**A**

**PROJECT REPORT**

**ON**

**“CONTACT MANAGEMENT SYSTEM”**

SUBMITTED BY:

**Miss. Sayali Bhaurao Gangurde(2124UCEF1085)**

SUBJECT:

**Programming and problem**

**Solving using C++**

Under the guidance of

**Miss. Ishwari Tirse**



**Department of Computer Science and Engineering**

**Sanjivani Rural Education Society’s**

**SANJIVANI UNIVERSITY**

**KOPARGAON-423603, DIST: AHMEDNAGAR**

**2024-2025**

**01**

**INDEX**

|  |  |  |
| --- | --- | --- |
| **SR.NO** | **CONTENT** | **PAGE NO.** |
| **1** | **INTRODUCTION** | **3** |
| **2** | **CODE** |  |
| **3** | **OUTPUT** |  |
| **4** | **CONCLUSION** |  |

**02**

**INTRODUCTION**

A **Contact Management System** is a software application designed to store, manage, and retrieve contact information such as names, phone numbers, email addresses, and other details. These systems are commonly used by individuals and organizations to maintain a database of their personal or professional contacts in an organized manner.

In the context of programming, building a contact management system helps to understand fundamental concepts like data structures (arrays or vectors), file handling (for more advanced systems), and basic operations like adding, deleting, searching, and displaying information. Such systems are crucial for customer relationship management (CRM), helping businesses maintain meaningful communication with clients and customers.

**Importance of a Contact Management System:**

* **Organized Data**: It ensures that contact details are well-organized and easy to access.
* **Efficiency**: Searching for and retrieving contact information becomes efficient, saving time.
* **Data Management**: It allows users to easily add, update, or remove contacts, ensuring that information remains current.

**Key Features of a Contact Management System:**

1. **Adding Contacts**: The system allows users to add new contact entries, including details like name, phone number, and email address.
2. **Viewing Contacts**: Users can view the list of all saved contacts, which helps in keeping track of personal or professional networks.
3. **Searching Contacts**: The system provides functionality to search for a specific contact using attributes like name. This helps in quick retrieval of information.
4. **Deleting Contacts**: Users can delete outdated or unnecessary contacts, keeping the system clutter-free and up-to-date.

By building such a system in C++, programmers gain hands-on experience with fundamental operations and can later extend the functionality to include more complex features like file handling, sorting, and filtering.

03

**CODE**

**#include<iostream>**

**using namespace std;**

**class Contact {**

**public:**

**Contact (const string& name, const string& phone)**

**name(name), phone(phone) {}**

**void display() const {**

**cout << "Name: " << name << ", Phone: " << phone << endl;**

**}**

**string getName() const {**

**return name;**

**}**

**private:**

**string name;**

**string phone;**

**};**

**class ContactManager {**

**public:**

**void addContact(const string& name, const string& phone) {**

**contacts.emplace\_back(name, phone);**

**cout << "Contact added: " << name << endl;**

**}**

**void displayContacts() const {**

**if (contacts.empty()) {**

**cout << "No contacts available." << endl;**

**return;**

**}**

**for (const auto& contact : contacts) { 04**

**contact.display();**

**}**

**}**

**void deleteContact(const string& name) {**

**auto it = remove\_if(contacts.begin(), contacts.end(),**

**[&name](const Contact& contact) {**

**return contact.getName() == name;**

**});**

**if (it != contacts.end()) {**

**contacts.erase(it, contacts.end());**

**cout << "Contact '" << name << "' deleted." << endl;**

**} else {**

**cout << "Contact '" << name << "' not found." << endl;**

**}**

**}**

**private:**

**vector<Contact> contacts;**

**};**

**int main() {**

**ContactManager manager;**

**int choice;**

**string name, phone;**

**do {**

**cout << "\nContact Management System\n";**

**cout << "1. Add Contact\n";**

**cout << "2. Display Contacts\n"; 05**

**cout << "3. Delete Contact\n";**

**cout << "4. Exit\n";**

**cout << "Enter your choice: ";**

**cin >> choice;**

**switch (choice) {**

**case 1:**

**cout << "Enter name: ";**

**cin >> name;**

**cout << "Enter phone: ";**

**cin >> phone;**

**manager.addContact(name, phone);**

**break;**

**case 2:**

**manager.displayContacts();**

**break;**

**case 3:**

**cout << "Enter name of the contact to delete: ";**

**cin >> name;**

**manager.deleteContact(name);**

**break;**

**case 4:**

**cout << "Exiting..." << endl;**

**break;**

**default:**

**cout << "Invalid choice. Please try again." << endl;**

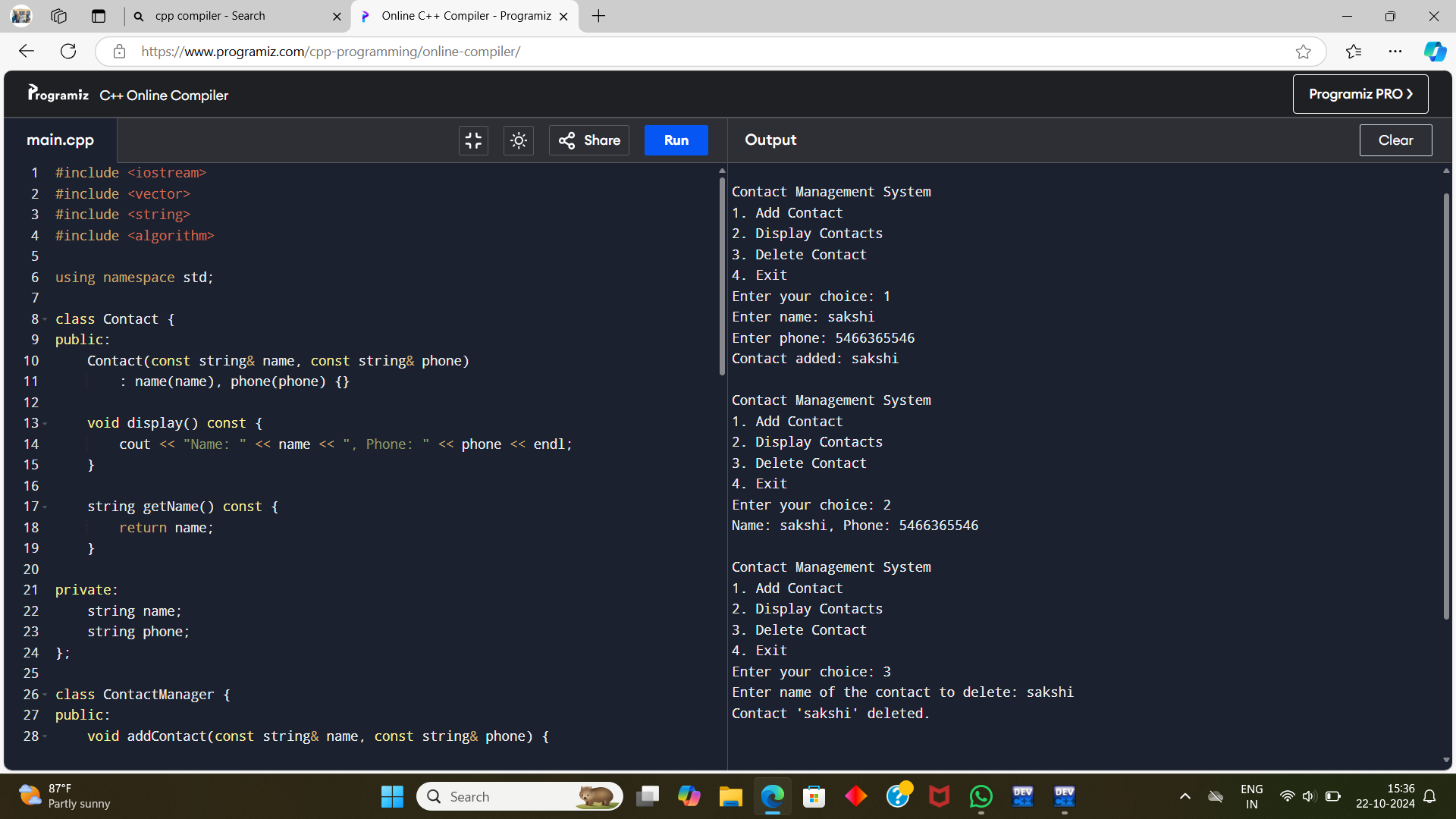
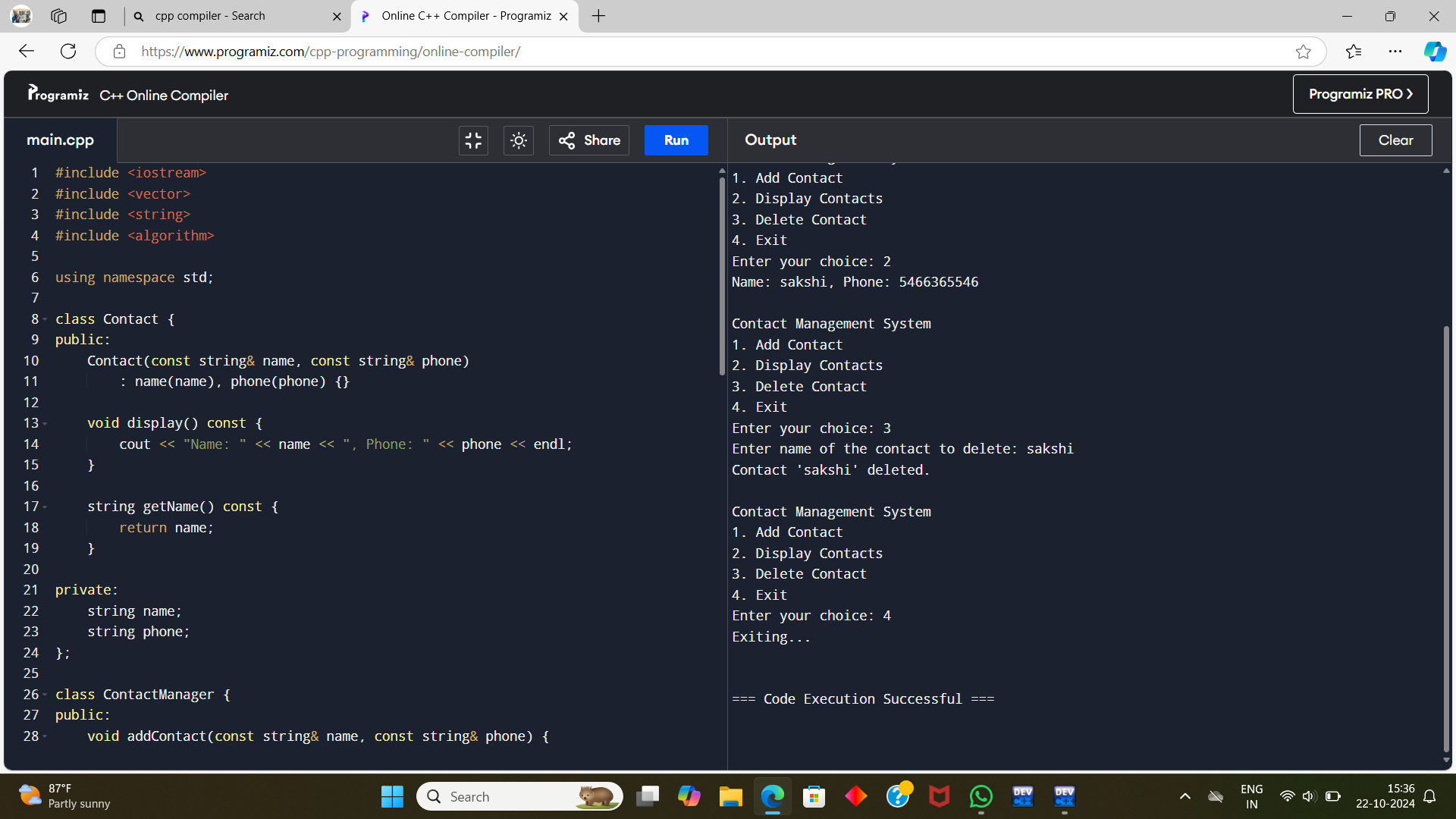
**}**

**} while (choice != 4);**

**return 0;**

**}**

**OUTPUT**

****

**CONCLUSION**

In conclusion, a **Contact Management System in C++** provides an efficient way to store, manage, and retrieve contact information using object-oriented programming principles. This system typically includes functionalities such as:

1. **Adding, Deleting, and Modifying Contacts:** Users can add new contacts with attributes like name, phone number, and email. They can also edit or delete existing records, ensuring the data remains up-to-date.
2. **Searching Contacts:** The system allows for efficient searching, often using algorithms like linear or binary search depending on how the data is stored.
3. **File Handling:** Contacts can be saved to and loaded from files, typically using text or binary file streams, making the data persistent across sessions.
4. **Efficient Data Structures:** Using data structures like arrays, linked lists, or even more advanced structures such as hash maps or trees allows for efficient contact retrieval and sorting.
5. **User-Friendly Interface:** The system can incorporate a menu-driven interface, offering ease of use to users who may not be familiar with complex programming concepts.