#include <iostream>

#include <fstream>

#include <string>

#include <map>

using namespace std;

struct Employee {

int id;

char name[50];

char designation[50];

float salary;

};

// Add employee and update index

void addEmployee() {

Employee emp;

cout << "Enter ID: ";

cin >> emp.id;

cout << "Enter Name: ";

cin.ignore();

cin.getline(emp.name, 50);

cout << "Enter Designation: ";

cin.getline(emp.designation, 50);

cout << "Enter Salary: ";

cin >> emp.salary;

// Write employee data to binary file

ofstream data("employee.dat", ios::binary | ios::app);

streampos pos = data.tellp(); // Get current write position

data.write((char\*)&emp, sizeof(emp));

data.close();

// Write index to the text file (store position as long)

ofstream index("index.txt", ios::app);

index << emp.id << " " << (long long)pos << endl; // Cast streampos to long long

index.close();

cout << "Employee added and indexed.\n";

}

// Display employee using index

void displayEmployee() {

int searchID;

cout << "Enter ID to search: ";

cin >> searchID;

ifstream index("index.txt");

if (!index) {

cout << "Index file not found.\n";

return;

}

int id;

long long pos;

bool found = false;

while (index >> id >> pos) {

if (id == searchID) {

found = true;

break;

}

}

index.close();

if (found) {

// Read employee data from binary file using the position found in the index

ifstream data("employee.dat", ios::binary);

data.seekg(pos); // Seek to the position in the binary file

Employee emp;

data.read((char\*)&emp, sizeof(emp));

data.close();

// Display employee information

cout << "\n--- Employee Found ---\n";

cout << "ID: " << emp.id << endl;

cout << "Name: " << emp.name << endl;

cout << "Designation: " << emp.designation << endl;

cout << "Salary: " << emp.salary << endl;

} else {

cout << "Employee not found in index.\n";

}

};

void deleteEmployee() {

int deleteId;

cout << "Enter Employee ID to delete: ";

cin >> deleteId;

ifstream index("index.txt");

if (!index) {

cout << "Index file not found.\n";

return;

}

ifstream data("employee.dat", ios::binary);

if (!data) {

cout << "Employee data file not found.\n";

return;

}

ofstream tempIndex("temp\_index.txt");

ofstream tempData("temp\_data.dat", ios::binary);

map<int, streampos> idMap;

int id;

long long pos;

// Read index data

while (index >> id >> pos) {

idMap[id] = pos;

}

// Check if ID was found in the map

if (idMap.find(deleteId) == idMap.end()) {

cout << "Employee with ID " << deleteId << " not found in the index.\n";

index.close();

data.close();

tempIndex.close();

tempData.close();

return;

}

// Now delete the employee by writing non-deleted records to the temporary files

for(auto &[id, pos] : idMap) {

data.seekg(pos);

Employee emp;

data.read((char \*)&emp, sizeof(emp));

if (id != deleteId) {

tempIndex << emp.id << " " << tempData.tellp() << endl;

tempData.write((char \*)&emp, sizeof(emp));

}

}

// Close file streams

index.close();

data.close();

tempIndex.close();

tempData.close();

// Remove old files and rename the temporary ones

remove("index.txt");

remove("employee.dat");

rename("temp\_index.txt", "index.txt");

rename("temp\_data.dat", "employee.dat");

cout << "Employee deletion completed.\n";

}

int main() {

int choice;

do {

cout << "\n=== Employee Menu ===\n";

cout << "1. Add Employee\n";

cout << "2. Display Employee\n";

cout << "3. Delete Employee\n";

cout<<"4.Exit \n";

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1: addEmployee(); break;

case 2: displayEmployee(); break;

case 3: deleteEmployee(); break;

default: cout << "Invalid choice.\n";

}

} while (choice != 4);

return 0;

}