Виконав:Шевчук К.В

Група:КІ 2-23-4

Викладачу:Кисіль В.В

Лабараторна 4

Перший рівень

#include <iostream>

#include <vector>

#include <string>

using namespace std;

class Person {

protected:

string name;

int age;

string gender;

public:

Person(string name = "", int age = 0, string gender = "")

: name(name), age(age), gender(gender) {}

virtual string getInfo() {

return "Iм'я: " + name + ", Роки: " + to\_string(age) + ", Гендр: " + gender;

}

virtual ~Person() {

cout << "Видалення особи: " << name << endl;

}

};

class MedicalHistory {

private:

vector<string> illnesses;

vector<string> treatments;

vector<string> testResults;

public:

void addIllness(const string& illness) {

illnesses.push\_back(illness);

}

void addTreatment(const string& treatment) {

treatments.push\_back(treatment);

}

void addTestResult(const string& result) {

testResults.push\_back(result);

}

void printHistory() {

cout << "Illnesses: " << endl;

for (const auto& illness : illnesses)

cout << illness << endl;

cout << "Treatments: " << endl;

for (const auto& treatment : treatments)

cout << treatment << endl;

cout << "Test Results: " << endl;

for (const auto& result : testResults)

cout << result << endl;

}

};

class Ward {

private:

int beds;

vector<string> equipment;

vector<Person\*> patients;

public:

Ward(int beds = 0, vector<string> equipment = {})

: beds(beds), equipment(equipment) {}

void addPatient(Person\* patient) {

if (patients.size() < beds) {

patients.push\_back(patient);

cout << "Пацiєнта додано до палати" << endl;

}

else {

cout << "Палата заповнена!" << endl;

}

}

void removePatient(Person\* patient) {

patients.erase(remove(patients.begin(), patients.end(), patient), patients.end());

cout << "Пацiєнта виведено з палати" << endl;

}

void requestAdditionalEquipment(const string& newEquipment) {

equipment.push\_back(newEquipment);

cout << "Необхiдне додаткове обладнання: " << newEquipment << endl;

}

~Ward() {

cout << "Видалення вiддiленя лiкарнi" << endl;

}

};

class Doctor : public Person {

private:

string specialization;

vector<Person\*> patients;

public:

Doctor(string name = "", int age = 0, string gender = "", string specialization = "")

: Person(name, age, gender), specialization(specialization) {}

void assignTreatment(Person\* patient, const string& treatment, MedicalHistory\* history) {

cout << "Лікар " << name << "Призначене лікування:" << treatment << endl;

history->addTreatment(treatment);

}

void performDiagnosis(Person\* patient) {

cout << "Лікар" << name << " проводить діагностику пацієнта" << endl;

}

void requestTests(const string& testType, MedicalHistory\* history) {

cout << "Лікар" << name << "тест запитів:" << testType << endl;

history->addTestResult("Test: " + testType);

}

~Doctor() {

cout << "Видалення лікаря:" << name << endl;

}

};

class Nurse : public Person {

public:

Nurse(string name = "", int age = 0, string gender = "")

: Person(name, age, gender) {}

void executeTreatment(const string& treatment, Person\* patient) {

cout << "Медсестра " << name << " проводить лікування:" << treatment << endl;

}

~Nurse() {

cout << "Видалення медсестри:" << name << endl;

}

};

class Patient : public Person {

private:

Doctor\* doctor;

MedicalHistory medicalHistory;

Ward\* ward;

string diagnosis;

public:

Patient(string name = "", int age = 0, string gender = "", Doctor\* doctor = nullptr, Ward\* ward = nullptr, string diagnosis = "")

: Person(name, age, gender), doctor(doctor), ward(ward), diagnosis(diagnosis) {}

void assignDoctor(Doctor\* doctor) {

this->doctor = doctor;

cout << "Лікар, призначений пацієнту" << name << endl;

}

void addToMedicalHistory(const string& record) {

medicalHistory.addIllness(record);

}

void admitToWard(Ward\* ward) {

this->ward = ward;

cout << "Хворий госпіталізований в палату" << endl;

}

void discharge() {

cout << "Пацієнт " << name << "був виписаний." << endl;

}

MedicalHistory\* getMedicalHistory() {

return &medicalHistory;

}

~Patient() {

cout << "Видалення пацієнта: " << name << endl;

}

void printMedicalHistory() {

medicalHistory.printHistory();

}

};

class Test {

private:

string testType;

string results;

public:

Test(string testType = "", string results = "")

: testType(testType), results(results) {}

void performTest() {

cout << "Виконання тесту: " << testType << endl;

}

void storeResults(const string& result, MedicalHistory\* history) {

results = result;

cout << "Результати, збережені для тесту: " << testType << endl;

history->addTestResult(results);

}

~Test() {

cout << "Удалення тесту: " << testType << endl;

}

};

int main() {

Doctor\* doc = new Doctor("Dr. Smith", 45, "Мужчина", "Кардiолог");

Ward\* ward = new Ward(2, { "апарат ЕКГ", "4 койка" });

Patient\* pat = new Patient("John Doe", 30, "Мужчина", doc);

Nurse\* nurse = new Nurse("Jane Roe", 28, "Жiнка");

Test\* test = new Test("Аналiз кровi");

setlocale(LC\_CTYPE, "ukr");

pat->assignDoctor(doc);

doc->assignTreatment(pat, "Лiки вiд артерiального тиску", pat->getMedicalHistory());

ward->addPatient(pat);

doc->requestTests("Тест кровi", pat->getMedicalHistory());

nurse->executeTreatment("Лiки вiд артерiального тиску", pat);

delete doc;

delete pat;

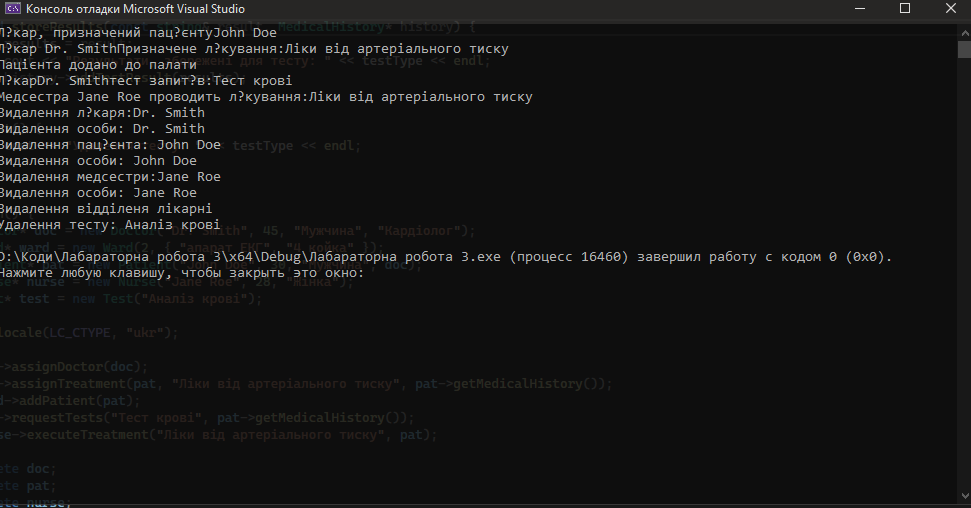
delete nurse;

delete ward;

delete test;

return 0;

}



+--------------+

| Person |

+--------------+

/ | \

+---------+ +---------+ +---------+

| Doctor | | Nurse | | Patient |

+---------+ +---------+ +---------+

|

+------------+

| MedicalRecord|

+------------+

+----------+ +------------+ +------------+

| Ward | | MedicalTest | | Hospital |

+----------+ +------------+ +------------+