MEDICAL INFORMATION SYSTEM

#include <iostream>

#include <string>

using namespace std;

const int MAX\_PEOPLE = 100; // Maximum number of people in the medical record

// Base class representing a person

class Person {

protected:

string name;

int age;

public:

// Constructor for initializing name and age

Person(const string& n, int a) : name(n), age(a) {}

// Virtual function to display basic information about a person

virtual void displayInfo() const {

cout << "Name: " << name << ", Age: " << age;

}

// Other methods as needed

};

// Derived class representing a patient

class Patient : public Person {

private:

string condition;

public:

// Constructor for initializing patient's name, age, and condition

Patient(const string& n, int a, const string& c) : Person(n, a), condition(c) {}

// Overridden displayInfo to include patient-specific information

void displayInfo() const override {

Person::displayInfo();

cout << ", Condition: " << condition;

}

// Other methods specific to patients

};

// Derived class representing a doctor

class Doctor : public Person {

private:

string specialty;

public:

// Constructor for initializing doctor's name, age, and specialty

Doctor(const string& n, int a, const std::string& s) : Person(n, a), specialty(s) {}

// Overridden displayInfo to include doctor-specific information

void displayInfo() const override {

Person::displayInfo();

cout << ", Specialty: " << specialty;

}

// Other methods specific to doctors

};

class MedicalRecord {

private:

Person\* people[MAX\_PEOPLE]; // Array of pointers to the base class

int peopleCount;

public:

// Constructor for initializing the medical record

MedicalRecord() : peopleCount(0) {}

// Destructor to clean up dynamically allocated objects

~MedicalRecord() {

for (int i = 0; i < peopleCount; ++i) {

delete people[i];

}

}

// Method to add a person to the medical record

void addPerson(Person\* person) {

if (peopleCount < MAX\_PEOPLE) {

people[peopleCount++] = person;

cout << "Person added successfully.\n";

} else {

cout << "Medical record is full. Cannot add more people.\n";

delete person; // Cleanup if adding fails

}

}

// Method to display information about all people in the medical record

void displayPeople() const {

if (peopleCount == 0) {

cout << "No people in the medical record.\n";

} else {

cout << "Medical Record:\n";

for (int i = 0; i < peopleCount; ++i) {

people[i]->displayInfo();

std::cout << '\n';

}

}

}

// Other methods as needed

};

// Function to display menu and get user choice

int getMenuChoice() {

std::cout<<" Medical Information Sytem \n";

cout << "\nMenu:\n";

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

cout << "2. Add Doctor\n";

cout << "3. Display People\n";

cout << "4. Exit\n";

cout << "Enter your choice: ";

int choice;

cin >> choice;

return choice;

}

// Simple user interface

int main() {

MedicalRecord medicalRecord;

while (true) {

int choice = getMenuChoice();

switch (choice) {

case 1: {

std::string name, condition;

int age;

// Get patient information from the user

cout << "Enter patient name: ";

cin.ignore(); // Ignore newline left in the buffer

getline(std::cin, name);

cout << "Enter patient age: ";

cin >> age;

cout << "Enter patient condition: ";

cin.ignore(); // Ignore newline left in the buffer

getline(std::cin, condition);

// Create a new patient and add to the medical record

medicalRecord.addPerson(new Patient(name, age, condition));

break;

}

case 2: {

string name, specialty;

int age;

// Get doctor information from the user

cout << "Enter doctor name: ";

cin.ignore(); // Ignore newline left in the buffer

getline(std::cin, name);

cout << "Enter doctor age: ";

cin >> age;

cout << "Enter doctor specialty: ";

cin.ignore(); // Ignore newline left in the buffer

getline(std::cin, specialty);

// Create a new doctor and add to the medical record

medicalRecord.addPerson(new Doctor(name, age, specialty));

break;

}

case 3:

// Display people information

medicalRecord.displayPeople();

break;

case 4:

// Exit the program

cout << "Exiting the medical information system.\n";

return 0;

default:

cout << "Invalid choice. Please enter a valid option.\n";

}

}

}

Output:

