МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний технічний університет

«Харківський політехнічний інститут»

Факультет Комп’ютерних наук та програмної інженерії

Кафедра Інформатики та інтелектуальної власності

ЗВІТ

До лабораторной роботи №4 з дисципліни

«Об’єкто-орієнтоване програмування»

Студент Карауланов А.О.

Викладач Івашко А.В.

Харків 2022

**Лабораторна робота №4**

**Тема: «Найпростіші класи та об’єкти»**

**Мета роботи:** навчитись створювати прості класи та використовувати їх на прикладі простих об’єктів.

**Завдання:** Розробити клас згідно до свого варіанта. Включити до класу методи **set** (…), **get** (…), **show** (…) та ін., використовуючи принцип інкапсуляції. Написати програму, яка створює список, масив чи колекцію, об’єктів на основі відповідного класу. Під час створення використати випадковий підхід для генерації числової та текстової інформації об’єктів списку. Передбачити можливість додання, редагування та видалення об’єктів зі списку за певними критеріями. В програмі реалізувати функціонал за своїм варіантом.

8. **Patient**: Прізвище, Ім'я, По-батькові, Адреса, Номер медичної карти, Діагноз. Створити масив об'єктів. Вивести:

· список пацієнтів, які мають цей діагноз;

· список пацієнтів, номер медичної картки яких перебуває у заданому інтервалі.

**1 Виконання роботи**

Програма побудована за допомогою масивів об’єктів на основі класі Circle, Square, Rectangle. У інформаційному полі, згідно завданню, існують п’ять змінних типу double ax, ay, cx, cy і r з інкапсуляцією private за умовчанням. Також, для доступу к змінним класу були об’явлені методи setax, getax, setay, getay, setcx, getcx setcy, getcy setr, getr.

#pragma once

#include <iostream>

#include <ctime>

#include <iomanip>

#include <math.h>

**using** **namespace** std;

**class** Square{

**private**:

**double** ax, ay, cx, cy;

**public**:

**void** setax(**double** valueax);

**double** getax();

**void** setay(**double** valueay);

**double** getay();

**void** setcx(**double** valuecx);

**double** getcx();

**void** setcy(**double** valuecy);

**double** getcy();

**void** Vvod2();

**void** Vivod2();

**void** Perem2();

**void** Izm\_raz2();

**void** Vrash2();

**friend** Square& **operator** + (**const** Square& ob1, **int** value);

**friend** Square& **operator** \* (**const** Square& ob1, **int** value);

};

**class** Circle{

**private**:

**double** cx, cy, r;

**public**:

Circle();

Circle(**double** cx, **double** cy, **double** r);

**void** setcx(**double** valuecx);

**double** getcx();

**void** setcy(**double** valuecy);

**double** getcy();

**void** setr(**double** valuer);

**double** getr();

**void** Vvod1();

**void** Vivod1();

**void** Perem1();

**void** Izm\_raz1();

**friend** Circle& **operator** + (**const** Circle& ob1, **int** value);

**friend** Circle& **operator** \* (**const** Circle& ob1, **int** value);

};

**class** Rectangle{

**private**:

**double** ax, ay, cx, cy;

**public**:

**void** setax(**double** valueax);

**double** getax();

**void** setay(**double** valueay);

**double** getay();

**void** setcx(**double** valuecx);

**double** getcx();

**void** setcy(**double** valuecy);

**double** getcy();

**void** Vvod3();

**void** Vivod3();

**void** Perem3();

**void** Izm\_raz3();

**void** Vrash3();

**friend** Rectangle& **operator** + (**const** Rectangle& ob1, **int** value);

**friend** Rectangle& **operator** \* (**const** Rectangle& ob1, **int** value);

};

**class** Main{

**public**:

**void** main() **const**;

};Повний код програми наведено у додатку Б.

Методи setax, getax, setay, getay, setcx, getcx setcy, getcy setr, getr.

дозволяють проініціалізувати змінні класу. Приклад коду наведено нижче та повна реалізація методу наведено нижче.

**void** setcx(**double** valuecx);

**double** getcx();

**void** setcy(**double** valuecy);

**double** getcy();

**void** setr(**double** valuer);Повний код програми наведено у додатку Б.

Що до методів getter, то вони отримують значення цих змінних шляхом return. Повна реалізація методів наведено нижче.

**void** patient::setsurname(string valuesurname){

surname = valuesurname;

}

string patient::getsurname(){

**return** surname;

}

**void** patient::setname(string valuename){

name = valuename;

}

string patient::getname(){

**return** name;

}

**void** patient::setsecname(string valuesecname){

secname = valuesecname;

}

string patient::getsecname(){

**return** secname;

}

**void** patient::setadress(string valueadress){

adress = valueadress;

}

string patient::getadress(){

**return** adress;

}

**void** patient::setnum(**int** valuenum){

num = valuenum;

}

**int** patient::getnum(){

**return** num;

}

**void** patient::setdiagnosis(string valuediagnosis){

diagnosis = valuediagnosis;

}

string patient::getdiagnosis(){

**return** diagnosis;

}

Повний код програми наведено у додатку Б.

Метод vivod1 дозволяє вивести у заданому порядку частину таблиці, що продемонстровано на рисунку 1.1.

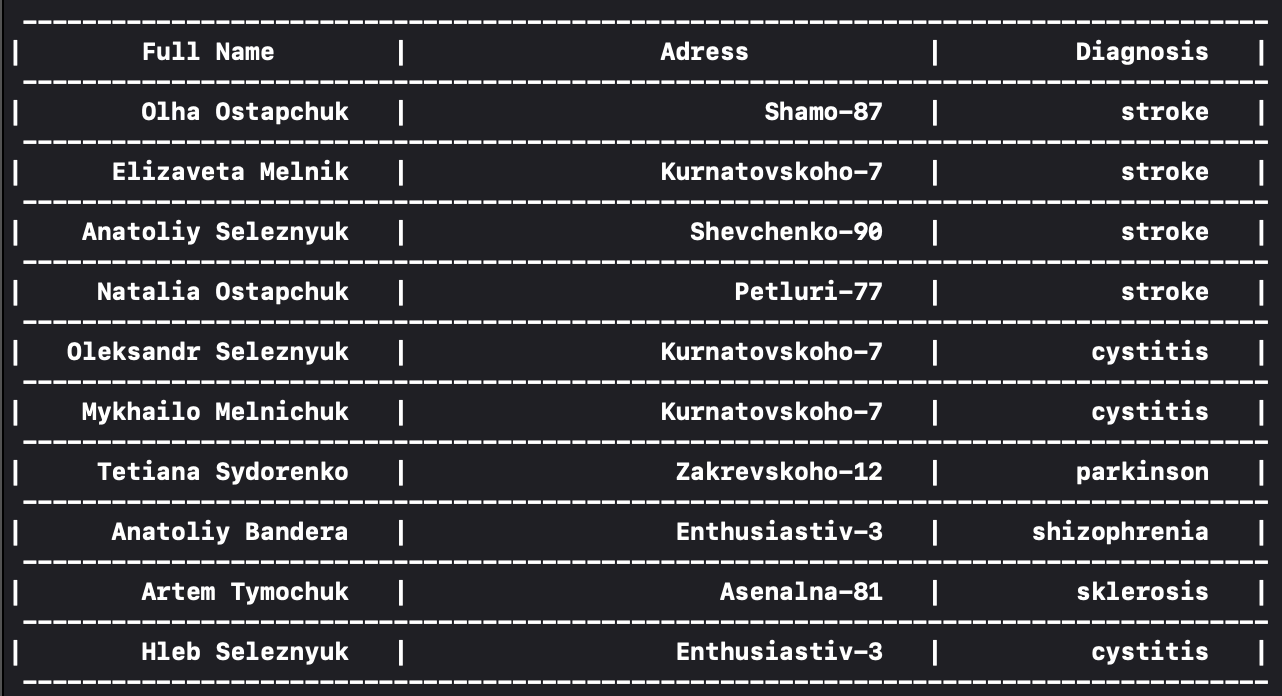


Рисунок 1.1 - Друк списку

Алгоритм методу search\_by\_num полягає у тому, що існує масив об’єктів класу списка, у який записуються данні студентів, які мають незадовільні оцінки, нижче шести десяти балів, та якщо такі є, то ці данні виводяться на екран з підрахунком студентів. У цьому алгоритмі були використані методи класу Abiturient, наведені вище. Приклад роботи методу наведено на рисунках 1.2 та 1.3.

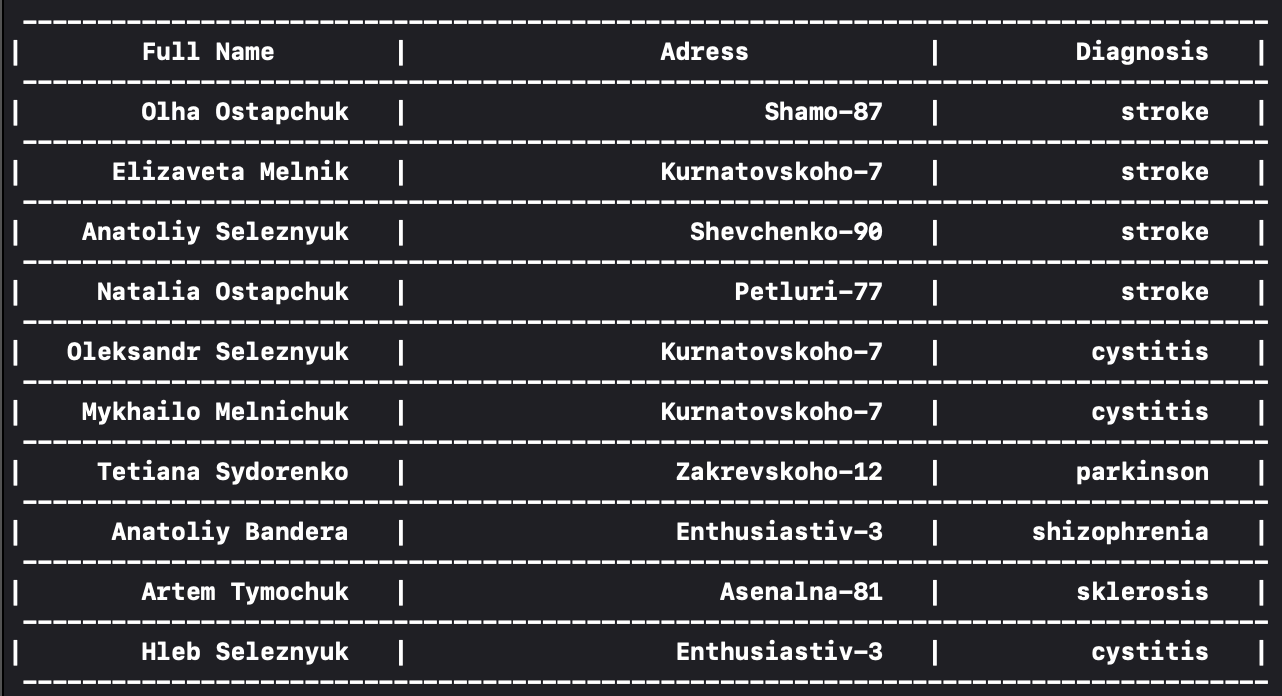


Рисунок 1.2 - Повний список із десяти паціентів

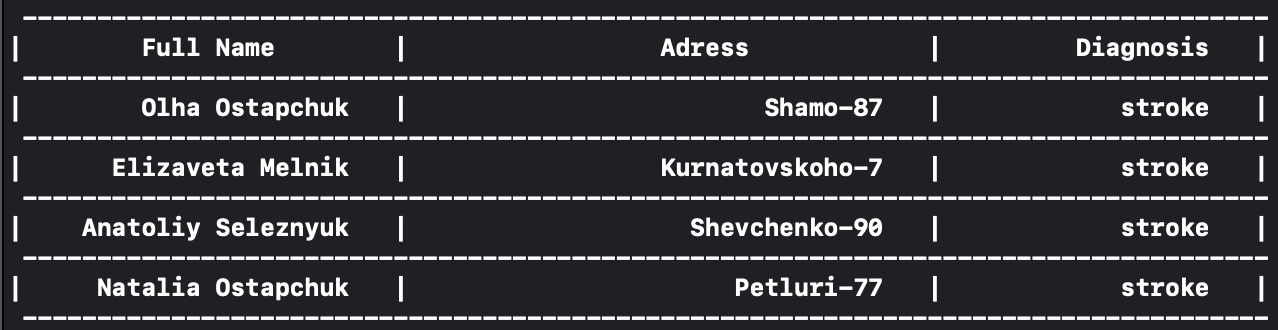


Рисунок 1.3 - Список паціентів, у яких є такий діагноз

Приклад реалізації цього методу наведено нижче.

Приклад коду метода vivod1:

**void** Circle::Vvod1(){

cout<<endl;

cout<<"Введите параметры круга \n";

cout<<" cx=";

cin>>cx;

cout<<" cy=";

cin>>cy;

cout<<" r=";

cin>>r;

cout<<endl;

}

**void** Circle::Vvod3(){

cout<<endl;

cout<<"Введите параметры круга \n";

cout<<" cx=";

cin>>cx;

cout<<" cy=";

cin>>cy;

cout<<" r=";

cin>>r;

cout<<endl;

}

**void** Circle::Vvod3(){

cout<<endl;

cout<<"Введите параметры круга \n";

cout<<" cx=";

cin>>cx;

cout<<" cy=";

cin>>cy;

cout<<" r=";

cin>>r;

cout<<endl;

}

Повний текст програми наведено у додатку Б.

Алгоритм методу search\_by\_num полягає у тому, що існує масив об’єктів класу списка, у який записуються данні паціентів, які мають в медецинські картки в діапазоні заданий користувачем, та якщо такі є, то ці данні виводяться на екран з підрахунком паціентів. У цьому алгоритмі були використані методи класу Patient, наведені вище. Приклад роботи методу наведено на рисунках 1.4 та 1.5.

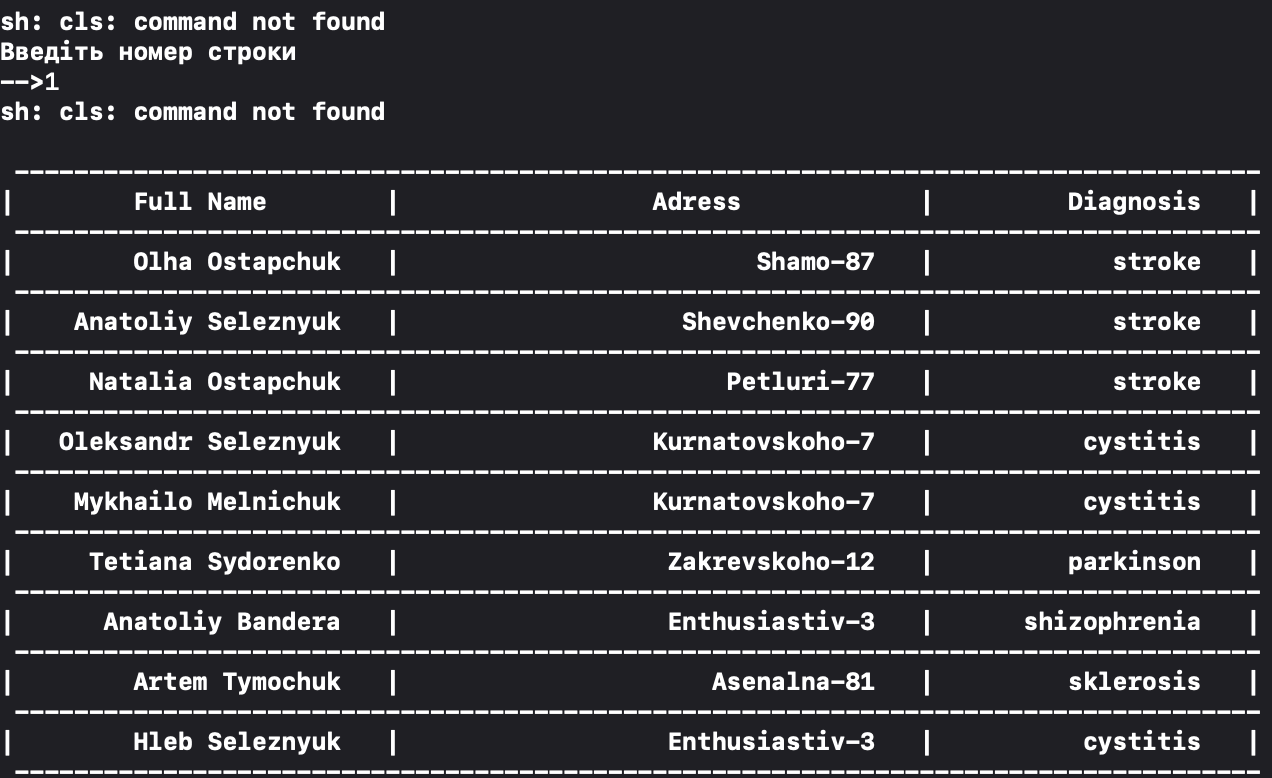


Рисунок 1.4 – Повний список із десяти паціентів

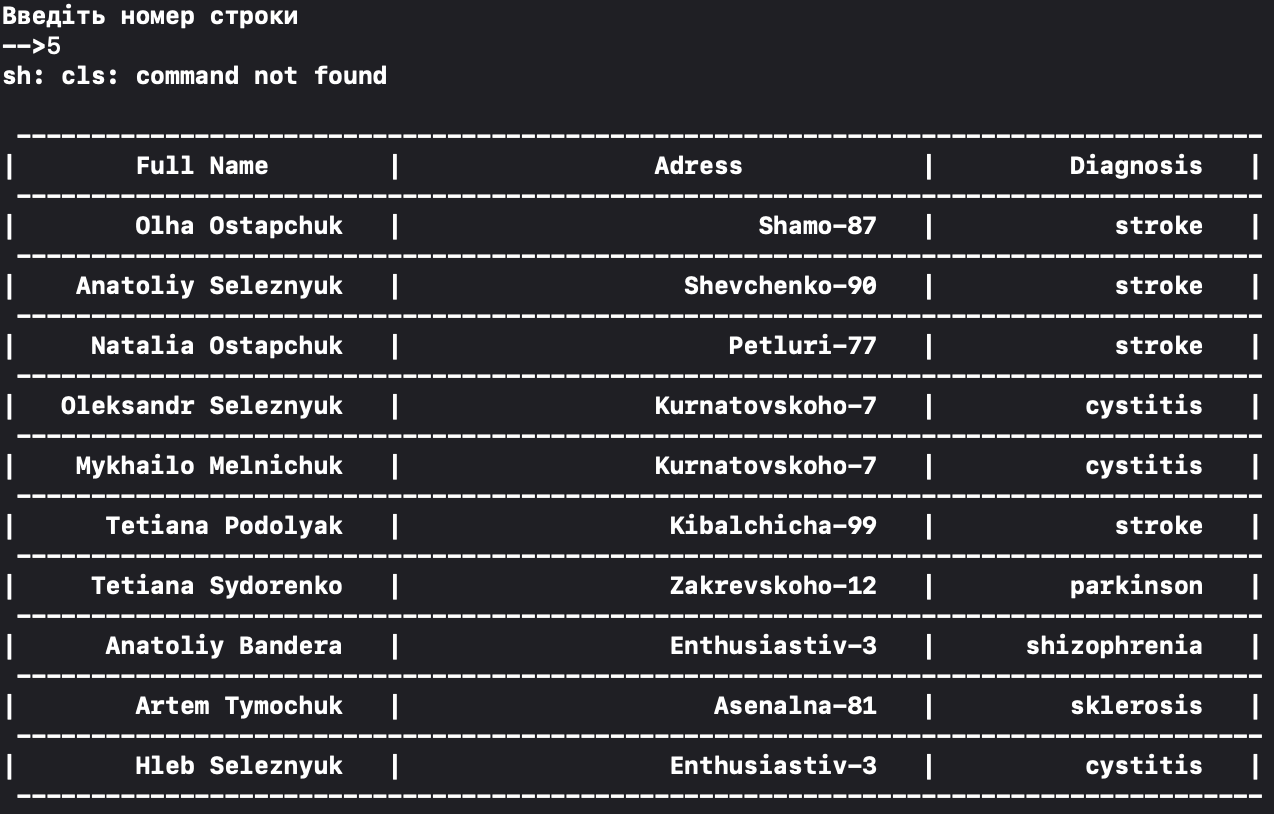


Рисунок 1.5 – Список паціенті медецинські карти, яких знаходяться в діапазоні від 1000300 до 1000800

Приклад реалізації цього методу наведено нижче.

Приклад коду метода search\_by\_num:

**void** patient::search\_by\_num(patient patient[]){

**int** num1;

**int** num2;

cout << "Введіть перший номер картки для інтервалу: " << endl;

cin >> num1;

cout << "Введіть другий номер картки для інтервал: " << endl;

cin >> num2;

cout << setw(4) << "-------------------------------------------------------------------------------------\n";

cout << setw(4) << "| Список пацієнтів |\n";

cout << setw(4) << "-------------------------------------------------------------------------------------\n";

**for** (**int** i = 0; i < N; i++) {

**if** (num1 < patient[i].num && num2 > patient[i].num) {

cout << "|"<<setw(10) << patient[i].surname << setw(4);

cout << "|" << setw(4) << patient[i].name << setw(2);

cout << "|" << setw(6) << patient[i].secname << setw(6);

cout << "|" << setw(3) << patient[i].adress <<setw(6);

cout << "|" << setw(3) << patient[i].num <<setw(6);

cout << "|" << setw(3) << patient[i].diagnosis <<setw(13);

cout<<"|\n";

cout<< "-------------------------------------------------------------------------------------\n";

}

}

}

Повний текст програми наведено у додатку Б.

**2 Довідка по роботі з програмою для користувача. Ознайомлення  
з інтерфейсом програми**

Головне меню зустрічає користувача чотирма пунктами. Перший пункт – це друк. Другий – пошук за діагнозом. Третій пункт – пошук за інтервалом. Та останній – вихід з програми. Меню наглядно продемонстровано на рисунку 2.1.

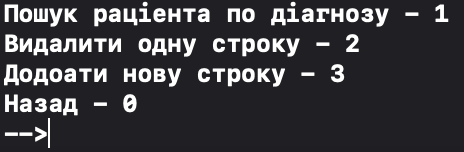


Рисунок 2.1 – Головне меню

Якщо обрати перший пункт ми зможемо побачити список пацієнтів. Що продемонстровано на рисунку 2.2.

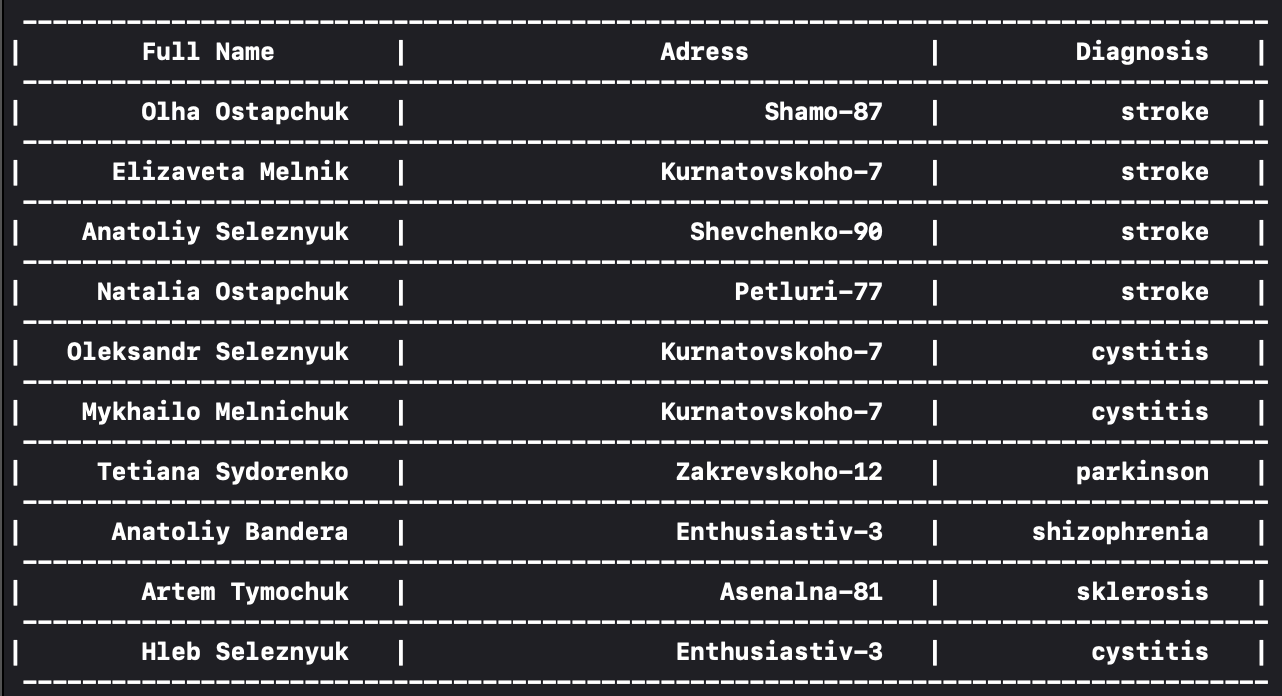


Рисунок 2.2 – Друк

Пункт другий пропонує здійснити пошук за діагнозом що наведено на рисунку 2.3.

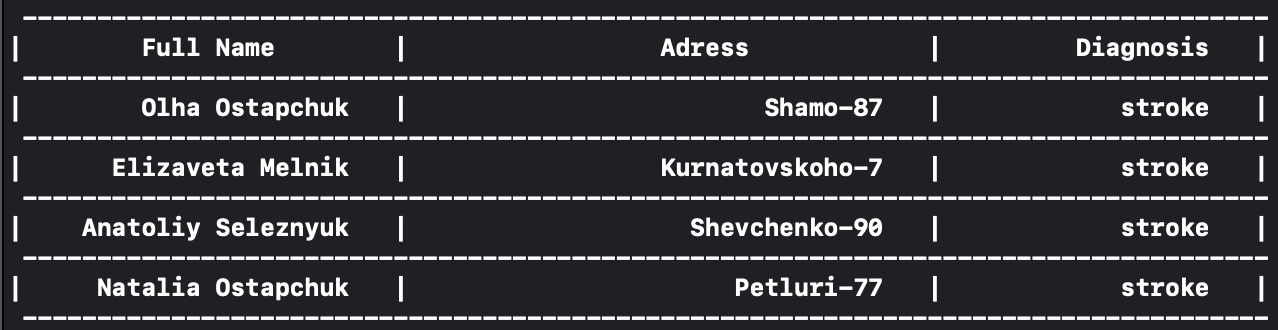


Рисунок 2.3 – Пошук за діагнозом

Пункт третій пропонує додати елемент в список наведено на рисунку 2.3.

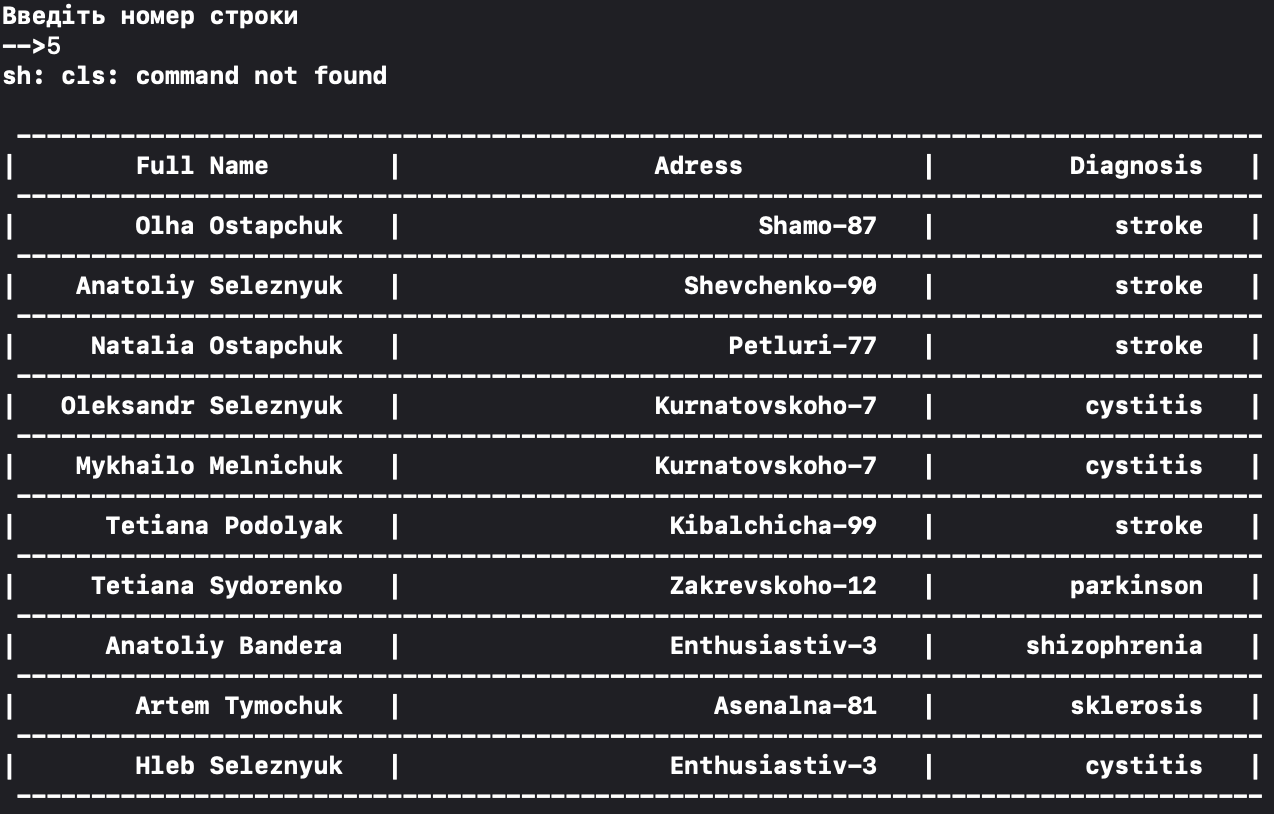


Рисунок 2.4 – добавлення елементу

**Висновок**

Підчас цієї лабораторної роботи було засвоєно базові навички по роботі з асоціацією, агрегацію, там композицією їх створенню та роботами над ними.

**Додаток А**

Lab.hpp

#pragma once

#include <iostream>

#include <iomanip>

#include <string>

//#define N 10

**using** **namespace** std;

**const** **int** N = 10;

**class** patient {

**private**:

string surname;

string name;

string secname;

string adress;

**int** num;

string diagnosis;

**public**:

patient();

**void** setsurname(string valuesurname);

string getsurname();

**void** setname(string valuename);

string getname();

**void** setsecname(string valuesecname);

string getsecname();

**void** setadress(string valueadress);

string getadress();

**void** setnum(**int** valuenum);

**int** getnum();

**void** setdiagnosis(string valuediagnosis);

string getdiagnosis();

**void** data(patient patient[]);

**void** print();

**void** show(patient patient[]);

**void** search\_by\_diagnisis(patient patient[]);

**void** search\_by\_num(patient patient[]);

**void** main(patient patient[]);

**void** append(patient patient[]);

**void** remove(patient patient[]);

};

**Додаток Б**

lab.cpp

#include "lab.hpp"

patient::patient(){

surname = "Шевченко";

name = "Тарас";

secname = "Григорович";

adress = "Шевченко-93";

num = 0;

diagnosis = "Перелом";

}

**void** patient::setsurname(string valuesurname){

surname = valuesurname;

}

string patient::getsurname(){

**return** surname;

}

**void** patient::setname(string valuename){

name = valuename;

}

string patient::getname(){

**return** name;

}

**void** patient::setsecname(string valuesecname){

secname = valuesecname;

}

string patient::getsecname(){

**return** secname;

}

**void** patient::setadress(string valueadress){

adress = valueadress;

}

string patient::getadress(){

**return** adress;

}

**void** patient::setnum(**int** valuenum){

num = valuenum;

}

**int** patient::getnum(){

**return** num;

}

**void** patient::setdiagnosis(string valuediagnosis){

diagnosis = valuediagnosis;

}

string patient::getdiagnosis(){

**return** diagnosis;

}

**void** patient:: data(patient patient[]){

**static** string surname\_rand[] = {"Шевченко", "Дашкевич", "Остапчук", "Сидорчук", "Науменко"};

**static** string name\_rand[] = {"Сидор", "Семен", "Тарас", "Вуйко", "Петро", "Денис"};

**static** string secname\_rand[] = {"Андрійович", "Михайлович", "Сергійович", "Вікторович", "Васильвоич"};

**static** string adress\_rand[] = {"Шевченко-93", "Хрещатик-26", "Нікітіна-27", "Либідска-33", "Сверстюка-8"};

**static** string diagnosis\_rand[] = {"Перелом", "Інфаркт", "Опухоль", "Склероз", "Інсульт"};

**for** (**int** i = 0; i < N; i++) {

patient[i].surname = surname\_rand[rand() % 5];

patient[i].name = name\_rand[rand() % 5];

patient[i].secname = secname\_rand[rand() % 5];

patient[i].adress = adress\_rand[rand() % 5];

patient[i].num = rand() % 1000 + 1000000;;

patient[i].diagnosis = diagnosis\_rand[rand() % 5];

}

cout << "Спісок паціентів створений" << endl;

}

**void** patient::print(){

cout << setw(4) << "-------------------------------------------------------------------------------------\n";

cout << setw(4) << "| Список пацієнтів |\n";

cout << setw(4) << "-------------------------------------------------------------------------------------\n";

}

**void** patient::show(patient patient[]){

**for** (**int** i = 0; i < N; i++) {

cout << "|"<<setw(10) << patient[i].surname << setw(4);

cout << "|" << setw(4) << patient[i].name << setw(2);

cout << "|" << setw(6) << patient[i].secname << setw(6);

cout << "|" << setw(3) << patient[i].adress <<setw(6);

cout << "|" << setw(3) << patient[i].num <<setw(6);

cout << "|" << setw(3) << patient[i].diagnosis <<setw(13);

cout<<"|\n";

cout<< "-------------------------------------------------------------------------------------\n";

}

}

**void** patient::search\_by\_diagnisis(patient patient[]){

string search;

cout << "Введіть діагноз: " << endl;

cin >> search;

cout << setw(4) << "-------------------------------------------------------------------------------------\n";

cout << setw(4) << "| Список пацієнтів |\n";

cout << setw(4) << "-------------------------------------------------------------------------------------\n";

**for** (**int** i = 0; i < N; i++) {

**if** (search == patient[i].diagnosis) {

cout << "|"<<setw(10) << patient[i].surname << setw(4);

cout << "|" << setw(4) << patient[i].name << setw(2);

cout << "|" << setw(6) << patient[i].secname << setw(6);

cout << "|" << setw(3) << patient[i].adress <<setw(6);

cout << "|" << setw(3) << patient[i].num <<setw(6);

cout << "|" << setw(3) << patient[i].diagnosis <<setw(13);

cout<<"|\n";

cout<< "-------------------------------------------------------------------------------------\n";

}

}

}

**void** patient::search\_by\_num(patient patient[]){

**int** num1;

**int** num2;

cout << "Введіть перший номер картки для інтервалу: " << endl;

cin >> num1;

cout << "Введіть другий номер картки для інтервал: " << endl;

cin >> num2;

cout << setw(4) << "-------------------------------------------------------------------------------------\n";

cout << setw(4) << "| Список пацієнтів |\n";

cout << setw(4) << "-------------------------------------------------------------------------------------\n";

**for** (**int** i = 0; i < N; i++) {

**if** (num1 < patient[i].num && num2 > patient[i].num) {

cout << "|"<<setw(10) << patient[i].surname << setw(4);

cout << "|" << setw(4) << patient[i].name << setw(2);

cout << "|" << setw(6) << patient[i].secname << setw(6);

cout << "|" << setw(3) << patient[i].adress <<setw(6);

cout << "|" << setw(3) << patient[i].num <<setw(6);

cout << "|" << setw(3) << patient[i].diagnosis <<setw(13);

cout<<"|\n";

cout<< "-------------------------------------------------------------------------------------\n";

}

}

}

**void** patient::main(patient patient[]){

**while** (**true**) {

**int** button;

cout << "Створити список паціентів - 1\nПечать - 2\nПошук за діагнозом - 3\nПошук за ноиером картки - 4 \nВидалити елемент - 5\nДобавити елемент - 6\nвихід - 0 \n\n-->"; cin >> button;

**if** (button == 1) {

patient-> data(patient);

}

**else** **if** (button == 2) {

patient-> print();

patient-> show(patient);

}

**else** **if**(button == 3){

patient-> search\_by\_diagnisis(patient);

}

**else** **if** (button == 4){

patient-> search\_by\_num(patient);

}

**else** **if** (button == 5){

patient->remove(patient);

}

**else** **if** (button == 6){

patient->append(patient);

}

**else** **if** (button == 0){

exit(0);

}

}

}

**void** patient::append(patient patient[]){

**int** k;

cout << "Скільки елементів ви бажаєте додати?" << endl;

cin >> k;

cout << setw(4) << "-------------------------------------------------------------------------------------\n";

cout << setw(4) << "| Список пацієнтів |\n";

cout << setw(4) << "-------------------------------------------------------------------------------------\n";

**for** (**int** i = 0; i < N; i++) {

cout << "|"<<setw(10) << patient[i].surname << setw(4);

cout << "|" << setw(4) << patient[i].name << setw(2);

cout << "|" << setw(6) << patient[i].secname << setw(6);

cout << "|" << setw(3) << patient[i].adress <<setw(6);

cout << "|" << setw(3) << patient[i].num <<setw(6);

cout << "|" << setw(3) << patient[i].diagnosis <<setw(13);

cout<<"|\n";

cout<< "-------------------------------------------------------------------------------------\n";

}

**for** (**int** i = 0; i < k; i++) {

cout << "|"<<setw(10) << surname << setw(4);

cout << "|" << setw(4) << name << setw(2);

cout << "|" << setw(6) << secname << setw(6);

cout << "|" << setw(3) << adress <<setw(6);

cout << "|" << setw(3) << num <<setw(6);

cout << "|" << setw(3) << diagnosis <<setw(13);

cout<<"|\n";

cout<< "-------------------------------------------------------------------------------------\n";

}

}

**void** patient::remove(patient patient[]){

**int** k;

**int** i;

cout << "How many elements do you want to delete(left to right)? ";

cin >> k;

cout << "Choose the position where you want to delete " << k << " elements ";

cin >> i;

**for** (; i < N - k; i++) {

patient[i] = patient[i + k];

}

cout << setw(4) << "-------------------------------------------------------------------------------------\n";

cout << setw(4) << "| Список пацієнтів |\n";

cout << setw(4) << "-------------------------------------------------------------------------------------\n";

**for** (i = 0; i < N - k; i++) {

cout << "|"<<setw(10) << patient[i].surname << setw(4);

cout << "|" << setw(4) << patient[i].name << setw(4);

cout << "|" << setw(6) << patient[i].secname << setw(6);

cout << "|" << setw(3) << patient[i].adress <<setw(6);

cout << "|" << setw(3) << patient[i].num <<setw(6);

cout << "|" << setw(3) << patient[i].diagnosis <<setw(12);

cout<<"|\n";

cout<< "-------------------------------------------------------------------------------------\n";

}

}