Company maintains employee information as employee ID, name, designation and salary. Allow user to add, delete information of employee. Display information of particular employee. If employee does not exist an appropriate message is displayed. If it is, then the system displays the employee details. Use index sequential file to maintain the data.

#include <iostream>

#include <fstream>

#include <vector>

#include <algorithm>

using namespace std;

class Employee {

public:

int id;

char name[50];

char designation[50];

float salary;

void input() {

cout << "Enter Employee ID: ";

cin >> id;

cin.ignore();

cout << "Enter Name: ";

cin.getline(name, 50);

cout << "Enter Designation: ";

cin.getline(designation, 50);

cout << "Enter Salary: ";

cin >> salary;

}

void display() const {

cout << "\nEmployee ID: " << id

<< "\nName: " << name

<< "\nDesignation: " << designation

<< "\nSalary: " << salary << "\n";

}

int getID() const {

return id;

}

};

// Index structure to map employee ID to position in file

struct IndexEntry {

int empID;

long position;

};

vector<IndexEntry> indexTable;

// Load index file

void loadIndex() {

indexTable.clear();

ifstream indexFile("index.dat", ios::binary);

if (!indexFile) return;

IndexEntry entry;

while (indexFile.read((char\*)&entry, sizeof(entry))) {

indexTable.push\_back(entry);

}

indexFile.close();

}

// Update index file from indexTable

void updateIndex() {

ofstream indexFile("index.dat", ios::binary | ios::trunc);

for (auto& entry : indexTable) {

indexFile.write((char\*)&entry, sizeof(entry));

}

indexFile.close();

}

// Binary search index

int searchIndex(int id) {

for (int i = 0; i < indexTable.size(); ++i) {

if (indexTable[i].empID == id)

return i;

}

return -1;

}

// Add employee record

void addEmployee() {

Employee emp;

emp.input();

if (searchIndex(emp.getID()) != -1) {

cout << "Employee ID already exists.\n";

return;

}

fstream file("employee.dat", ios::binary | ios::app);

long pos = file.tellp(); // Position where the record will be written

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

file.close();

indexTable.push\_back({emp.getID(), pos});

sort(indexTable.begin(), indexTable.end(), [](IndexEntry a, IndexEntry b) {

return a.empID < b.empID;

});

updateIndex();

cout << "Employee record added successfully.\n";

}

// Display employee by ID

void displayEmployee(int id) {

int idx = searchIndex(id);

if (idx == -1) {

cout << "Employee with ID " << id << " not found.\n";

return;

}

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

file.seekg(indexTable[idx].position);

Employee emp;

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

file.close();

emp.display();

}

// Delete employee record

void deleteEmployee(int id) {

int idx = searchIndex(id);

if (idx == -1) {

cout << "Employee with ID " << id << " not found.\n";

return;

}

fstream inFile("employee.dat", ios::binary);

ofstream outFile("temp.dat", ios::binary);

Employee emp;

long newPos = 0;

vector<IndexEntry> newIndex;

while (inFile.read((char\*)&emp, sizeof(emp))) {

if (emp.getID() == id)

continue;

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

newIndex.push\_back({emp.getID(), newPos});

newPos += sizeof(emp);

}

inFile.close();

outFile.close();

remove("employee.dat");

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

indexTable = newIndex;

updateIndex();

cout << "Employee with ID " << id << " deleted successfully.\n";

}

int main() {

loadIndex();

int choice, id;

do {

cout << "\n--- Company Employee Management ---\n";

cout << "1. Add Employee\n2. Display Employee\n3. Delete Employee\n4. Exit\nEnter choice: ";

cin >> choice;

switch (choice) {

case 1:

addEmployee();

break;

case 2:

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

cin >> id;

displayEmployee(id);

break;

case 3:

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

cin >> id;

deleteEmployee(id);

break;

case 4:

cout << "Exiting...\n";

break;

default:

cout << "Invalid choice!\n";

}

} while (choice != 4);

return 0;

}

**OUTPUT**

--- Company Employee Management ---

1. Add Employee

2. Display Employee

3. Delete Employee

4. Exit

Enter choice: 1

Enter Employee ID: 101

Enter Name: John Doe

Enter Designation: Software Engineer

Enter Salary: 75000

Employee record added successfully.

--- Company Employee Management ---

1. Add Employee

2. Display Employee

3. Delete Employee

4. Exit

Enter choice: 1

Enter Employee ID: 102

Enter Name: Alice Smith

Enter Designation: HR Manager

Enter Salary: 65000

Employee record added successfully.

--- Company Employee Management ---

1. Add Employee

2. Display Employee

3. Delete Employee

4. Exit

Enter choice: 2

Enter Employee ID to search: 101

Employee ID: 101

Name: John Doe

Designation: Software Engineer

Salary: 75000

--- Company Employee Management ---

1. Add Employee

2. Display Employee

3. Delete Employee

4. Exit

Enter choice: 2

Enter Employee ID to search: 999

Employee with ID 999 not found.

--- Company Employee Management ---

1. Add Employee

2. Display Employee

3. Delete Employee

4. Exit

Enter choice: 3

Enter Employee ID to delete: 102

Employee with ID 102 deleted successfully.

--- Company Employee Management ---

1. Add Employee

2. Display Employee

3. Delete Employee

4. Exit

Enter choice: 2

Enter Employee ID to search: 102

Employee with ID 102 not found.

--- Company Employee Management ---

1. Add Employee

2. Display Employee

3. Delete Employee

4. Exit

Enter choice: 4

Exiting...