СТЕКОВІ ОБ’ЄКТИ. КОНСТРУКТОР, ДЕСТРУКТОР, ВІДОБРАЖЕННЯ, ПЕРЕДАЧА

Лабораторна робота №1

Мета:

* навчитися створювати об’єкти
* отримати розуміння створення об’єкта на стеку, а також передачу об’єкта по значенню.

1. ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

Варіант 10. Створити клас даних <Ємність> та клас відображення даних <View>. Об’єкт відображення конструюється на стеку функції main() об’єктом даних, що заздалегідь створений на стеку. Передавати <Ємність> як значення. <Ємність> має всі публічні поля та методи. <View> лише виконує відображення даних у форматі <Назва поля>=<Значення>.

1. РОЗРОБКА ПРОГРАМИ

Для реалізації програми було створено:

* клас <Ємність> з полями одиниці вимірювання, об’єм;
* клас <View> з методом відображення даних;

1. Засоби

У розробленій програмі використані наступні засоби ООП:

* розділення програми на ієрархію класів (інкапсуляція);

1. Ієрархія та структура класів

На рис.2.2 наведена структура розроблених класів

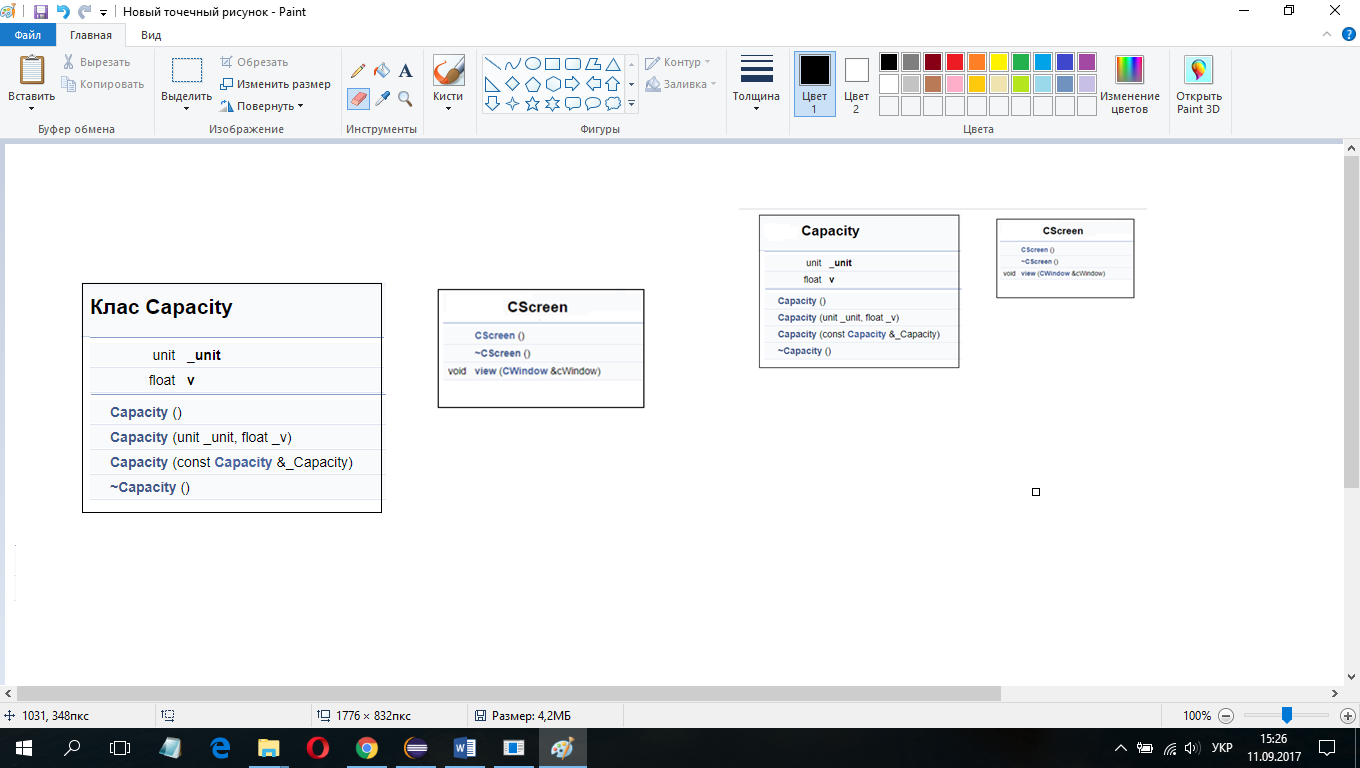


Рисунок 2.2 – Ієрархія класів

1. Опис програми

На рис.2.3 наведена структура розробленого проекту

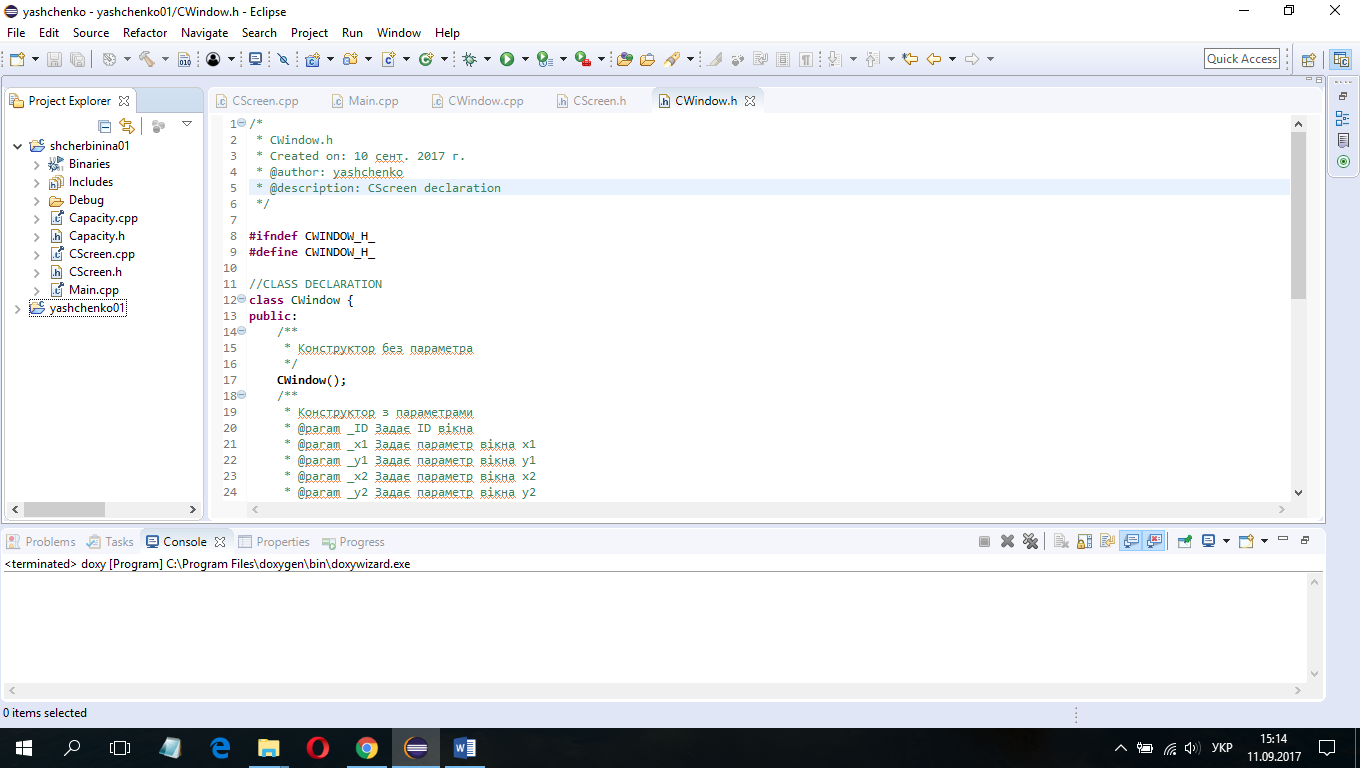


Рисунок 2.3 – Структура проекту

Призначення спроектованих класів наведено на рис.2.4

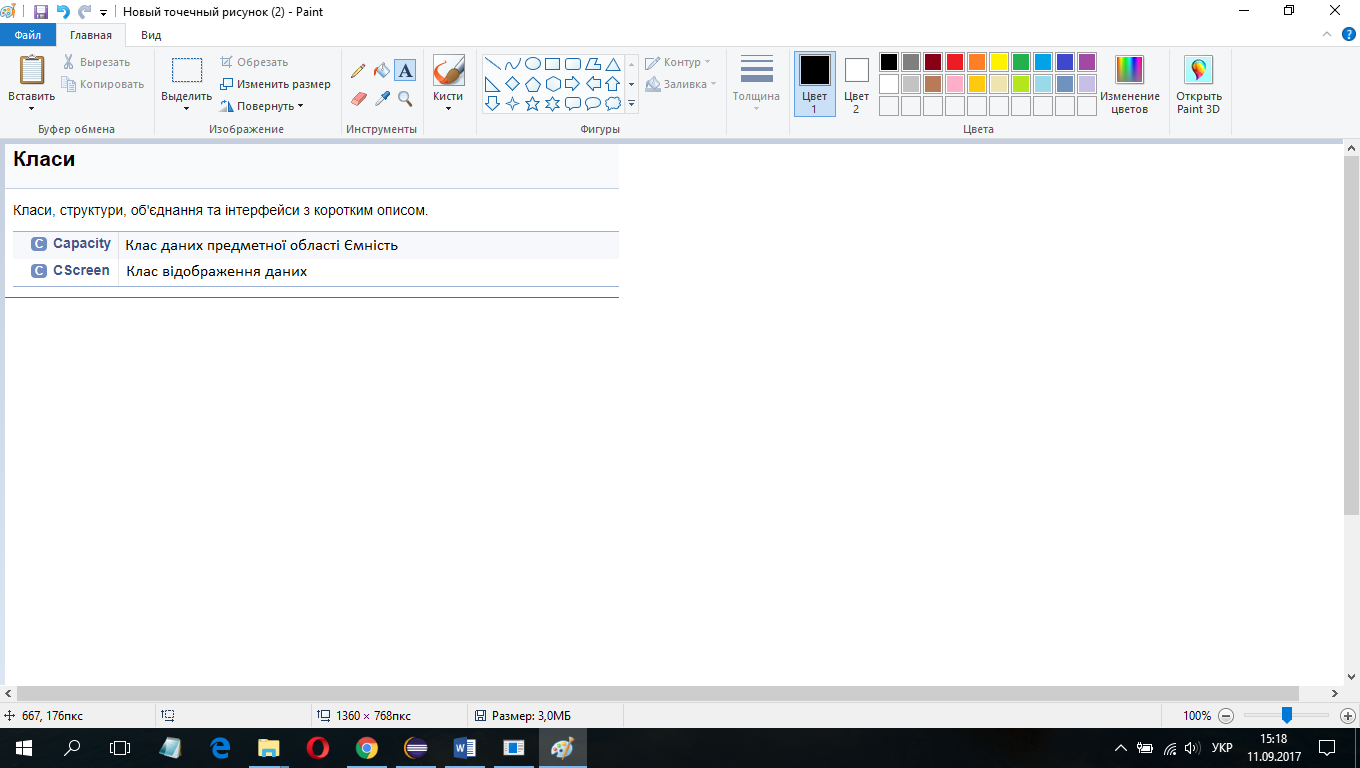


Рисунок 2.4 – Призначення класів

1. Важливі фрагменти програми
2. Файл Capacity.h

/\*

\* Capacity.h

\* Created on: 10 сент. 2017 г.

\* Author: shcherbinina

\* Description: Capacity declaration

\*/

**#ifndef** CAPACITY\_H\_

**#define** CAPACITY\_H\_

**#include** <Windows.h>

**enum** unit { *F*, *mkF*, *nF*, *pF* };

//CLASS DECLARATION

**class** Capacity {

**public**:

/\*\*

\* Конструктор без параметра

\*/

**Capacity**();

/\*\*

\* Конструктор з параметрами

\* @param \_unit Задає одиницю вимірювання

\* @param \_v Задає об'єм

\*/

**Capacity**(unit \_unit, **float** \_v);

/\*\*

\* Конструктор копіювання

\* @param \_Capacity Вихідний об'єкт із даними для копіювання

\*/

**Capacity**(**const** Capacity& \_Capacity);

/\*\*

\* Деструктор

\*/

**~Capacity**();

**public**:

//одиниця вимірювання

unit \_unit;

//об'єм

**float** v;

};

**#endif** /\* CAPACITY\_H\_ \*/

1. Файл Capacity.cpp

/\*

\* Capacity.cpp

\* Created on: 10 сент. 2017 г.

\* Author: shcherbinina

\* Description: Capacity implementation

\*/

**#include** "Capacity.h"

**#include** <Windows.h>

//Конструктор без параметра

**Capacity::Capacity**() {

\_unit = *F*;

v = 0;

}

//Конструктор копіювання

**Capacity::Capacity**(**const** Capacity& \_Capacity){

\_unit = \_Capacity.\_unit;

v = \_Capacity.v;

}

//Деструктор

**Capacity::~Capacity**() {

OutputDebugString("Destructor is called\n");

}

//Конструктор із параметрами

**Capacity::Capacity**(unit unitc, **float** \_v){

\_unit = unitc;

v = \_v;

}

1. Файл CScreen.h

/\*

\* CScreen.h

\* Created on: 10 сент. 2017 г.

\* Author: shcherbinina

\* Description: CScreen declaration

\*/

**#ifndef** CSCREEN\_H\_

**#define** CSCREEN\_H\_

**#include** <Capacity.h>

//CLASS DECLARATION

**class** CScreen {

**public**:

/\*\*

\* Конструктор без параметра

\*/

**CScreen**();

/\*\*

\* Деструктор

\*/

**~CScreen**();

/\*\*

\* Функція відображення даних об'єкта класу Capacity

\*/

**void** **view**(Capacity& cCapacity);

};

**#endif** /\* CSCREEN\_H\_ \*/

1. Файл CScreen.cpp

/\*

\* CScreen.cpp

\* Created on: 10 сент. 2017 г.

\* Author: shcherbinina

\* Description: CScreen implementation

\*/

**#include** <iostream>

**#include** <stdio.h>

**#include** <Windows.h>

**#include** "CScreen.h"

//конструктор без параметра

CScreen::CScreen() {

// **TODO** Auto-generated constructor stub

}

//Деструктор

CScreen::~CScreen() {

// **TODO** Auto-generated destructor stub

}

//Функція відображення даних об'єкта класу Capacity

**void** CScreen::view(Capacity& cCapacity) {

**switch**(cCapacity.\_unit){

**case** F:

std::cout << "unit = F" << std::endl;

**break**;

**case** mkF:

std::cout << "unit = mkF" << std::endl;

**break**;

**case** nF:

std::cout << "unit = nF" << std::endl;

**break**;

**case** pF:

std::cout << "unit = pF" << std::endl;

**break**;

}

std::cout << "V = " << cCapacity.v << std::endl;

}

1. Файл Main.cpp

/\*

\* Main.cpp

\* Created on: 10 сент. 2017 г.

\* Author: shcherbinina

\* Description: point of entry

\*/

**#include** <CScreen.h>

**#include** <iostream>

**#include** <stdio.h>

**#include** <Windows.h>

//point of entry

**int** **main**() {

Capacity cCapacity1(*F*, 200);

CScreen view;

view.view(cCapacity1);

}

1. РЕЗУЛЬТАТИ РОБОТИ

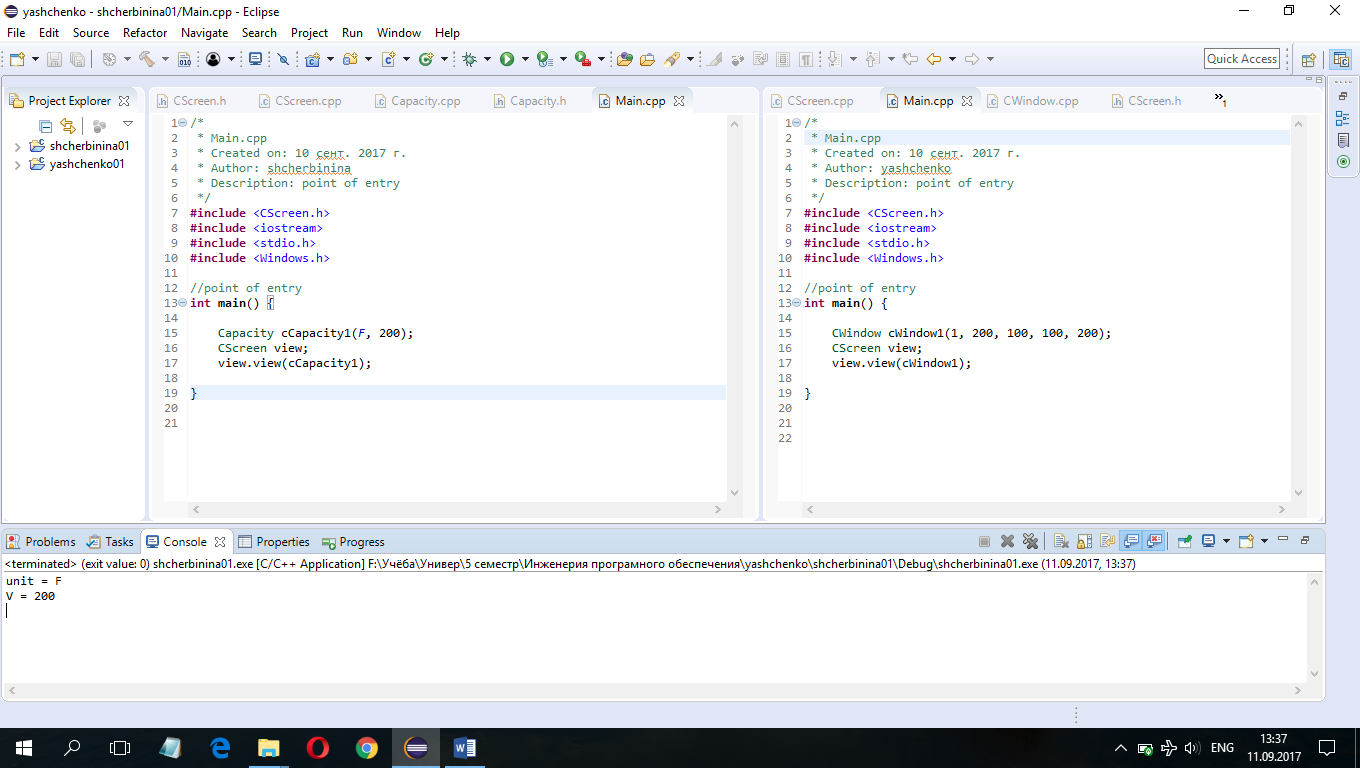


Рисунок 3.1 – Приклад роботи програми

ВИСНОВКИ

В розробленій програмі я навчилася створювати об’єкти, отримала розуміння створення об’єктів на стеку, а також передачі об’єкта по значенню.