МІНІСТЕРСТВО ОСВІТИ І НАУКИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ “ХПІ”

Кафедра “Обчислювальна техніка та програмування”

Розрахункове завдання з програмування

Тема: «РОЗРОБКА ІНФОРМАЦІЙНО-ДОВІДКОВОЇ СИСТЕМИ»

Пояснювальна записка

1КІТ.102.8А. 18036-01 81 01-1 –АЗ

Розробник

Виконав:

студент групи 1КІТ-102.8А

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Кліщов Б.Р./

Перевірив:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Старший викладач. Молчанов Г.І./

Харків 2019

ЗАТВЕРДЖЕНО

1КІТ102.8А.18036-01 81 01-1 –АЗ

Розрахункове завдання з дисципліни

«Програмування»

Пояснювальна записка

1КІТ.102.8А.18036-01 81 01-1 -АЗ

Листів 20

Харків 2019

РОЗРАХУНКОВОГО ЗАВДАННЯ З ДИСЦИПЛІНИ

«ПРОГРАМУВАННЯ»

*Тема роботи.* Розробка інформаційно-довідкової системи.

*Мета роботи.* Закріпити отримані знання з дисципліни «Програмування» шляхом виконання типового комплексного завдання.

1 ВИМОГИ

**1.1 Розробник**

- Кліщов Богдан Романович;

- Студент групи КІТ 102.8(а);

- 07-06-2019р..

**1.2 Загальне завдання**

*Завдання до роботи:*

Кожний студент отримує індивідуальне завдання. Варіант завдання обирається за номером прізвища студента у журналі групи. При виконанні завдання з розробки інформаційно-довідкової системи необхідно виконати наступне:

1) з табл. 1, відповідно до варіанта завдання, обрати прикладну галузь;

2) дослідити літературу стосовно прикладної галузі. За результатами аналізу літератури оформити перший, аналітичний розділ пояснювальної записки обсягом 2–3 сторінки;

3) для прикладної галузі розробити розгалужену ієрархію класів, яка складається з не менш ніж трьох класів, один з яких є «батьком» для інших (класів-спадкоємців). Класи повинні мати перевантажені оператори введення-виведення даних та порівняння;

4) розробити клас-контролер, що буде включати колекцію розроблених класів, та наступні методи роботи з цією колекцією:

а) читання даних з файлу та їх запис у контейнер;

б) запис даних з контейнера у файл;

в) сортування елементів у контейнері за вказаними критеріями: поле та напрям сортування, які задаються користувачем з клавіатури;

г) пошук елементів за вказаним критерієм (див. «Завдання для обходу колекції» в табл. 1);

5) розробити клас, який має відображати діалогове меню для демонстрації реалізованих функцій класу контролера;

6) оформити схеми алгоритмів функцій класів контролера та діалогового меню;

7) оформити документацію: пояснювальну записку (див. розділ 2 даних методичних вказівок).

*Увага.* Текст програми та результати роботи програми мають бути подані в додатках.

*Вимоги:*

− усі класи повинні мати конструктори та деструктори;

− якщо функція не змінює поля класу, вона має бути декларована як константна;

− рядки повинні бути типу string;

− при перевантаженні функції треба використовувати ключове слово override;

− програмний код усіх класів має бути 100 % doxygenдокументований;

− у звіті текст програми слід оформляти стилем Courier new 8 пт, інтервал – одиничний; довжина рядка не повинна перевищувати 80 символів.

*Додаткові вимоги на оцінку «добре»:*

− виконання основного завдання та додаткових наступних вимог:

− додати обробку помилок; при цьому функція, що генерує виключення, при її декларуванні повинна мати ключове слово throw;

− виконати перевірку вхідних даних за допомогою регулярних виразів.

*Додаткові вимоги на оцінку «відмінно»:*

− виконати завдання відповідно до вимог на оцінку «добре» та додаткові наступні вимоги:

− критерій для пошуку та сортування задавати у вигляді функтора;

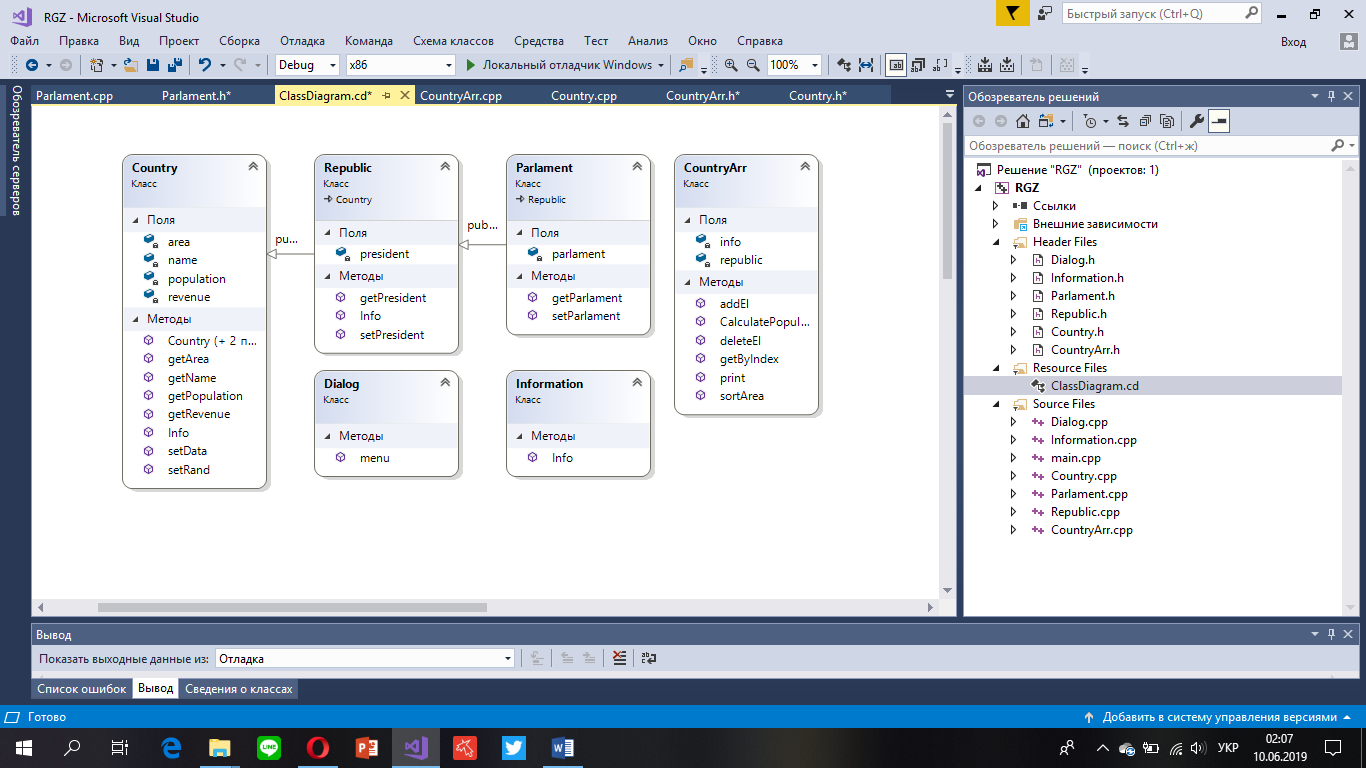
− розробити клас-тестер, основною метою якого буде перевірка коректності роботи класу-контролера.

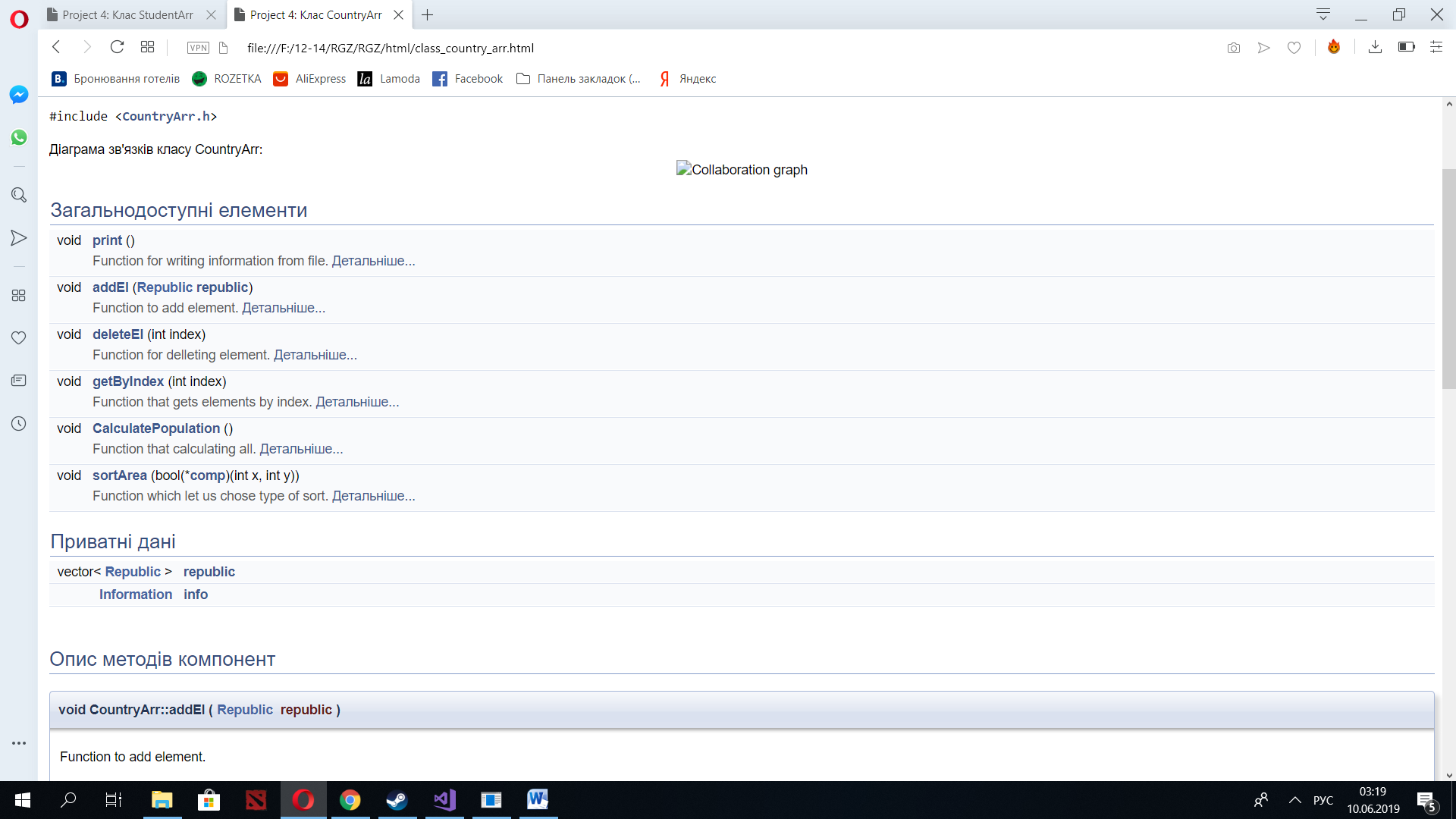
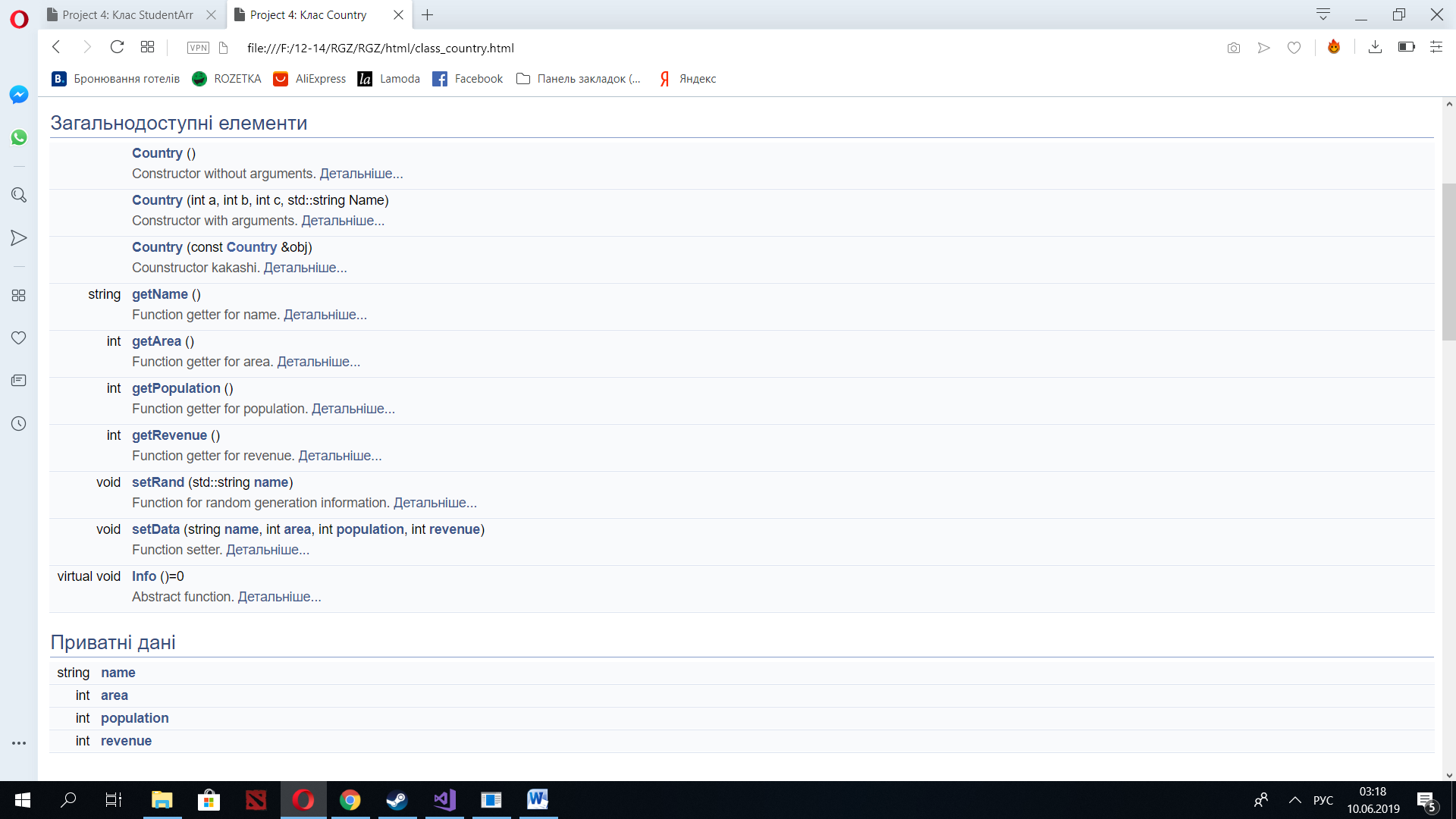
2 ОПИС ПРОГРАМИ

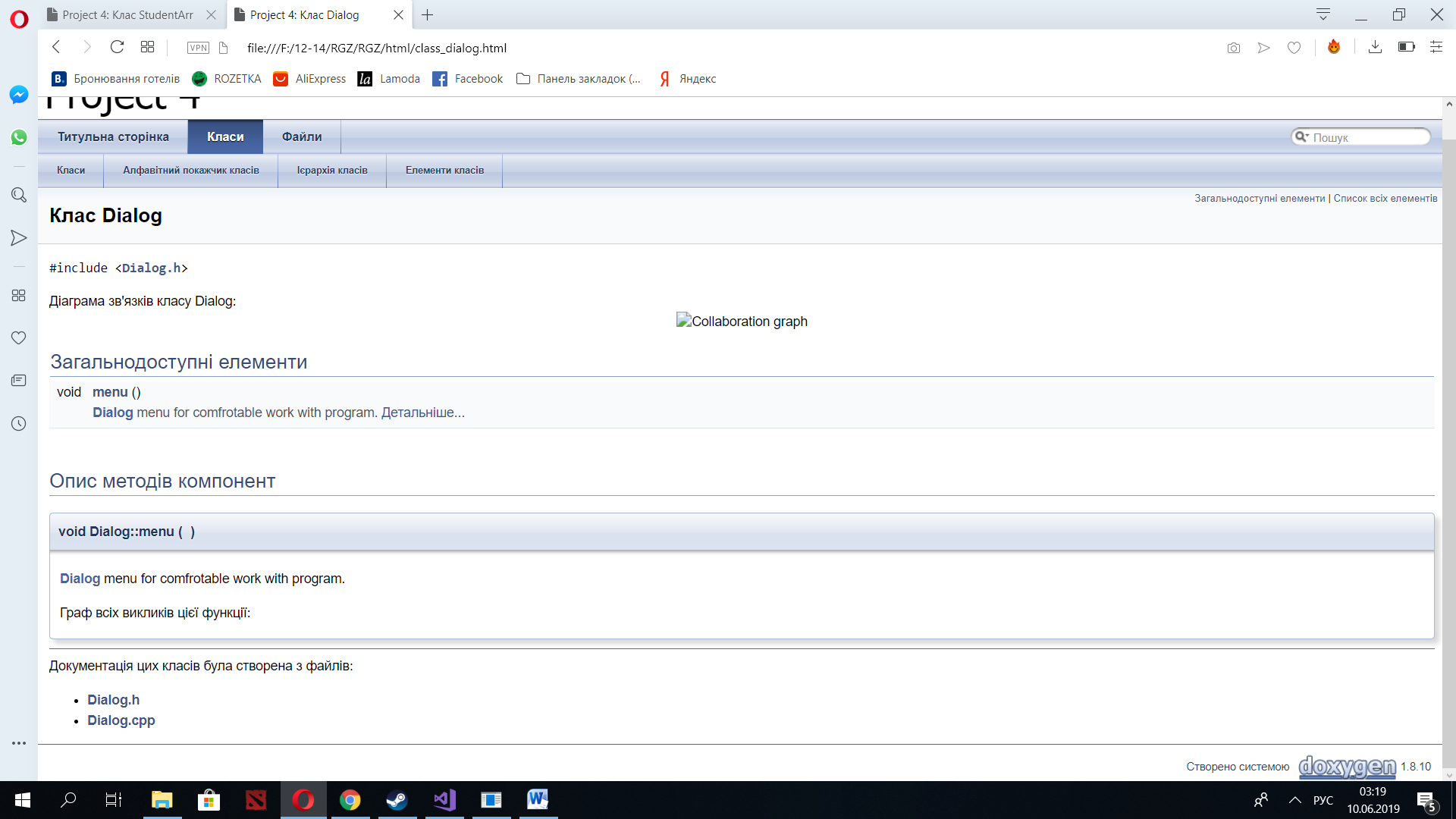
**2.1 Функціональне призначення**

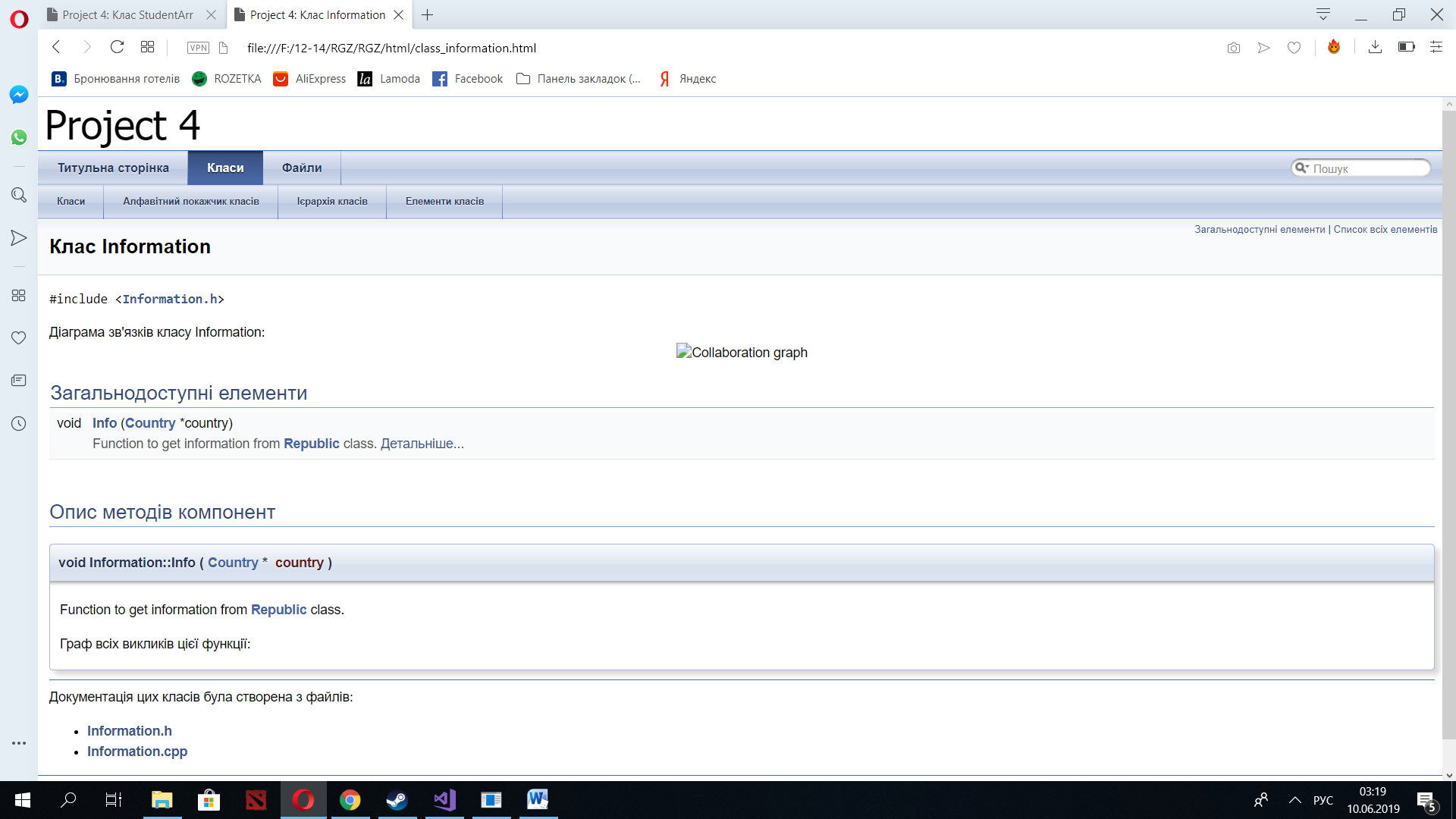
Програма призначена для отримання даних про країн світу

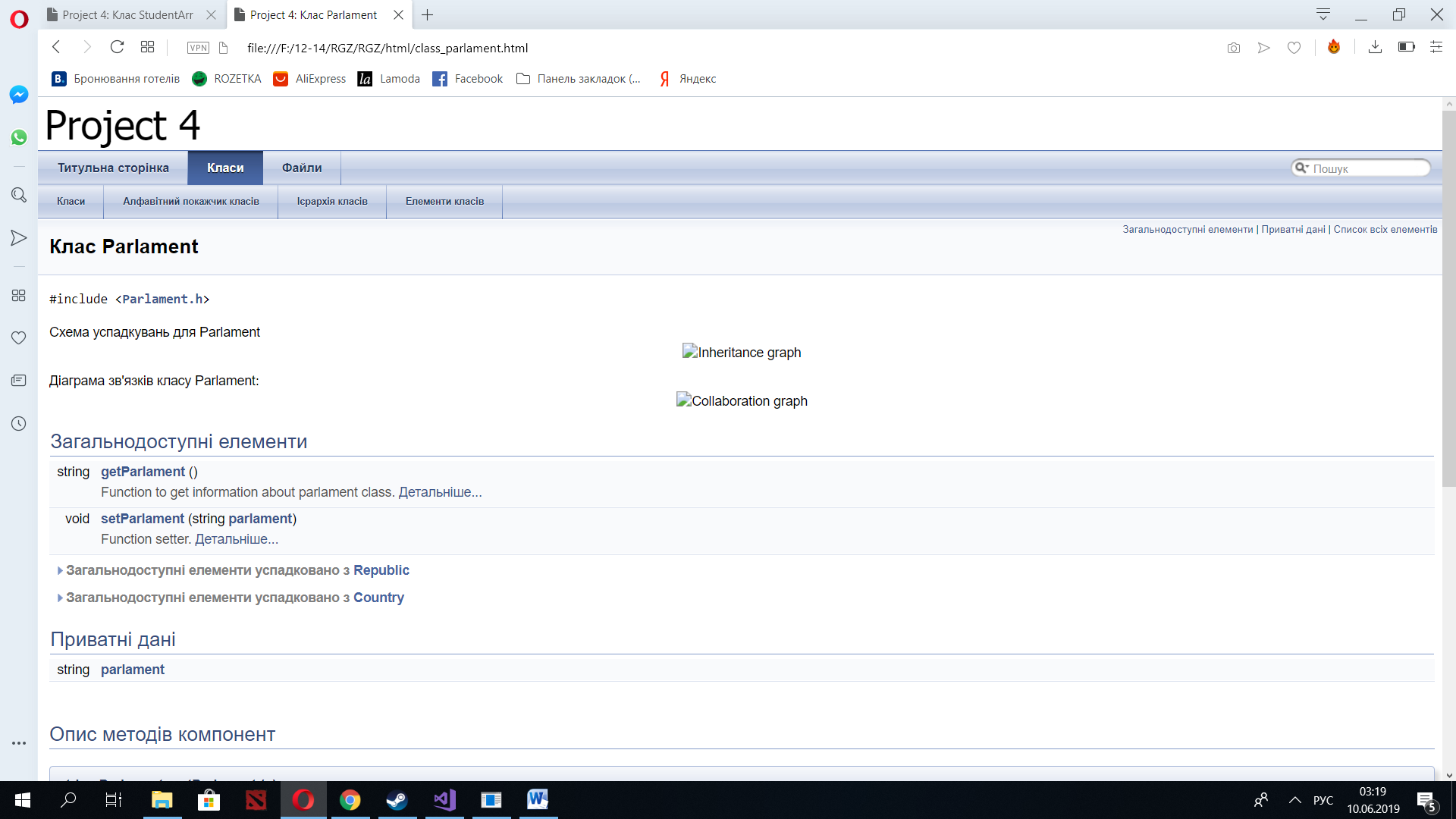
**2.2 Опис логічної структури**

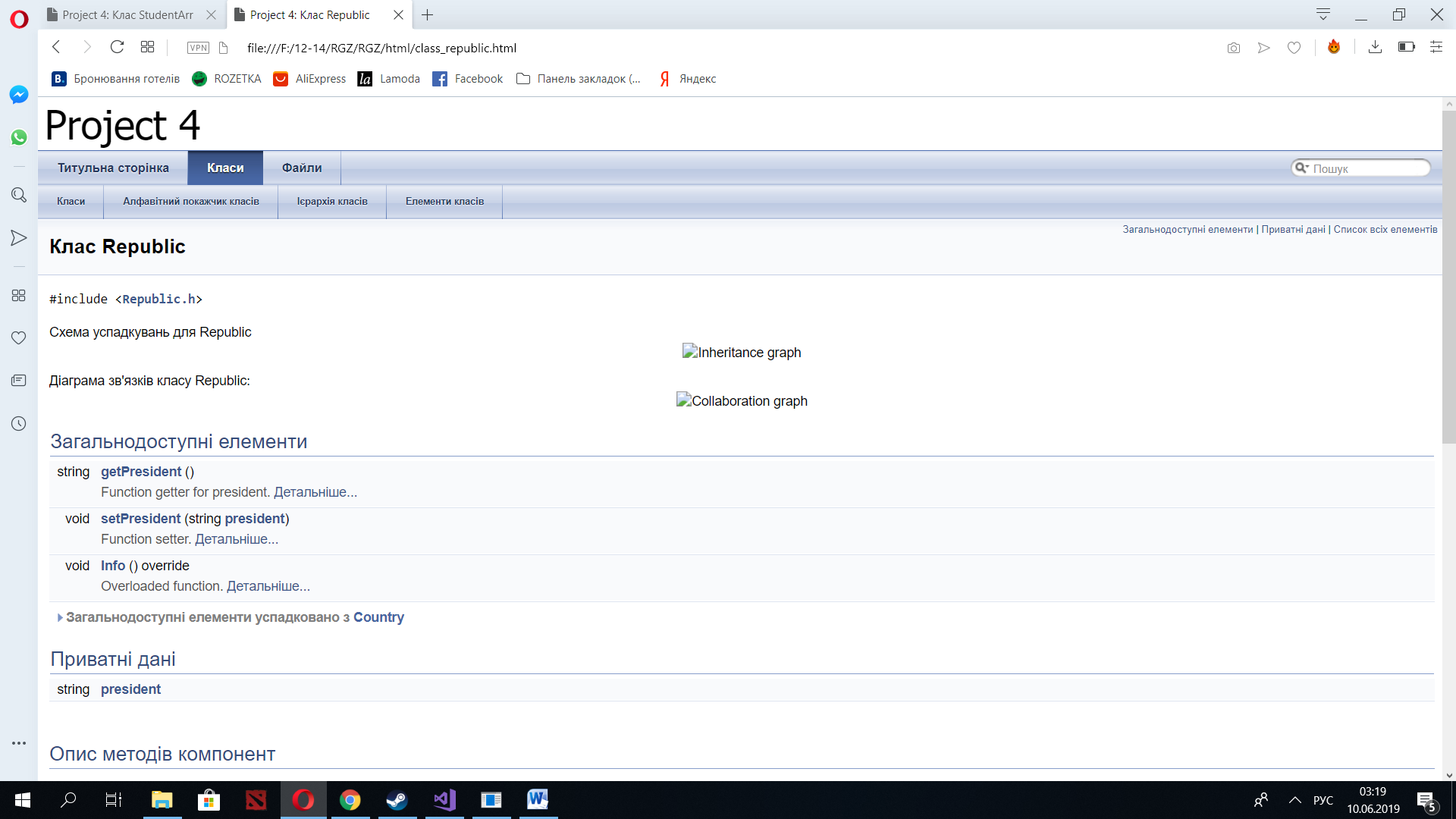












**3 ВАРІАНТИ ВИКОРИСТАННЯ**

**3.1 Ілюстрація роботи програми**

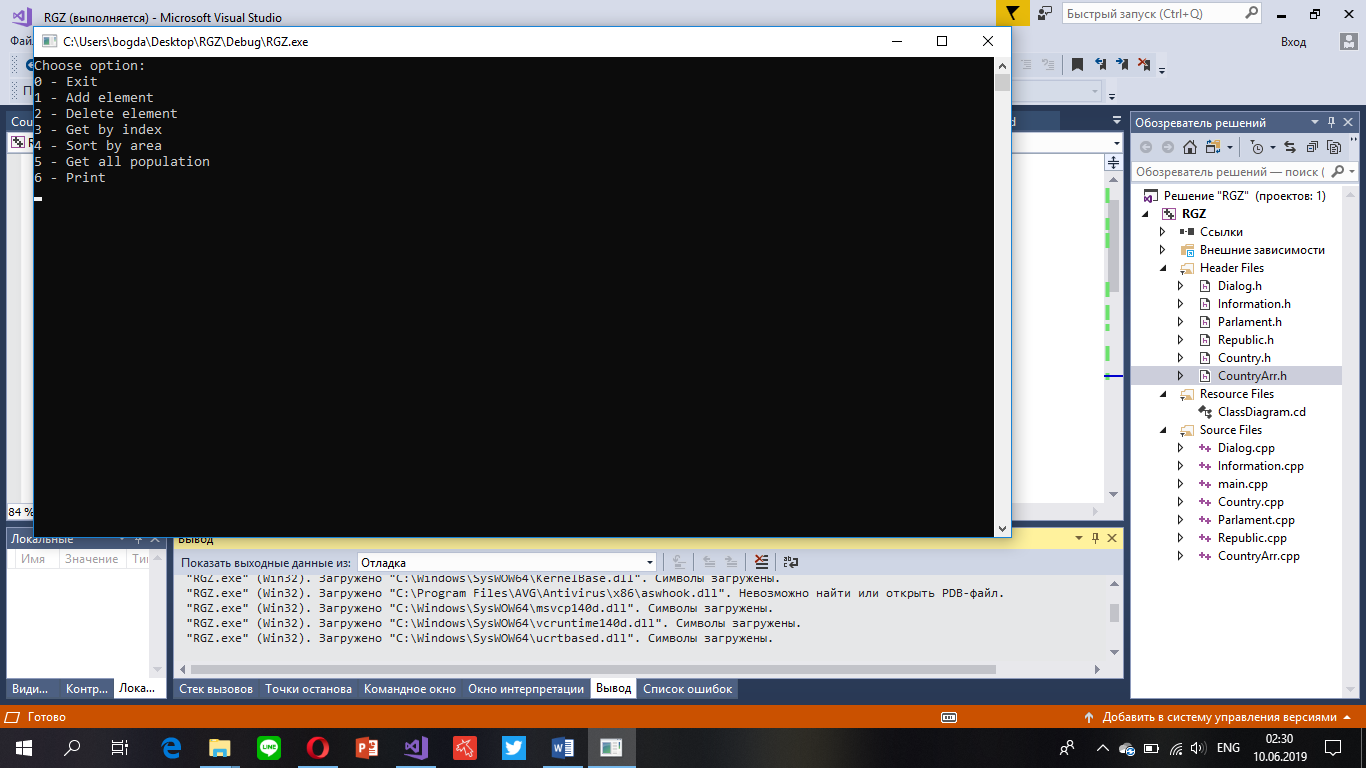


Рисунок 3.1 – Діалогове меню

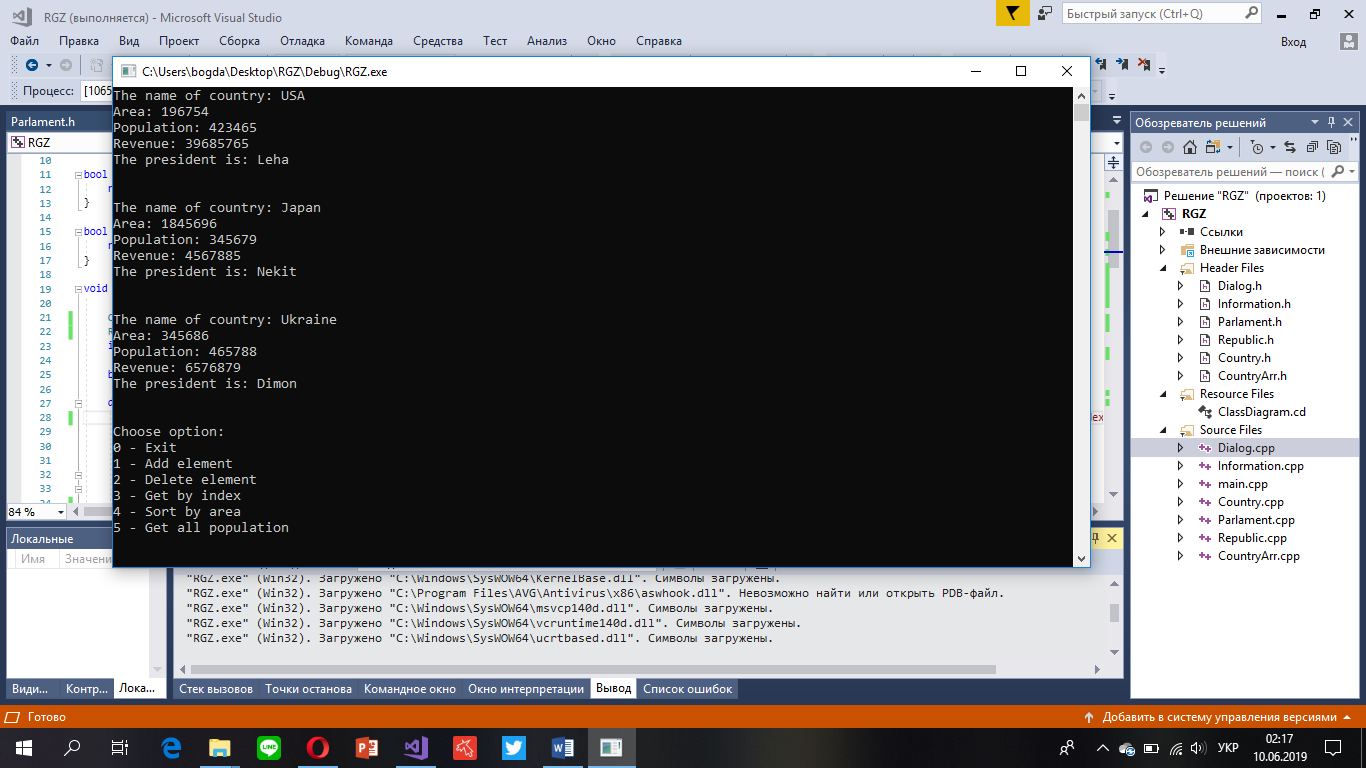


Рисунок 3.2 – Читання даних з файлу

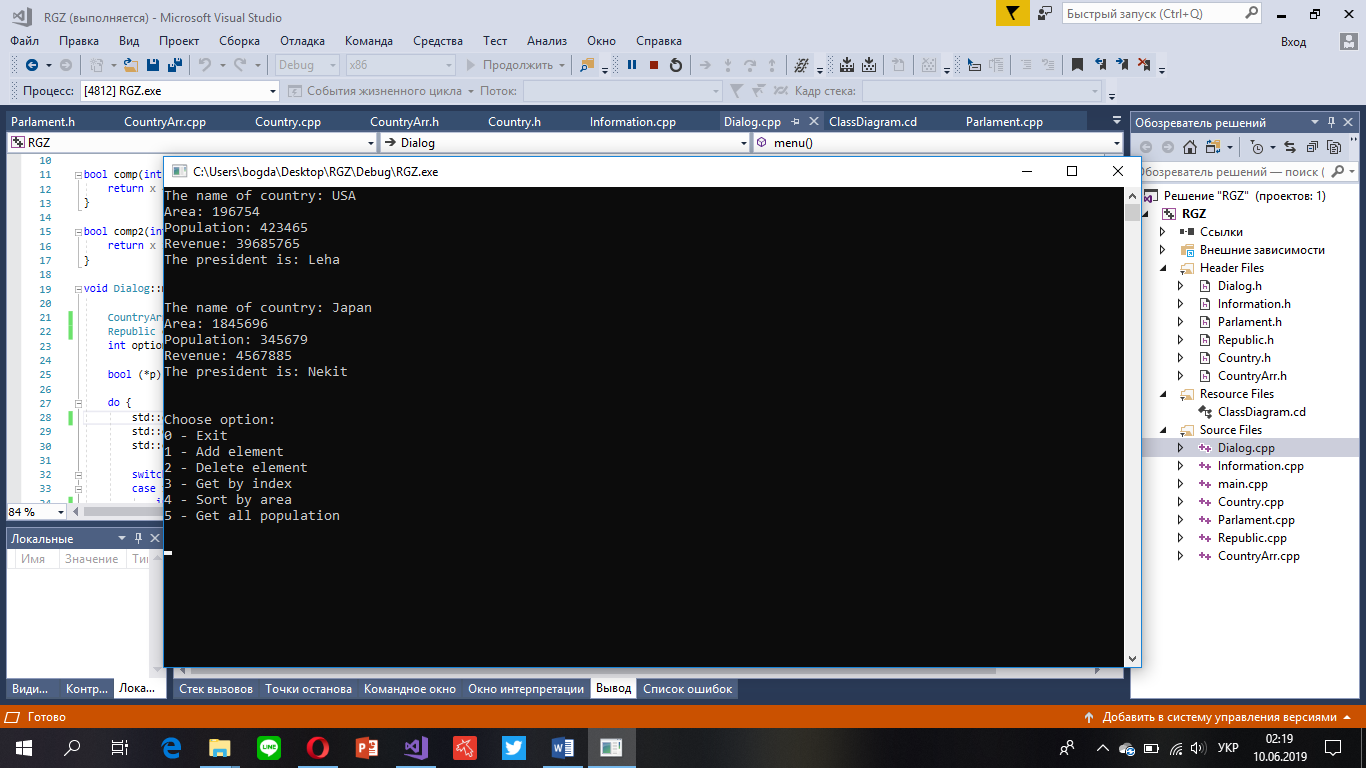


Рисунок 3.3 – Видалення елементу за індексом

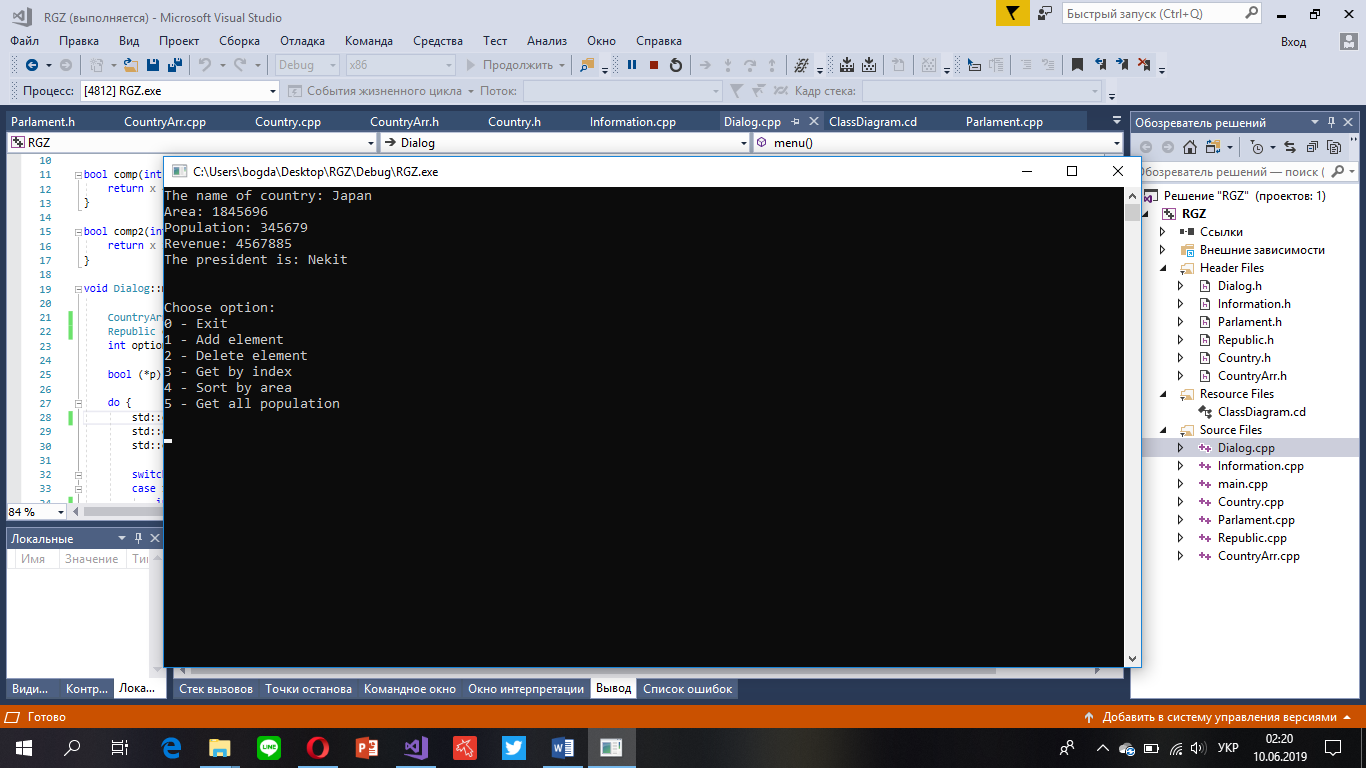


Рисунок 3.4 – Отримання елементу за індексом

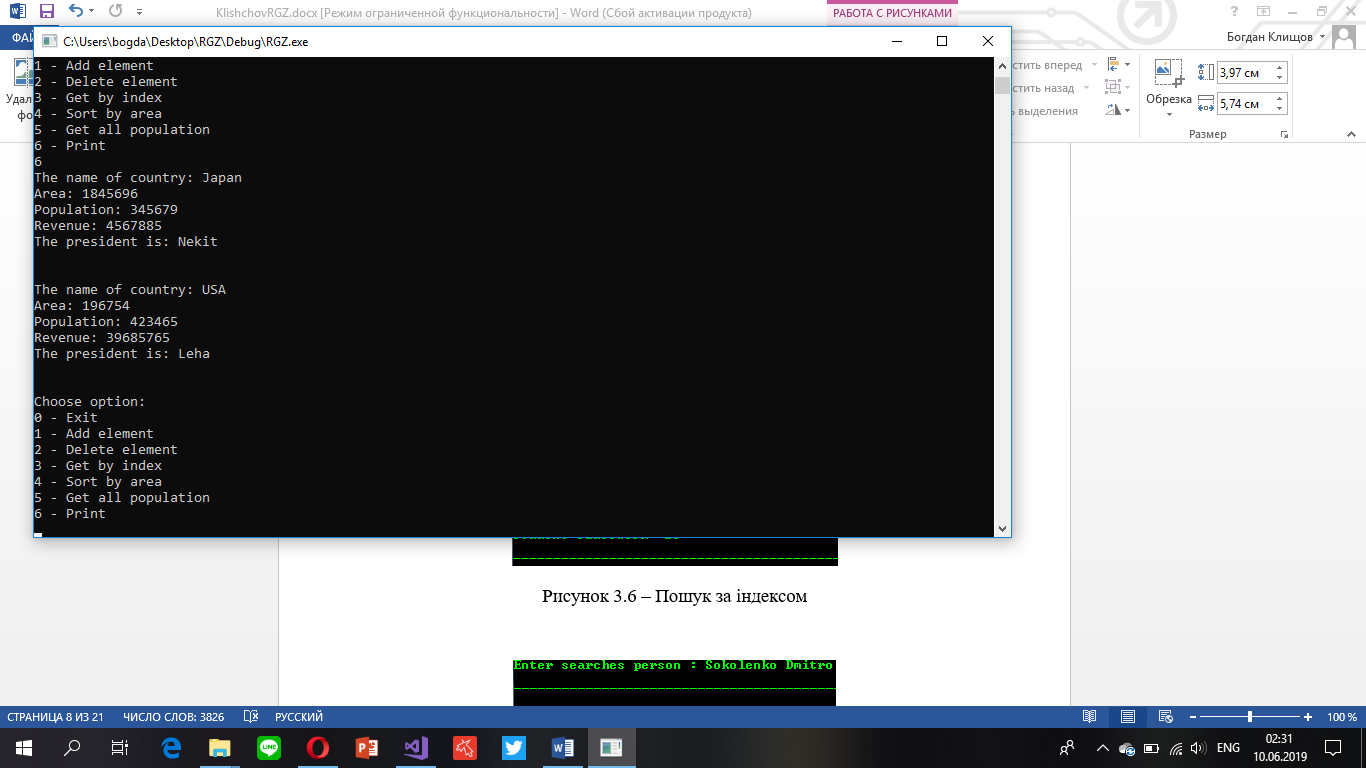


Рисунок 3.5 – Сортування за спаданням по площі країни

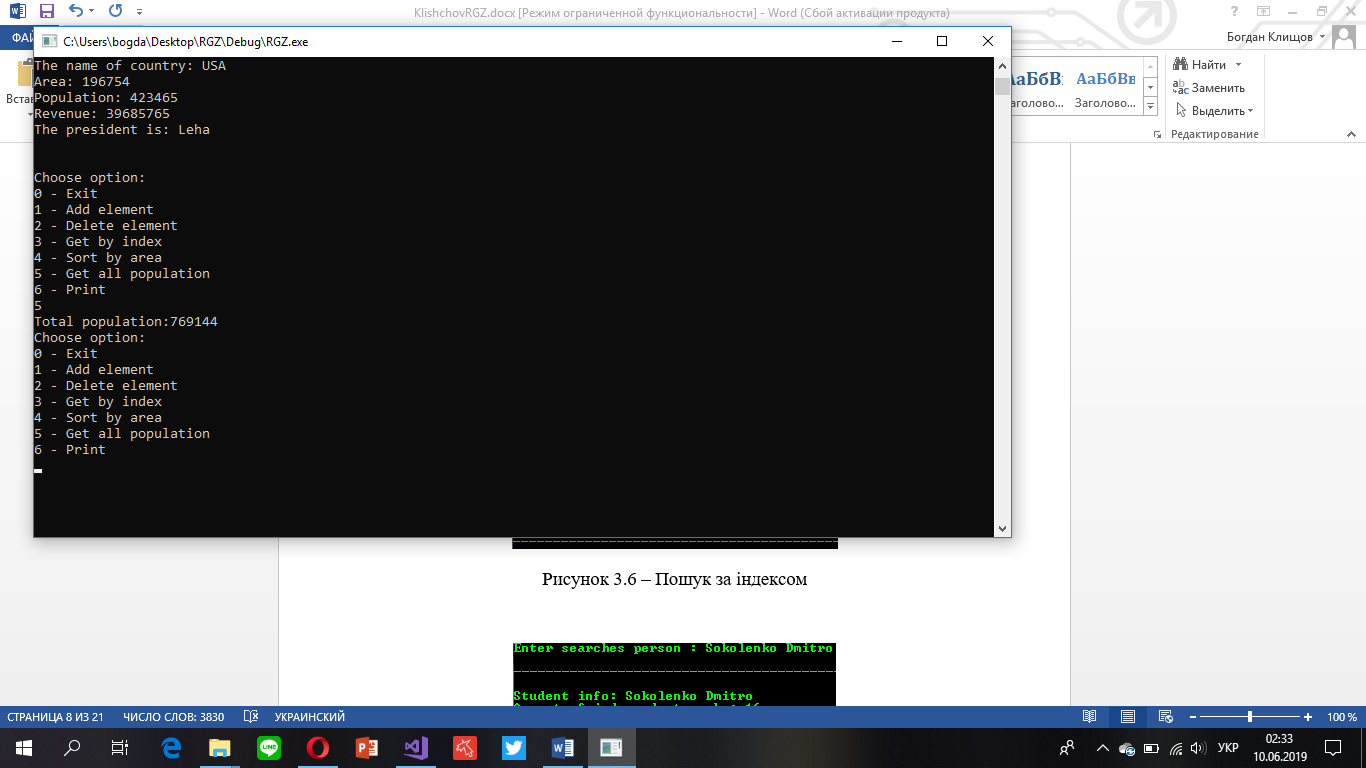


Рисунок 3.6 – Обчислення населення з усіх країн

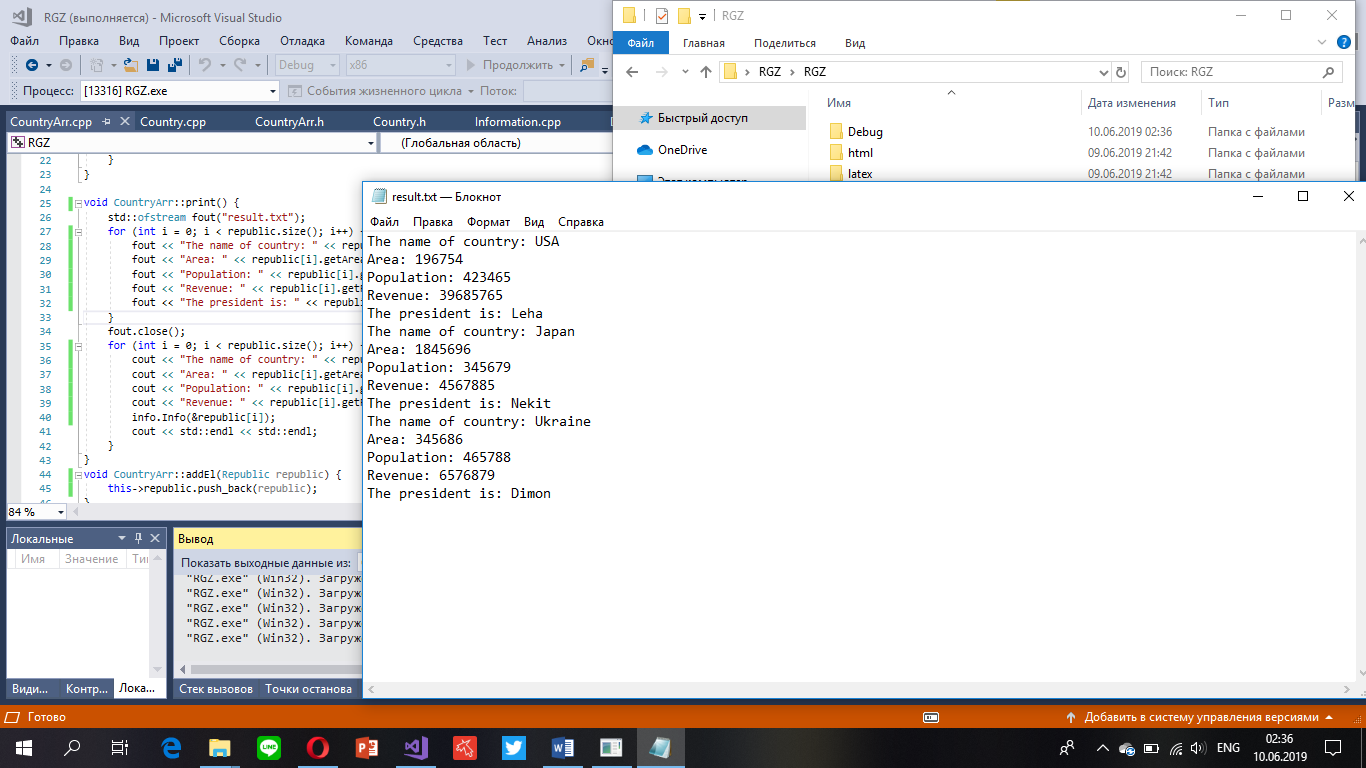


Рисунок 3.7 – Запис в файл

**ВИСНОВОК**

В ході виконання поставленої задачі були закріплені отримані знання з дисципліни «Програмування» шляхом виконання типового комплексного завдання.

Main.cpp

/\*\*

\* @file main.cpp

\* Implementation of all functions of Information class.

\* @author Klishchov Bohdan

\* @version 1.0

\* @date 2019.09.06

\*/

#include "CountryArr.h"

#include "Dialog.h"

int main() {

CountryArr country;

Dialog dialog;

dialog.menu();

return 0;

}

Country.h

/\*\*

\* @file Country.h

\* Declaration of Country class.

\* @author Klishchov Bohdan

\* @version 1.0

\* @date 2019.09.06

\*/

#pragma once

#include <iostream>

#include <ctime>

#include <fstream>

#include <string>

#include <cstdio>

#include <regex>

#include <vector>

using std::string;

using std::cin;

using std::cout;

using std::vector;

using std::endl;

class Country {

private:

string name; // Строкове поле країни (Назва країни)

int area; // Числове поле країни (Площа країни)

int population; // Числове поле країни (Населення країни)

int revenue; // Числове поле країни (Населення країни)

public:

/\*\*

\* Constructor without arguments.

\*/

Country();

/\*\*

\* Constructor with arguments.

\*/

Country(int a, int b, int c, std::string Name);

/\*\*

\* Counstructor kakashi.

\*/

Country(const Country &obj);

/\*\*

\* Function getter for name.

\*/

string getName();

/\*\*

\* Function getter for area.

\*/

int getArea();

/\*\*

\* Function getter for population.

\*/

int getPopulation();

/\*\*

\* Function getter for revenue.

\*/

int getRevenue();

/\*\*

\* Function for random generation information.

\*/

void setRand(std::string name);

/\*\*

\* Function setter.

\* \param string name, int area, population, revenue.

\*/

void setData(string name, int area, int population, int revenue);

/\*\*

\* Abstract function.

\*/

virtual void Info() = 0;

};

Country.cpp

/\*\*

\* @file Country.cpp

\* Implementation of all functions of Country class.

\* @author Klishchov Bohdan

\* @version 1.0

\* @date 2019.09.06

\*/

#include "Country.h"

#include"Country.h"

Country::Country() :population(0), area(0), revenue(0), name() {

};

Country::Country(int a, int b, int c, std::string Name) :population(a), area(b), revenue(c) {

name = Name;

};

Country::Country(const Country &obj) :population(obj.population), area(obj.area), revenue(obj.revenue), name(obj.name) {};

std::string Country::getName() {

return this->name;

}

int Country::getArea() {

return this->area;

}

int Country::getPopulation() {

return this->population;

}

int Country::getRevenue() {

return this->revenue;

}

void Country::setRand(std::string name) {

this->area = rand() % 654365 + 65436;

this->population = rand() % 654236 + 85676;

this->revenue = rand() % 45678 + 34567;

this->name = name;

}

void Country::setData(string name, int area, int population, int revenue) {

this->name = name;

this->population = population;

this->area = area;

this->revenue = revenue;

}

CountryArr.h

/\*\*

\* @file CountryArr.h

\* Declaration of StudentArr class.

\* @author Klishchov Bohdan

\* @version 1.0

\* @date 2019.09.06

\*/

#pragma once

#include "Republic.h"

#include "Information.h"

class CountryArr {

private:

vector<Republic> republic;

Information info;

public:

/\*\*

\* Function for writing information from file.

\*/

void print();

/\*\*

\* Function to add element.

\* @param Republic republic.

\*/

void addEl(Republic republic);

/\*\*

\* Function for delleting element.

\* @param int index.

\*/

void deleteEl(int index);

/\*\*

\* Function that gets elements by index.

\* @param int index.

\*/

void getByIndex(int index);

/\*\*

\* Function that calculating all.

\*/

void CalculatePopulation();

/\*\*

\* Function which let us chose type of sort.

\* \param \*comp, int x, int y.

\*/

void sortArea(bool(\*comp)(int x, int y));

};

CountryArr.cpp

/\*\*

\* @file StudentArr.cpp

\* Implementation of all functions of CountryArr class.

\* @author Klishchov Bohdan

\* @version 1.0

\* @date 2019.09.06

\*/

#include "CountryArr.h"

void CountryArr::sortArea(bool(\*comp)(int x, int y)) {

Republic temp;

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

for (int j = 0; j < republic.size(); j++) {

if (comp(republic[i].getArea(), republic[j].getArea())) {

temp = republic[i];

republic[i] = republic[j];

republic[j] = temp;

}

}

}

}

void CountryArr::print() {

std::ofstream fout("result.txt");

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

fout << "The name of country: " << republic[i].getName() << std::endl;

fout << "Area: " << republic[i].getArea() << std::endl;

fout << "Population: " << republic[i].getPopulation() << std::endl;

fout << "Revenue: " << republic[i].getRevenue() << std::endl;

fout << "The president is: " << republic[i].getPresident() << std::endl;

}

fout.close();

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

cout << "The name of country: " << republic[i].getName() << std::endl;

cout << "Area: " << republic[i].getArea() << std::endl;

cout << "Population: " << republic[i].getPopulation() << std::endl;

cout << "Revenue: " << republic[i].getRevenue() << std::endl;

info.Info(&republic[i]);

cout << std::endl << std::endl;

}

}

void CountryArr::addEl(Republic republic) {

this->republic.push\_back(republic);

}

void CountryArr::deleteEl(int index) {

republic.erase(republic.begin() + index);

}

void CountryArr::getByIndex(int index) { //change function name!!!

std::ofstream fout("resultID.txt");

fout << "The name of country: " << republic[index - 1].getName() << std::endl;

fout << "Area: " << republic[index - 1].getArea() << std::endl;

fout << "Population: " << republic[index - 1].getPopulation() << std::endl;

fout << "Revenue: " << republic[index - 1].getRevenue() << std::endl;

fout.close();

cout << "The name of country: " << republic[index - 1].getName() << std::endl;

cout << "Area: " << republic[index - 1].getArea() << std::endl;

cout << "Population: " << republic[index - 1].getPopulation() << std::endl;

cout << "Revenue: " << republic[index - 1].getRevenue() << std::endl;

info.Info(&republic[index - 1]);

cout << std::endl << std::endl;

}

void CountryArr::CalculatePopulation() {

auto popul = 0;

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

popul += republic[i].getPopulation();

}

cout << "Total population:" << popul << endl;

}

Dialig.h

/\*\*

\* @file Dialog.h

\* Declaration of Dialog class.

\* @author Klishchov Bohdan

\* @version 1.0

\* @date 2019.09.06

\*/

#pragma once

#include "CountryArr.h"

class Dialog {

public:

/\*\*

\* Dialog menu for comfrotable work with program.

\*/

void menu();

};

Dialog.cpp

/\*\*

\* @file Dialog.cpp

\* Implementation of all functions of Dialog class.

\* @author Klishchov Bohdan

\* @version 1.0

\* @date 2019.09.06

\*/

#include "Dialog.h"

bool comp(int x, int y) {

return x < y;

}

bool comp2(int x, int y) {

return x > y;

}

void Dialog::menu() {

CountryArr world;

Republic country;

int option = 0;

bool (\*p)(int, int);

do {

std::cout << "Choose option:" << std::endl << "0 - Exit " << std::endl << "1 - Add element" << std::endl << "2 - Delete element" << std::endl << "3 - Get by index" << std::endl << "4 - Sort by area" << std::endl << "5 - Get all population" << std::endl << "6 - Print";

std::cout << std::endl;

std::cin >> option;

switch (option) {

case 1: {

int area, population, revenue;

string name, president;

std::ifstream fin("data.txt");

fin >> area >> population >> revenue >> name >> president;

country.setData(name, area, population, revenue);

country.setPresident(president);

world.addEl(country);

fin >> area >> population >> revenue >> name >> president;

country.setData(name, area, population, revenue);

country.setPresident(president);

world.addEl(country);

fin >> area >> population >> revenue >> name >> president;

country.setData(name, area, population, revenue);

country.setPresident(president);

world.addEl(country);

system("cls");

world.print();

break;

}

case 2: {

auto id = 0;

cout << std::endl << "Enter index: ";

cin >> id;

cout << endl;

world.deleteEl(id);

system("cls");

world.print();

break;

}

case 3: {

auto index = 0;

std::cout << std::endl << "Enter index : ";

std::cin >> index;

std::cout << std::endl;

system("cls");

world.getByIndex(index);

break;

}

case 4: {

int ch;

std::cout << "Please, enter the type of sort" << std::endl << "1 - up, 0 - down: ";

std::cin >> ch;

std::cout << std::endl;

if (ch == 1) {

p = comp;

}

else if (ch == 0) {

p = comp2;

}

else {

std::cout << "You enter false variant" << std::endl;

break;

}

world.sortArea(p);

break;

}

case 5: {

world.CalculatePopulation();

break;

}

case 6: {

world.print();

}

default: {

break;

}

}

} while (option != 0);

}

Information.h

/\*\*

\* @file Information.h

\* Declaration of Information class.

\* @author Klishchov Bohdan

\* @version 1.0

\* @date 2019.09.06

\*/

#pragma once

#include "Country.h"

class Information {

public:

/\*\*

\* Function to get information from Republic class.

\*/

void Info(Country \*country);

};

Information.cpp

/\*\*

\* @file Information.cpp

\* Contain main function.

\* @author Klishchov Bohdan

\* @version 1.0

\* @date 2019.09.06

\*/

#include "Information.h"

void Information::Info(Country \*country)

{

country->Info();

}

Parlament.h

/\*\*

\* @file Parlament.h

\* Declaration of Parlament class.

\* @author Klishchov Bohdan

\* @version 1.0

\* @date 2019.09.06

\*/

#pragma once

#include "Republic.h"

class Parlament : public Republic {

private:

string parlament;

public:

/\*\*

\* Function to get information about parlament class.

\*/

string getParlament();

/\*\*

\* Function setter.

\* \param string parlament.

\*/

void setParlament(string parlament);

};

Parlament.cpp

/\*\*

\* @file Parlament.cpp

\* Implementation of all functions of Parlament class.

\* @author Klishchov Bohdan

\* @version 1.0

\* @date 2019.09.06

\*/

#include "Parlament.h"

string Parlament::getParlament() {

return this->parlament;

}

void Parlament::setParlament(string parlament) {

this->parlament = parlament;

}

Republic.h

/\*\*

\* @file Republic.h

\* Declaration of Republic class.

\* @author Klishchov Bohdan

\* @version 1.0

\* @date 2019.09.06

\*/

#pragma once

#include "Country.h"

class Republic : public Country {

private:

string president; //Строкове поле республіки(Ім'я президента)

public:

/\*\*

\* Function getter for president.

\*/

string getPresident();

/\*\*

\* Function setter.

\* \param string president.

\*/

void setPresident(string president);

/\*\*

\* Overloaded function.

\*/

void Info() override;

};

Republic.cpp

/\*\*

\* @file Republic.cpp

\* Implementation of all functions of Republic class.

\* @author Klishchov Bohdan

\* @version 1.0

\* @date 2019.09.06

\*/

#include "Republic.h"

string Republic::getPresident() {

return president;

}

void Republic::setPresident(string president) {

this->president = president;

}

void Republic::Info() {

cout << "The president is: " << president << endl;

}