**Лабораторна робота №1. Класи**

Мета: Отримати базові знання про класи. Дослідити механізм інкапсуляції.

**ВИМОГИ**

**1.1 Інформація про розробника:**

* Кліщов Б. Р.
* КІТ 102.8а
  1. **Загальне завдання**

Для предметної галузі з табл. 1.2 розробити два класи:

- клас, що відображає сутність «базового класу». При цьому, в даному класі повинно бути мінімум три числових поля (бажано, щоб одне з цих полів було унікальним ідентифікатором об’єкту);

- клас, що має в собі динамічний масив об’єктів базового класу та має в собі методи додавання, видалення елементу, отримання елементу по індексу (або ідентифікатору), вивід усіх елементів на екран.

* 1. **Додаткові умови виконання завдання:**

- усі поля «базового класу» повинні бути приватними, та мати публічні гетери та сетери (модифікатори доступу), використовувати механізм інкапсуляції);

усі функції, що не повинні змінювати поля поточного об’єкту, повинні бути константними;

- усі аргументи функцій, що не змінюються, по можливості також повинні бути константними. Якщо їх не можна зробити константними, в такому разі повинно бути обґрунтування цього;

- в класі-списку метод додавання елемента не повинен вводити дані з клавіатури або файлу, а повинен приймати вже готовий об’єкт для додавання. Метод вводу даних має бути відокремленим;

- продемонструвати відсутність витоків пам’яті;

- продемонструвати роботу розроблених методів класу-списку за допомогою модульних тестів.

**Індивідуальні завдання:**

В табл. 1.2 обрати прикладну галузь по варіанту у відповідності до номера у журналі групи.

1. **ОПИС ПРОГРАМИ**

**2.1 Функціональне призначення**

Програма призначена щоб отримувати інформацію щодо різних країн світу.

* 1. **Опис логічної структури**

На рисунку № 1 зображена діаграма класу

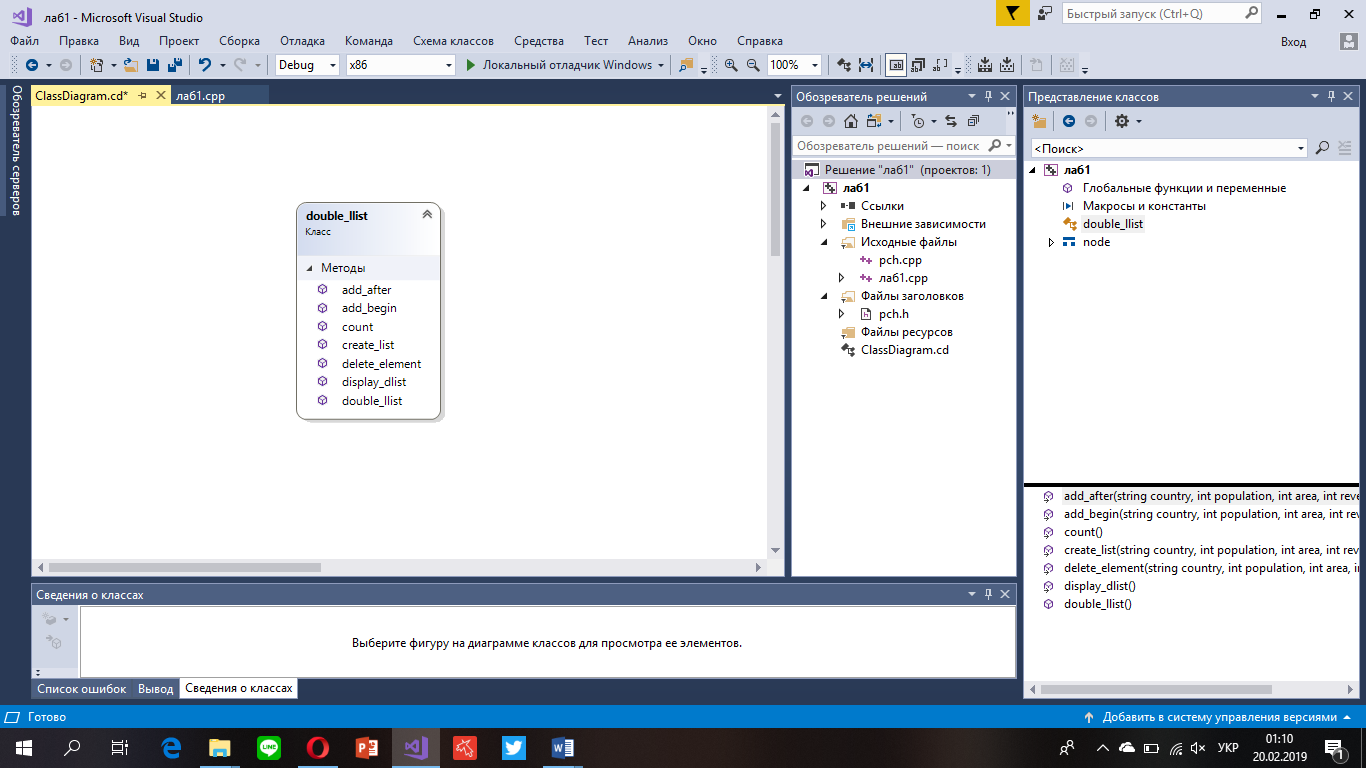


Рис №1. Діаграма класу

На рисунку № 2 зоображена структура програми:

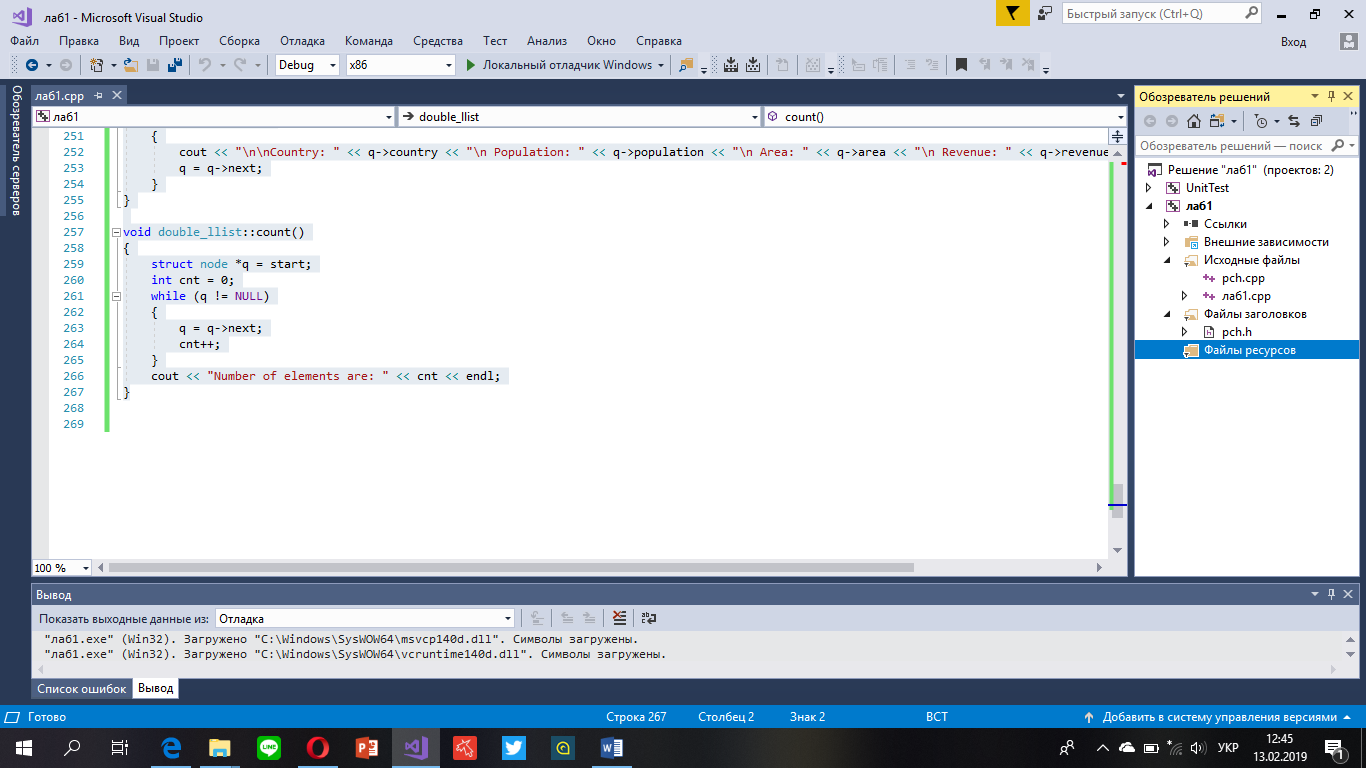


Рис №2: Cтруктура програми

**2.3 Важливі фрагменти програми**

Код програми:

#include "Func.h"

#include "pch.h"

#include <iostream>

#include <cstdio>

#include <cstdlib>

#include <string>

#include <time.h>

using namespace std;

struct node

{

string country;

int population;

int area;

int revenue;

struct node \*next;

struct node \*prev;

} \*start;

class double\_llist

{

public:

void create\_list(string country, int population, int area, int revenue);

void add\_begin(string country, int population, int area, int revenue);

void add\_after(string country, int population, int area, int revenue, int position);

void delete\_element(string country, int population, int area, int revenue);

void display\_dlist();

void count();

double\_llist()

{

start = NULL;

}

};

int main()

{

srand(time(NULL));

int choice, position;

int population, area, revenue;

string country;

double\_llist dl;

while (1)

{

cout << endl << "----------------------------" << endl;

cout << endl << " Operations" << endl;

cout << endl << "----------------------------" << endl;

cout << "1.Create Node" << endl;

cout << "2.Add at begining" << endl;

cout << "3.Add after position" << endl;

cout << "4.Delete" << endl;

cout << "5.Display" << endl;

cout << "6.Count" << endl;

cout << "7.Quit" << endl;

cout << "Enter your choice : ";

cin >> choice;

switch (choice)

{

case 1:

cout << "Enter the country name: ";

cin >> country;

population = rand() % 1000000 + 100000;

area = rand() % 100000 + 10000;

revenue = rand() % 10000 + 1000;

dl.create\_list(country, population, area, revenue);

cout << endl;

break;

case 2:

cout << "Enter the country name: ";

cin >> country;

population = rand() % 1000000 + 100000;

area = rand() % 100000 + 10000;

revenue = rand() % 10000 + 1000;

dl.add\_begin(country, population, area, revenue);

cout << endl;

break;

case 3:

cout << "Enter the country name: ";

cin >> country;

population = rand() % 1000000 + 100000;

area = rand() % 100000 + 10000;

revenue = rand() % 10000 + 1000;

cout << "Insert Element after postion: ";

cin >> position;

dl.add\_after(country, population, area, revenue, position);

cout << endl;

break;

case 4:

if (start == NULL)

{

cout << "List empty,nothing to delete" << endl;

break;

}

cout << "Enter the country name: ";

cin >> country;

dl.delete\_element(country, population, area, revenue);

cout << endl;

break;

case 5:

dl.display\_dlist();

cout << endl;

break;

case 6:

dl.count();

break;

case 7:

exit(1);

default:

cout << "Wrong choice" << endl;

}

}

return 0;

}

void double\_llist::create\_list(string country, int population, int area, int revenue)

{

struct node \*s, \*temp;

temp = new(struct node);

temp->country = country;

temp->population = population;

temp->area = area;

temp->revenue = revenue;

temp->next = NULL;

if (start == NULL)

{

temp->prev = NULL;

start = temp;

}

else

{

s = start;

while (s->next != NULL)

s = s->next;

s->next = temp;

temp->prev = s;

}

}

void double\_llist::add\_begin(string country, int population, int area, int revenue)

{

if (start == NULL)

{

cout << "First Create the list." << endl;

return;

}

struct node \*temp;

temp = new(struct node);

temp->prev = NULL;

temp->country = country;

temp->population = population;

temp->area = area;

temp->revenue = revenue;

temp->next = start;

start->prev = temp;

start = temp;

cout << "Element Inserted" << endl;

}

void double\_llist::add\_after(string country, int population, int area, int revenue, int pos)

{

if (start == NULL)

{

cout << "First Create the list." << endl;

return;

}

struct node \*tmp, \*q;

int i;

q = start;

for (i = 0; i < pos - 1; i++)

{

q = q->next;

if (q == NULL)

{

cout << "There are less than ";

cout << pos << " elements." << endl;

return;

}

}

tmp = new(struct node);

tmp->country = country;

tmp->population = population;

tmp->area = area;

tmp->revenue = revenue;

if (q->next == NULL)

{

q->next = tmp;

tmp->next = NULL;

tmp->prev = q;

}

else

{

tmp->next = q->next;

tmp->next->prev = tmp;

q->next = tmp;

tmp->prev = q;

}

cout << "Element Inserted" << endl;

}

void double\_llist::delete\_element(string country, int population, int area, int revenue)

{

struct node \*tmp, \*q;

/\*first element deletion\*/

if (start->country == country)

{

tmp = start;

start = start->next;

start->prev = NULL;

cout << "Element Deleted" << endl;

free(tmp);

return;

}

q = start;

while (q->next->next != NULL)

{

/\*element deleted in between\*/

if (q->next->country == country)

{

tmp = q->next;

q->next = tmp->next;

tmp->next->prev = q;

cout << "Element Deleted" << endl;

free(tmp);

return;

}

q = q->next;

}

/\*last element deleted\*/

if (q->next->country == country)

{

tmp = q->next;

free(tmp);

q->next = NULL;

cout << "Element Deleted" << endl;

return;

}

cout << "Element " << country << " not found" << endl;

}

void double\_llist::display\_dlist()

{

struct node \*q;

if (start == NULL)

{

cout << "List empty,nothing to display" << endl;

return;

}

q = start;

cout << "The Doubly Link List is :" << endl;

while (q != NULL)

{

cout << "\n\nCountry: " << q->country << "\n Population: " << q->population << "\n Area: " << q->area << "\n Revenue: " << q->revenue;

q = q->next;

}

}

void double\_llist::count()

{

struct node \*q = start;

int cnt = 0;

while (q != NULL)

{

q = q->next;

cnt++;

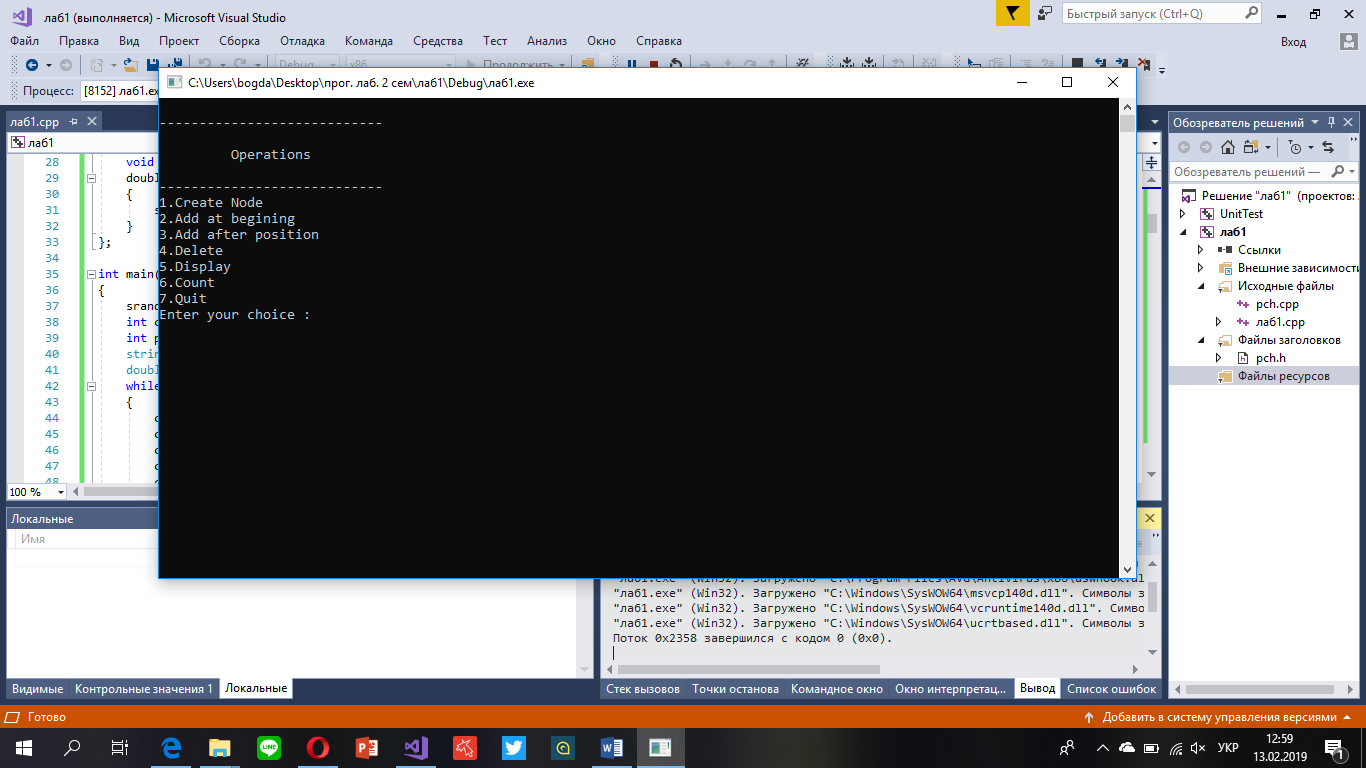
}

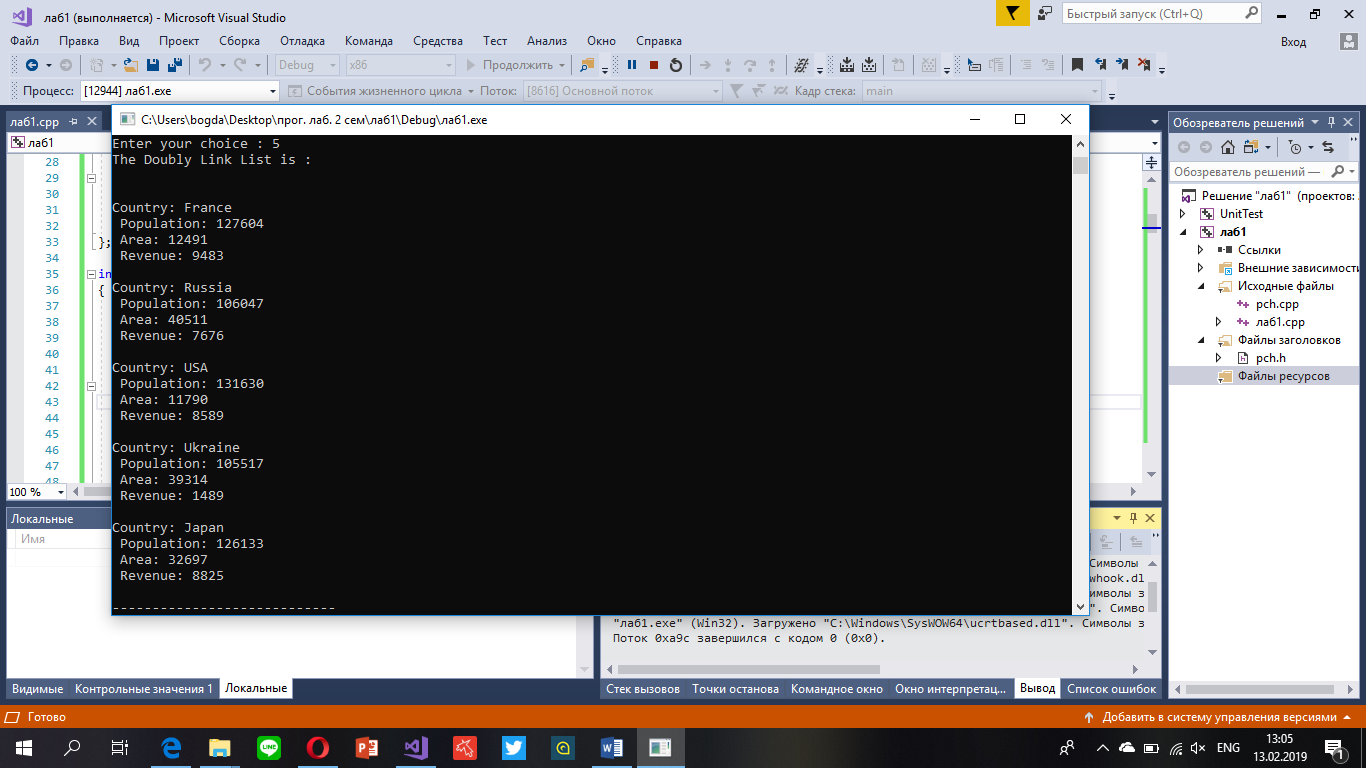
cout << "Number of elements are: " << cnt << endl;

}

1. **ВАРІАНТИ ВИКОРИСТАННЯ**

На рисунку № 3 зоображено результат роботи програми.





Висновок: Я придбав практичні навички щодо розроблення програм із використанням структурованих типів даних.