#include <iostream>

#include <fstream>

#include <sstream>

#include <vector>

class Node {

public:

int id;

std::string name;

Node\* next;

Node(int \_id, const std::string& \_name) : id(\_id), name(\_name), next(nullptr) {}

};

class Table {

private:

Node\* head;

public:

Table() : head(nullptr) {}

void append(int id, const std::string& name) {

Node\* newNode = new Node(id, name);

if (!head) {

head = newNode;

} else {

Node\* current = head;

while (current->next) {

current = current->next;

}

current->next = newNode;

}

}

void display() const {

Node\* current = head;

while (current) {

std::cout << "ID: " << current->id << ", Name: " << current->name << std::endl;

current = current->next;

}

std::cout << std::endl;

}

std::vector<Node\*> searchRecord(int id) const {

std::vector<Node\*> matchingRecords;

Node\* current = head;

while (current) {

if (current->id == id) {

matchingRecords.push\_back(current);

}

current = current->next;

}

return matchingRecords;

}

void deleteRecord(int id) {

if (!head) {

return;

}

if (head->id == id) {

Node\* temp = head;

head = head->next;

delete temp;

return;

}

Node\* current = head;

while (current->next && current->next->id != id) {

current = current->next;

}

if (current->next) {

Node\* temp = current->next;

current->next = current->next->next;

delete temp;

}

}

void saveToCSV(const std::string& filename) const {

std::ofstream file(filename);

if (!file.is\_open()) {

std::cerr << "Error opening file for writing: " << filename << std::endl;

return;

}

Node\* current = head;

while (current) {

file << current->id << "," << current->name << std::endl;

current = current->next;

}

file.close();

std::cout << "Data saved to CSV file: " << filename << std::endl;

}

void clear() {

Node\* current = head;

while (current) {

Node\* next = current->next;

delete current;

current = next;

}

head = nullptr;

}

};

void readDataFromFile(const std::string& filename, Table& table) {

std::ifstream file(filename);

if (!file.is\_open()) {

std::cerr << "Error opening file: " << filename << std::endl;

return;

}

std::string line;

while (std::getline(file, line)) {

std::istringstream iss(line);

int id;

std::string name;

if (iss >> id >> name) {

table.append(id, name);

} else {

std::cerr << "Error parsing line: " << line << std::endl;

}

}

file.close();

}

int main() {

std::string filePath = "C:/Users/Chirangi gupta/Desktop/PBL/database contents(single table) spreadsheet - Student.csv";

Table studentTable;

readDataFromFile(filePath, studentTable);

int choice;

std::string newName;

do {

std::cout << "\nMenu:\n";

std::cout << "1. Search records by ID\n";

std::cout << "2. Append a new record\n";

std::cout << "3. Delete a record by ID\n";

std::cout << "4. Display all records\n";

std::cout << "5. Save to CSV file\n";

std::cout << "6. Exit\n";

std::cout << "Enter your choice (1-6): ";

std::cin >> choice;

switch (choice) {

case 1:

// Search records by ID

int searchId;

std::cout << "Enter the ID to search: ";

std::cin >> searchId;

{

std::vector<Node\*> foundRecords = studentTable.searchRecord(searchId);

if (!foundRecords.empty()) {

std::cout << "Records with ID " << searchId << " found:" << std::endl;

for (const auto& record : foundRecords) {

std::cout << "ID: " << record->id << ", Name: " << record->name << std::endl;

}

} else {

std::cout << "No records with ID " << searchId << " found." << std::endl;

}

}

break;

case 2:

// Append a new record

int newId;

std::cout << "Enter the new record ID: ";

std::cin >> newId;

std::cout << "Enter the new record Name: ";

std::cin.ignore();

std::getline(std::cin, newName);

studentTable.append(newId, newName);

std::cout << "Student Table after appending new record:" << std::endl;

studentTable.display();

studentTable.saveToCSV(filePath);

break;

case 3:

// Delete a record by ID

int deleteId;

std::cout << "Enter the ID to delete: ";

std::cin >> deleteId;

studentTable.deleteRecord(deleteId);

std::cout << "Student Table after deleting record with ID " << deleteId << ":" << std::endl;

studentTable.display();

studentTable.saveToCSV(filePath);

break;

case 4:

// Display all records

std::cout << "All records:" << std::endl;

studentTable.display();

break;

case 5:

// Save to CSV file

studentTable.saveToCSV(filePath);

break;

case 6:

// Exit

std::cout << "Exiting program.\n";

break;

default:

std::cout << "Invalid choice. Please enter a number between 1 and 6.\n";

break;

}

} while (choice != 6);

return 0;

}