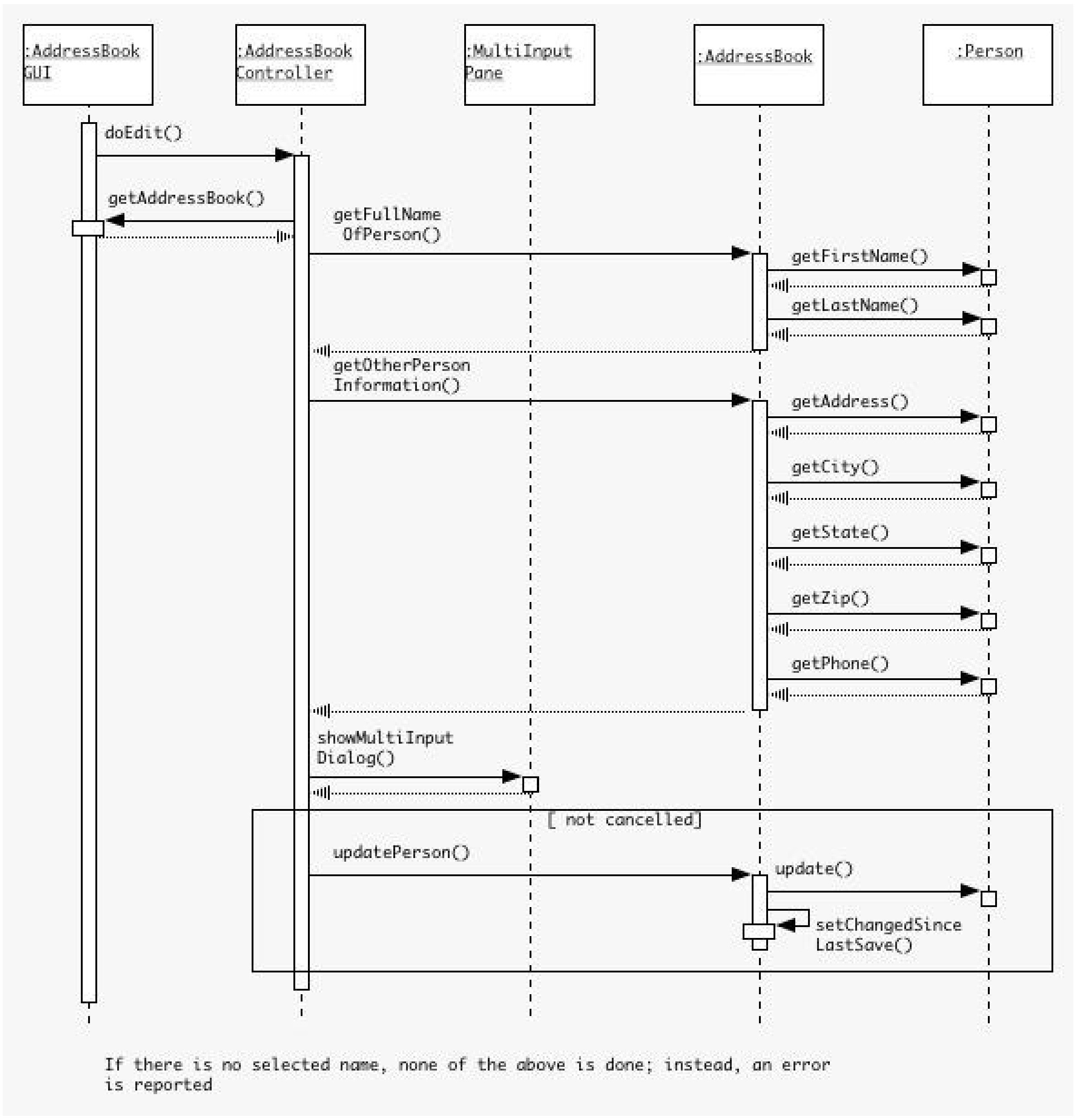
- A class diagram for an address book in C++ would typically show classes such as Contact, AddressBook, and possibly a Database class. The Contact class might have attributes such as name, phone number, and email,and methods for adding, editing, and deleting contacts. The AddressBook class would manage a collection of Contact objects and provide methods for searching and displaying contacts.



A sequence diagram for an address book in C++ shows the interaction between user and system components, including adding, searching, and deleting contacts. It illustrates the flow of messages and operations, highlighting the communication and behavior of the address book functionality.

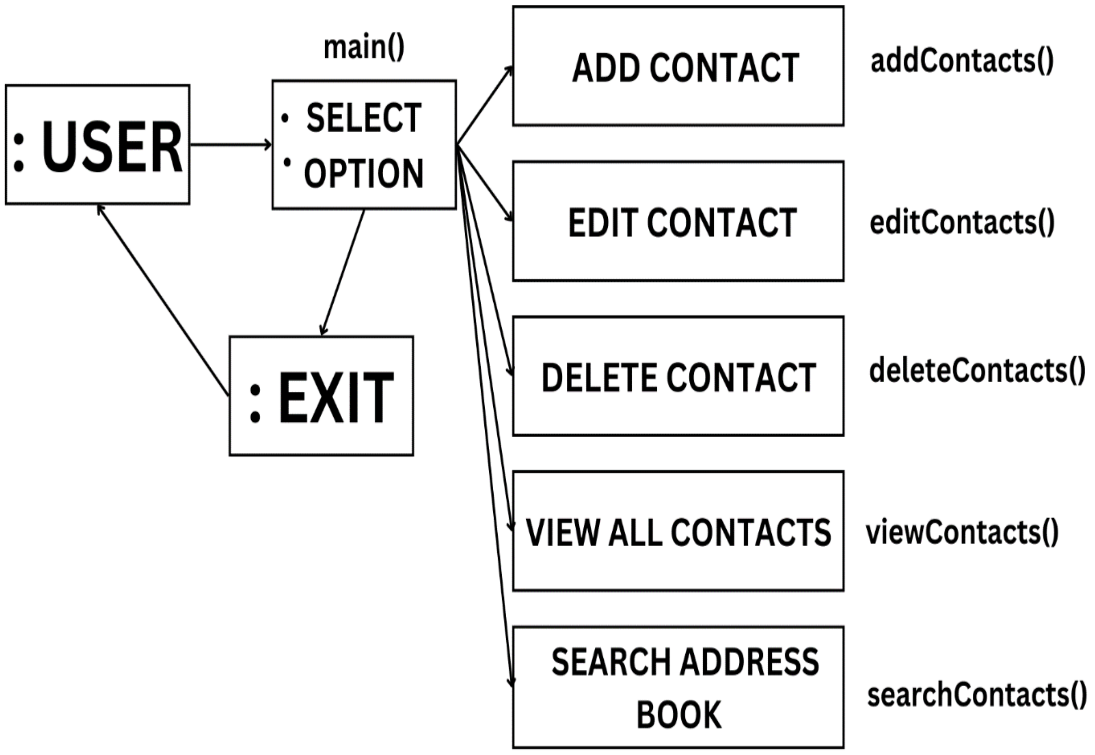
3.3) SEQUENCE DIAGRAM

- A class diagram for an address book in C++ would typically show classes such as Contact, AddressBook, and possibly a Database class. The Contact class might have attributes such as name, phone number, and email, and methods for adding, editing, and deleting contacts. The AddressBook class would manage a collection of Contact objects and provide methods for searching and displaying contacts.



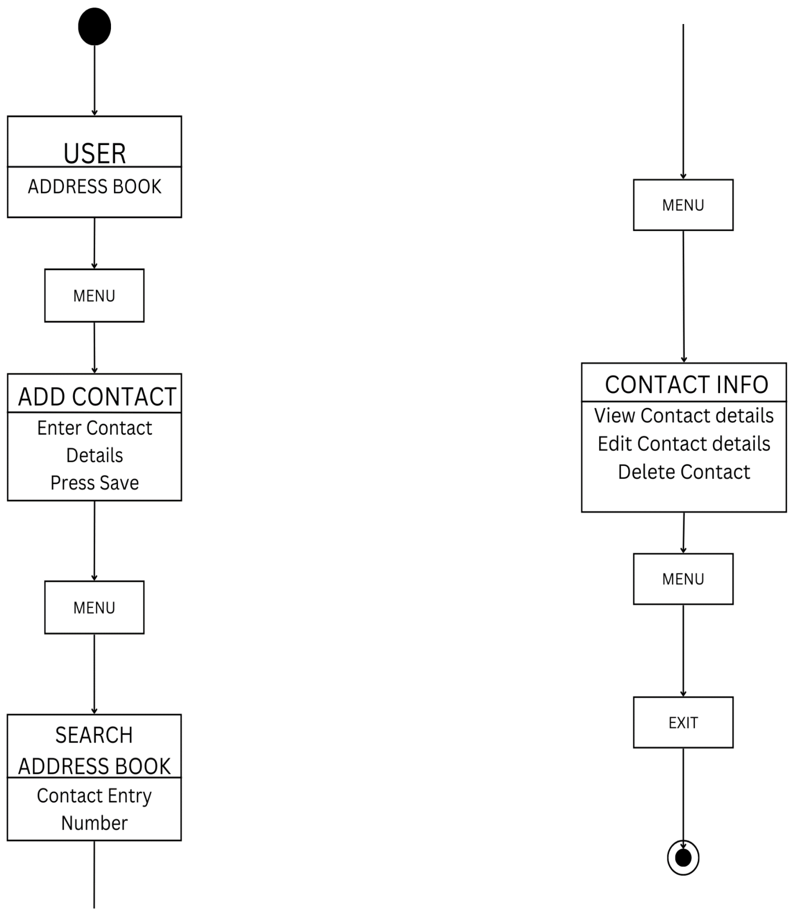
3.4) COLLABORATION DIAGRAM:

- A collaboration diagram for an address book in C++ would typically show objects such as the user interface, the AddressBook class, and possibly a Database class, and the messages exchanged between them. It would illustrate how objects work together to perform tasks such as adding, editing, deleting, searching for, or displaying contacts.



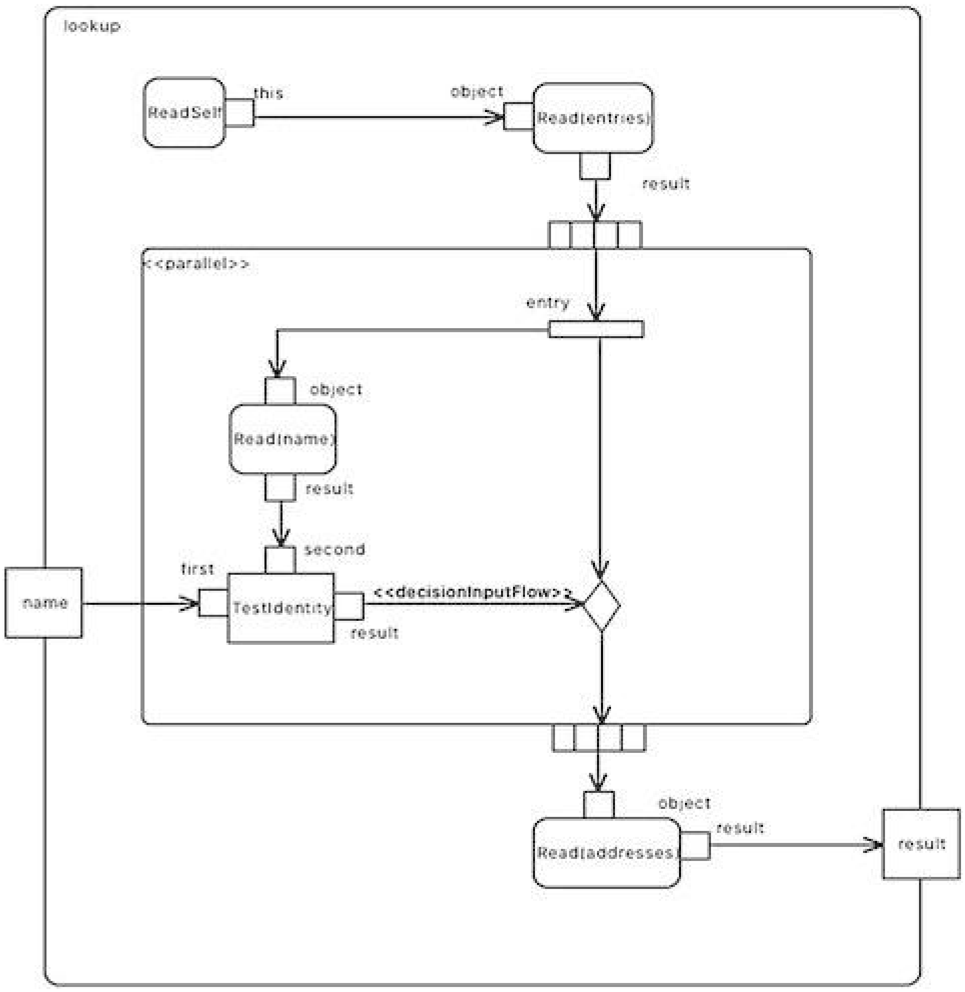
3.5) STATE CHART DIAGRAM:

- A state chart diagram for an address book in C++ could illustrate the different states and transitions of the program, such as adding, deleting, and editing contacts, as well as displaying and searching for specific entries. It could also show how the program handles errors and user inputs.



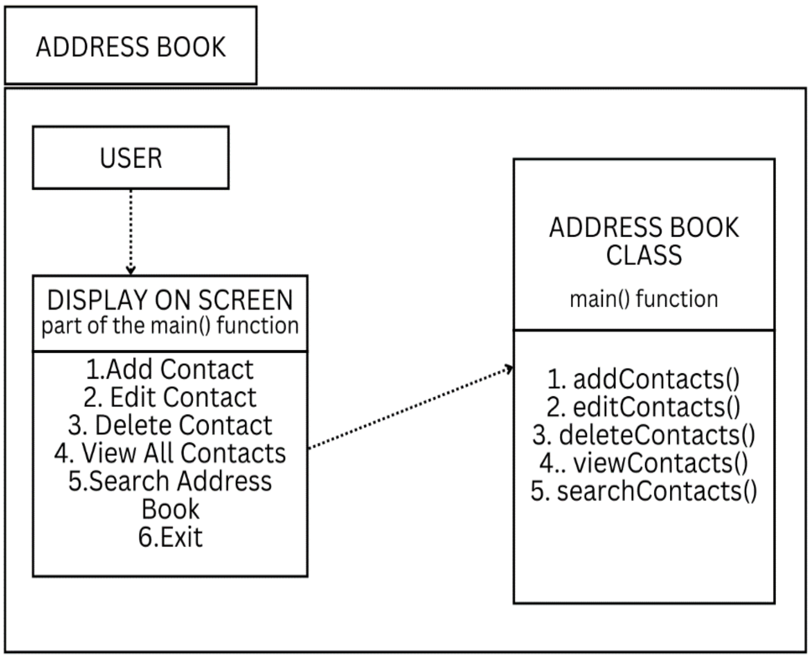
3.6) ACTIVITY DIAGRAM:

- An activity diagram for an address book in C++ could depict the sequence of actions and decision points involved in performing tasks such as adding or deleting contacts, searching for entries, and navigating the program's interface. It could also show how the program responds to user inputs and handles exceptions.



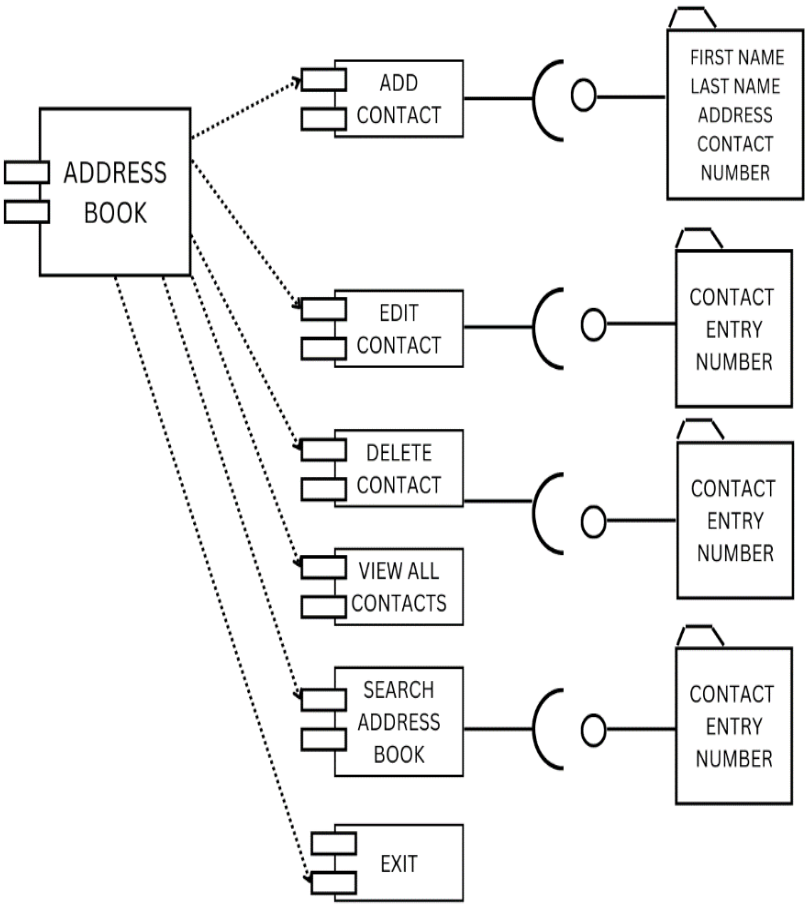
3.7) PACKAGE DIAGRAM:

- A package diagram for an address book in C++ could represent the organization of the program's code into logical units, such as classes and modules, that are responsible for different aspects of the program's functionality. It could also show how these units depend on each other and interact to achieve the program's goals.



3.8) COMPONENT DIAGRAM:

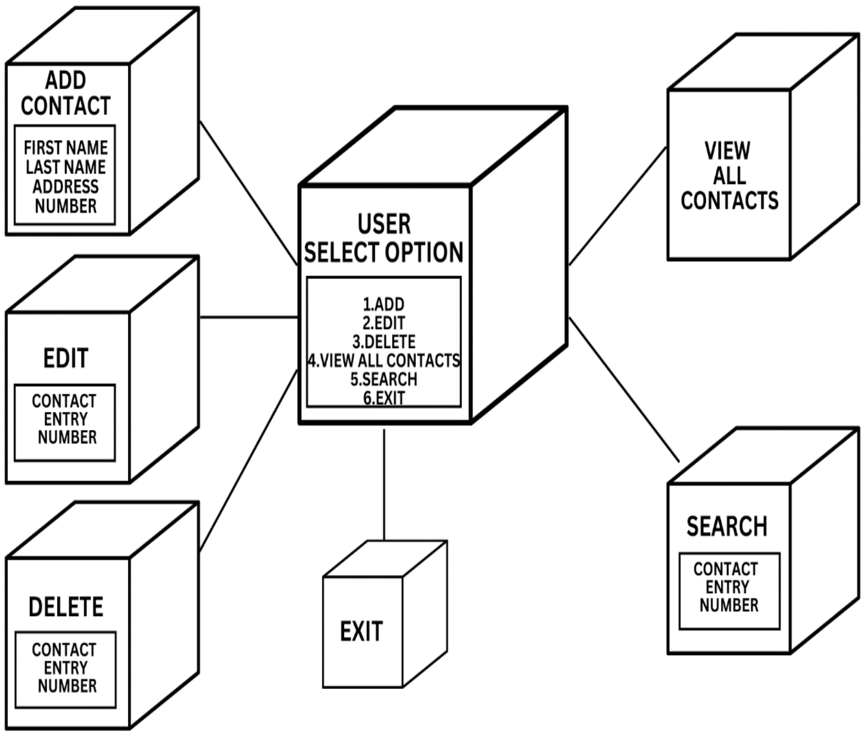
- A component diagram for an address book in C++ could illustrate the physical and logical components of the system, such as modules, libraries, and external dependencies, and how they interact to provide the program's functionality. It could also show how these components are deployed and distributed across different hardware or software platforms.



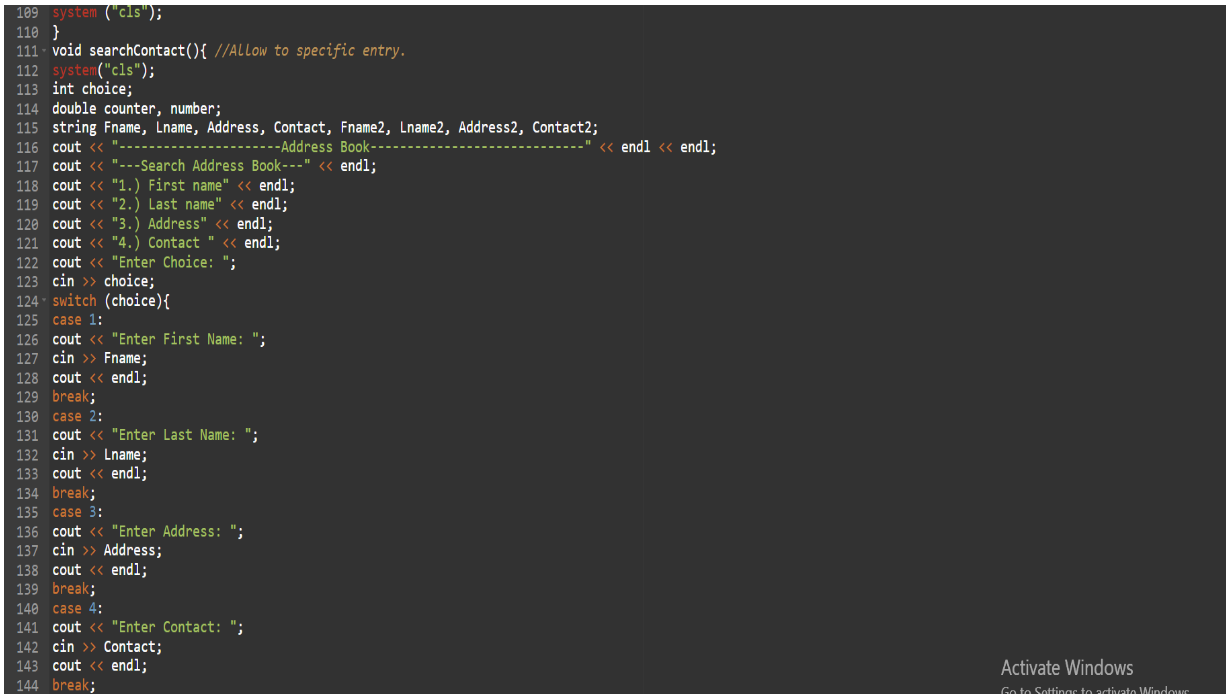
DESCRIPTION: The use case diagram for an address book in C++ shows actors such as users, the system, and possibly an external database. The diagram includes adding a contact, editing a contact, deleting a contact, searching for a contact, and displaying a list of contacts.

3.9) DEPLOYMENT DIAGRAM:

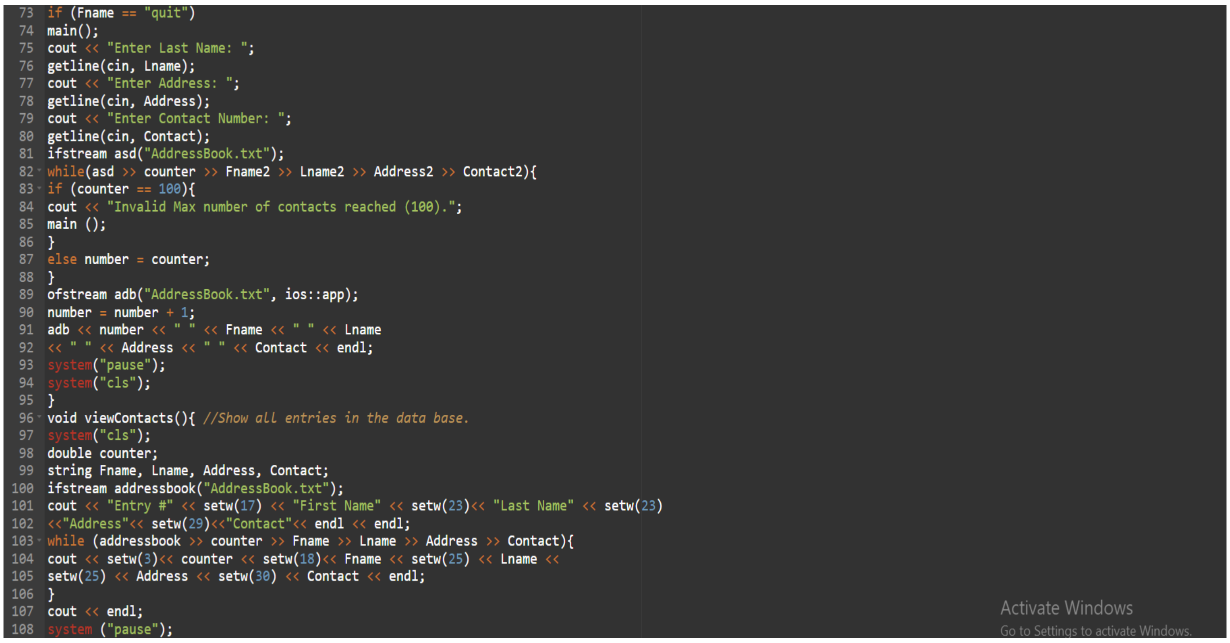
- A deployment diagram for an address book in C++ could illustrate the physical nodes, such as servers, workstations, or mobile devices, and the software components that are deployed on them. It could also show the connections and communication protocols used between these nodes to enable the program's functionality.



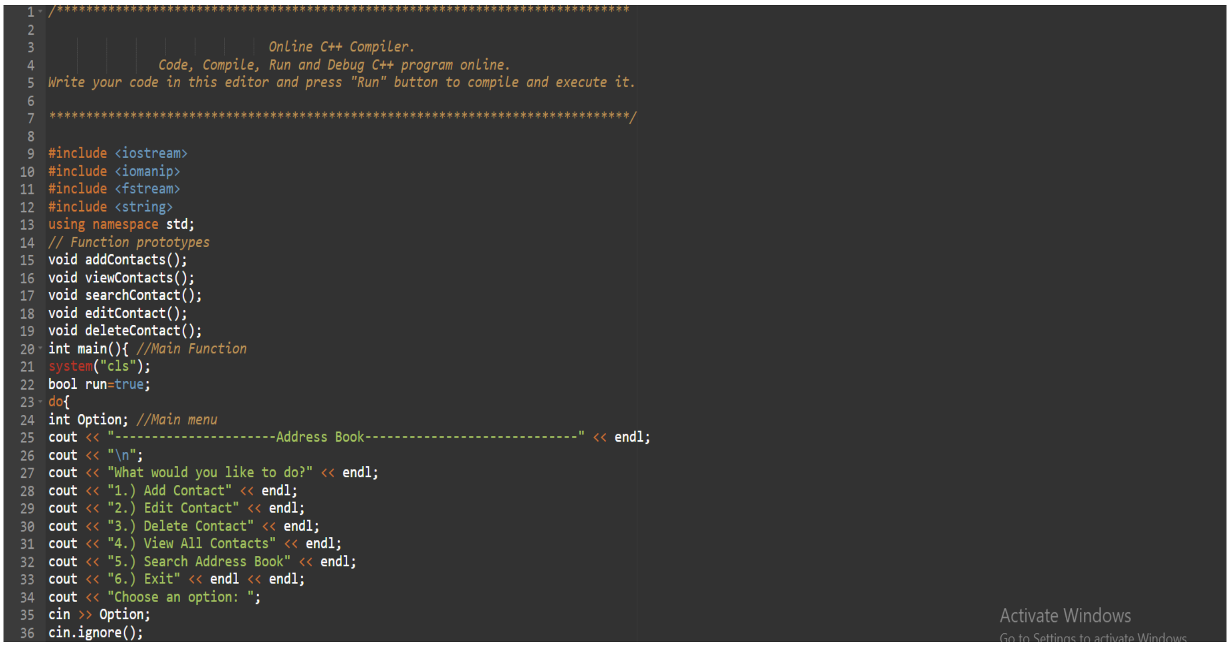
CODE



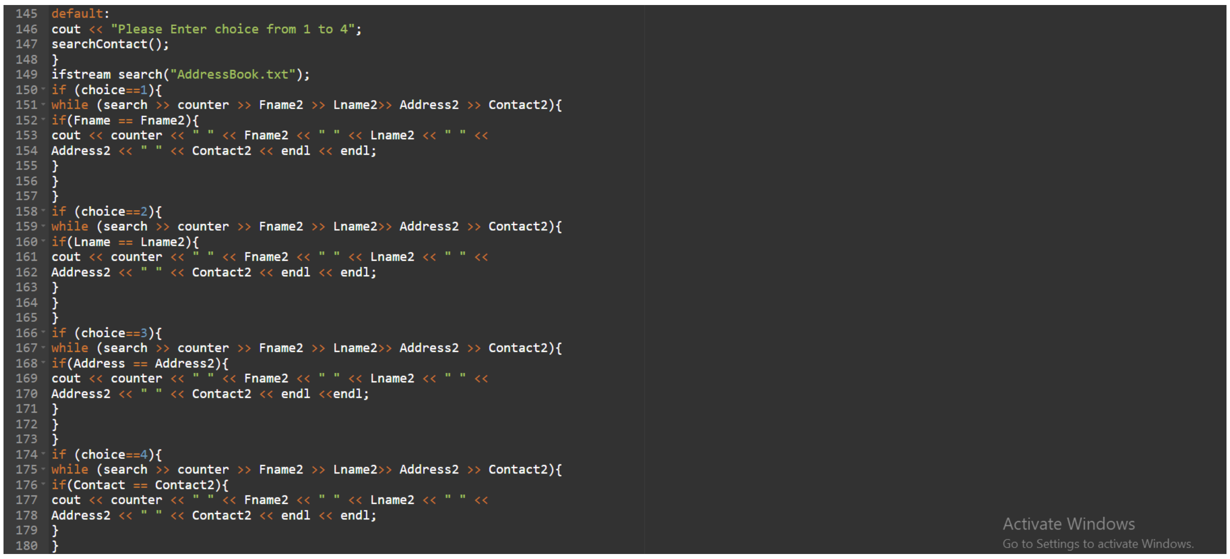
```
system ("cls");
}
void searchContact(){ //Allow to specific entry.
system("cls");
int choice;
double counter, number;
string Fname, Lname, Address, Contact, Fname2, Lname2, Address2, Contact2;
cout << "---------------------Address Book----------------------------" << endl << endl;
cout << "---Search Address Book---" << endl;
cout << "1.) First name" << endl;
cout << "2.) Last name" << endl;
cout << "3.) Address" << endl;
cout << "4.) Contact " << endl;
cout << "Enter Choice: ";
cin >> choice;
switch (choice){
case 1:
cout << "Enter First Name: ";
cin >> Fname;
cout << endl;
break;
case 2:
cout << "Enter Last Name: ";
cin >> Lname;
cout << endl;
break;
case 3:
cout << "Enter Address: ";
cin >> Address;
cout << endl;
break;
case 4:
cout << "Enter Contact: ";
cin >> Contact;
cout << endl;
break;
```



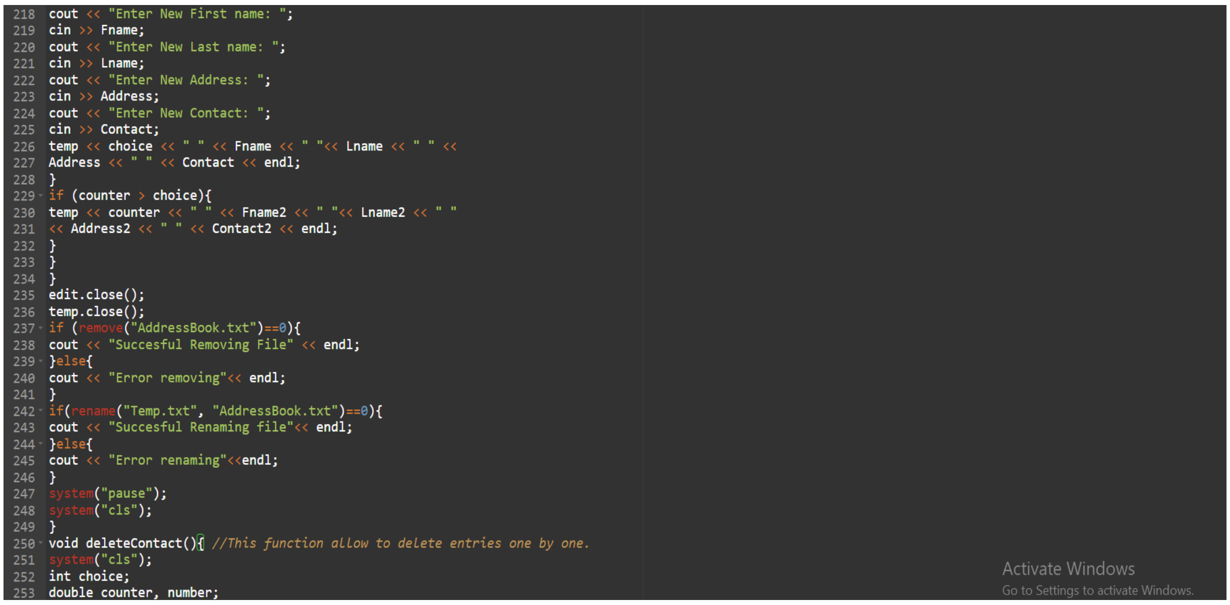
```
if (Fname == "quit")
main();
cout << "Enter Last Name: ";
getline(cin, Lname);
cout << "Enter Address: ";
getline(cin, Address);
cout << "Enter Contact Number: ";
getline(cin, Contact);
ifstream asd("AddressBook.txt");
while(asd >> counter >> Fname2 >> Lname2 >> Address2 >> Contact2){
if (counter == 100){
cout << "Invalid Max number of contacts reached (100).";
main ();
}
else number = counter;
}
ofstream adb("AddressBook.txt", ios::app);
number = number + 1;
adb << number << " " << Fname << " " << Lname
<< " " << Address << " " << Contact << endl;
system("pause");
system("cls");
}
void viewContacts(){ //Show all entries in the data base.
system("cls");
double counter;
string Fname, Lname, Address, Contact;
ifstream addressbook("AddressBook.txt");
cout << "Entry #" << setw(17) << "First Name" << setw(23)<< "Last Name" << setw(23)
<<"Address"<< setw(29)<<"Contact"<< endl << endl;
while (addressbook >> counter >> Fname >> Lname >> Address >> Contact){
cout << setw(3)<< counter << setw(18)<< Fname << setw(25) << Lname <<
setw(25) << Address << setw(30) << Contact << endl;
}
cout << endl;
system ("pause");
```



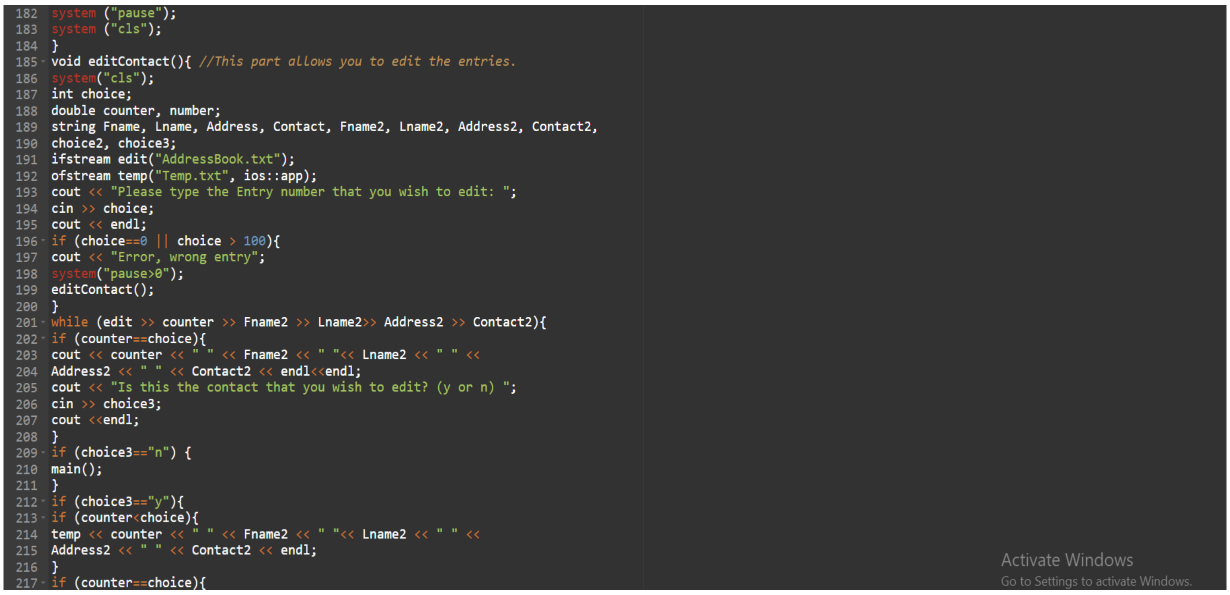
```
/******************************************************************************

                              Online C++ Compiler.
               Code, Compile, Run and Debug C++ program online.
Write your code in this editor and press "Run" button to compile and execute it.

*******************************************************************************/

#include <iostream>
#include <iomanip>
#include <fstream>
#include <string>
using namespace std;
// Function prototypes
void addContacts();
void viewContacts();
void searchContact();
void editContact();
void deleteContact();
int main(){ //Main Function
system("cls");
bool run=true;
do{
int Option; //Main menu
cout << "---------------------Address Book----------------------------" << endl;
cout << "\n";
cout << "What would you like to do?" << endl;
cout << "1.) Add Contact" << endl;
cout << "2.) Edit Contact" << endl;
cout << "3.) Delete Contact" << endl;
cout << "4.) View All Contacts" << endl;
cout << "5.) Search Address Book" << endl;
cout << "6.) Exit" << endl << endl;
cout << "Choose an option: ";
cin >> Option;
cin.ignore();
```



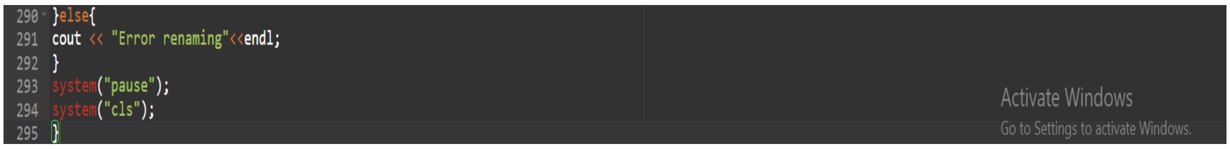
```
default:
cout << "Please Enter choice from 1 to 4";
searchContact();
}
ifstream search("AddressBook.txt");
if (choice==1){
while (search >> counter >> Fname2 >> Lname2>> Address2 >> Contact2){
if(Fname == Fname2){
cout << counter << " " << Fname2 << " " << Lname2 << " " <<
Address2 << " " << Contact2 << endl << endl;
}
}
}
if (choice==2){
while (search >> counter >> Fname2 >> Lname2>> Address2 >> Contact2){
if(Lname == Lname2){
cout << counter << " " << Fname2 << " " << Lname2 << " " <<
Address2 << " " << Contact2 << endl << endl;
}
}
}
if (choice==3){
while (search >> counter >> Fname2 >> Lname2>> Address2 >> Contact2){
if(Address == Address2){
cout << counter << " " << Fname2 << " " << Lname2 << " " <<
Address2 << " " << Contact2 << endl <<endl;
}
}
}
if (choice==4){
while (search >> counter >> Fname2 >> Lname2>> Address2 >> Contact2){
if(Contact == Contact2){
cout << counter << " " << Fname2 << " " << Lname2 << " " <<
Address2 << " " << Contact2 << endl << endl;
}
}
```



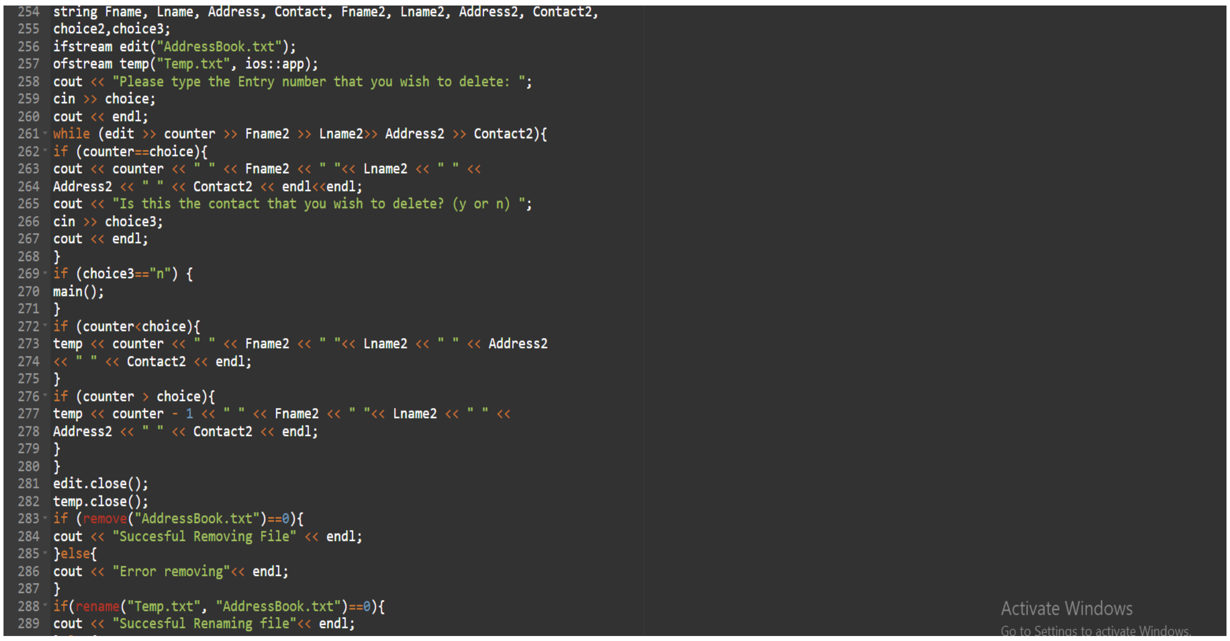
```cpp
cout << "Enter New First name: ";
cin >> Fname;
cout << "Enter New Last name: ";
cin >> Lname;
cout << "Enter New Address: ";
cin >> Address;
cout << "Enter New Contact: ";
cin >> Contact;
temp << choice << " " << Fname << " "<< Lname << " " <<
Address << " " << Contact << endl;
}
if (counter > choice){
temp << counter << " " << Fname2 << " "<< Lname2 << " "
<< Address2 << " " << Contact2 << endl;
}
}
}
edit.close();
temp.close();
if (remove("AddressBook.txt")==0){
cout << "Succesful Removing File" << endl;
}else{
cout << "Error removing"<< endl;
}
if(rename("Temp.txt", "AddressBook.txt")==0){
cout << "Succesful Renaming file"<< endl;
}else{
cout << "Error renaming"<<endl;
}
system("pause");
system("cls");
}
void deleteContact(){ //This function allow to delete entries one by one.
system("cls");
int choice;
double counter, number;
```



```cpp
system ("pause");
system ("cls");
}
void editContact(){ //This part allows you to edit the entries.
system("cls");
int choice;
double counter, number;
string Fname, Lname, Address, Contact, Fname2, Lname2, Address2, Contact2,
choice2, choice3;
ifstream edit("AddressBook.txt");
ofstream temp("Temp.txt", ios::app);
cout << "Please type the Entry number that you wish to edit: ";
cin >> choice;
cout << endl;
if (choice==0 || choice > 100){
cout << "Error, wrong entry";
system("pause>0");
editContact();
}
while (edit >> counter >> Fname2 >> Lname2>> Address2 >> Contact2){
if (counter==choice){
cout << counter << " " << Fname2 << " "<< Lname2 << " " <<
Address2 << " " << Contact2 << endl<<endl;
cout << "Is this the contact that you wish to edit? (y or n) ";
cin >> choice3;
cout <<endl;
}
if (choice3=="n") {
main();
}
if (choice3=="y"){
if (counter<choice){
temp << counter << " " << Fname2 << " "<< Lname2 << " " <<
Address2 << " " << Contact2 << endl;
}
if (counter==choice){
```

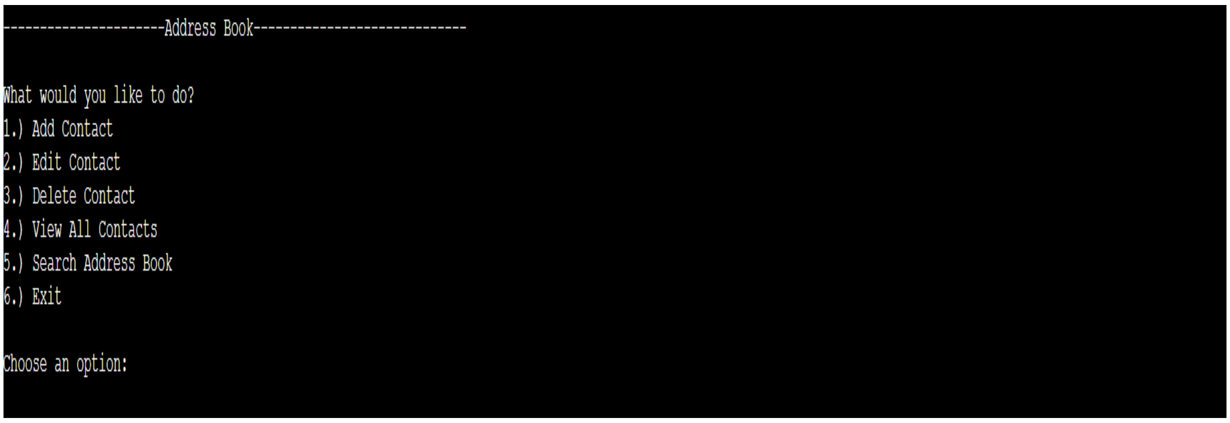


```
}else{
cout << "Error renaming"<<endl;
}
system("pause");
system("cls");
}
```

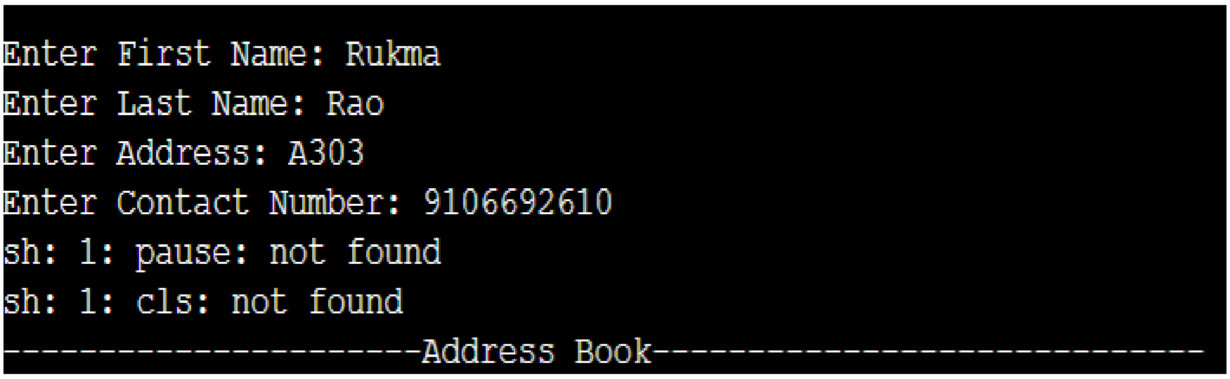


```
string Fname, Lname, Address, Contact, Fname2, Lname2, Address2, Contact2,
choice2,choice3;
ifstream edit("AddressBook.txt");
ofstream temp("Temp.txt", ios::app);
cout << "Please type the Entry number that you wish to delete: ";
cin >> choice;
cout << endl;
while (edit >> counter >> Fname2 >> Lname2>> Address2 >> Contact2){
if (counter==choice){
cout << counter << " " << Fname2 << " "<< Lname2 << " " <<
Address2 << " " << Contact2 << endl<<endl;
cout << "Is this the contact that you wish to delete? (y or n) ";
cin >> choice3;
cout << endl;
}
if (choice3=="n") {
main();
}
if (counter<choice){
temp << counter << " " << Fname2 << " "<< Lname2 << " " << Address2
<< " " << Contact2 << endl;
}
if (counter > choice){
temp << counter - 1 << " " << Fname2 << " "<< Lname2 << " " <<
Address2 << " " << Contact2 << endl;
}
}
edit.close();
temp.close();
if (remove("AddressBook.txt")==0){
cout << "Succesful Removing File" << endl;
}else{
cout << "Error removing"<< endl;
}
if(rename("Temp.txt", "AddressBook.txt")==0){
cout << "Succesful Renaming file"<< endl;
```

OUTPUT



```
---------------------Address Book----------------------------

What would you like to do?
1.) Add Contact
2.) Edit Contact
3.) Delete Contact
4.) View All Contacts
5.) Search Address Book
6.) Exit

Choose an option:
```



```
Enter First Name: Rukma
Enter Last Name: Rao
Enter Address: A303
Enter Contact Number: 9106692610
sh: 1: pause: not found
sh: 1: cls: not found
----------------------Address Book-----------------------------
```

CONCLUSION AND RESULTS

We have successfully created an Address Book (Simple Address Book, Address Booklist) using the C++ programming language. Creating an address book program in C++ was a challenging but ultimately rewarding experience for us. We developed the project by breaking down the program into modules such as data structure, input/output, operations, and file handling; we were able to design a flexible and scalable solution that met the user's needs. Additionally, by using good programming practices, such as error handling, memory management, and code optimization, we were able to ensure that the program is efficient, robust, and easy to maintain. We would like to thank my OODP (Object-Oriented Design and Programming) lecturer for sharing his knowledge and expertise with us and for assisting us in improving as programmers. We are appreciative of their mentorship since it has motivated us to strive for excellence in software design and development.

REFERENCES

https://www.lucidchart.com › pages › uml_diagram_tool

https://www.codewithrandom.com/2022/02/05/simple-address-book-project-using-c-c-address-book-source-code/

https://www.geeksforgeeks.org/c-classes-and-objects/

https://www.w3schools.com/cpp/cpp_functions.asp