ПРАВА ДОСТУПУ, CONST, ПОКАЖЧИКИ, ПОСИЛАННЯ

Лабораторна робота №2

Мета:

* отримати навики при передаванні об'єктів із застосуванням прав доступу та const-модифікаторів.

1. ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

Варіант 10. Розподілити в класі Capacity права доступу private, public. Реалізувати клас GraphScreen основна задача якого полягає у більш багатому відображенню даних Capacity із застосуванням псевдографіки для наочного відображення пов'язаного об'єкта. GraphScreen повинен містити поля згідно опису в індивідуальному завданні та наступні методи:

- SetDataSource() - для зміни об'єкта-джерела даних.

- PrintData() - виводитиме інформацію про отриманий об'єкт у якості аргументу. Оновити Screen для збереження функціональності цього класу при роботі з оновленим Capacity.

1. РОЗРОБКА ПРОГРАМИ

Для реалізації програми було створено клас GraphScreen та оновлено існуючі класи та методи, згідно індивідуального завдання.

1. Засоби ООП

У розробленій програмі використані наступні засоби ООП:

* розділення програми на ієрархію класів (інкапсуляція);

1. Ієрархія та структура класів

На рис.2.2 наведена ієрархія розроблених класів

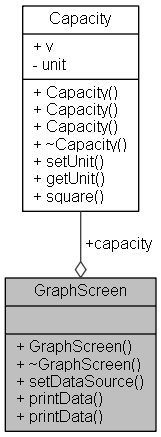
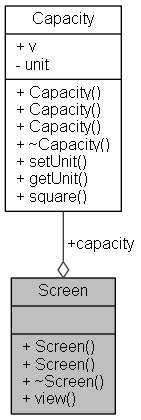
 

Рисунок 2.2 – Ієрархія класів

1. Опис програми

На рис.2.3 наведена структура розробленого проекту

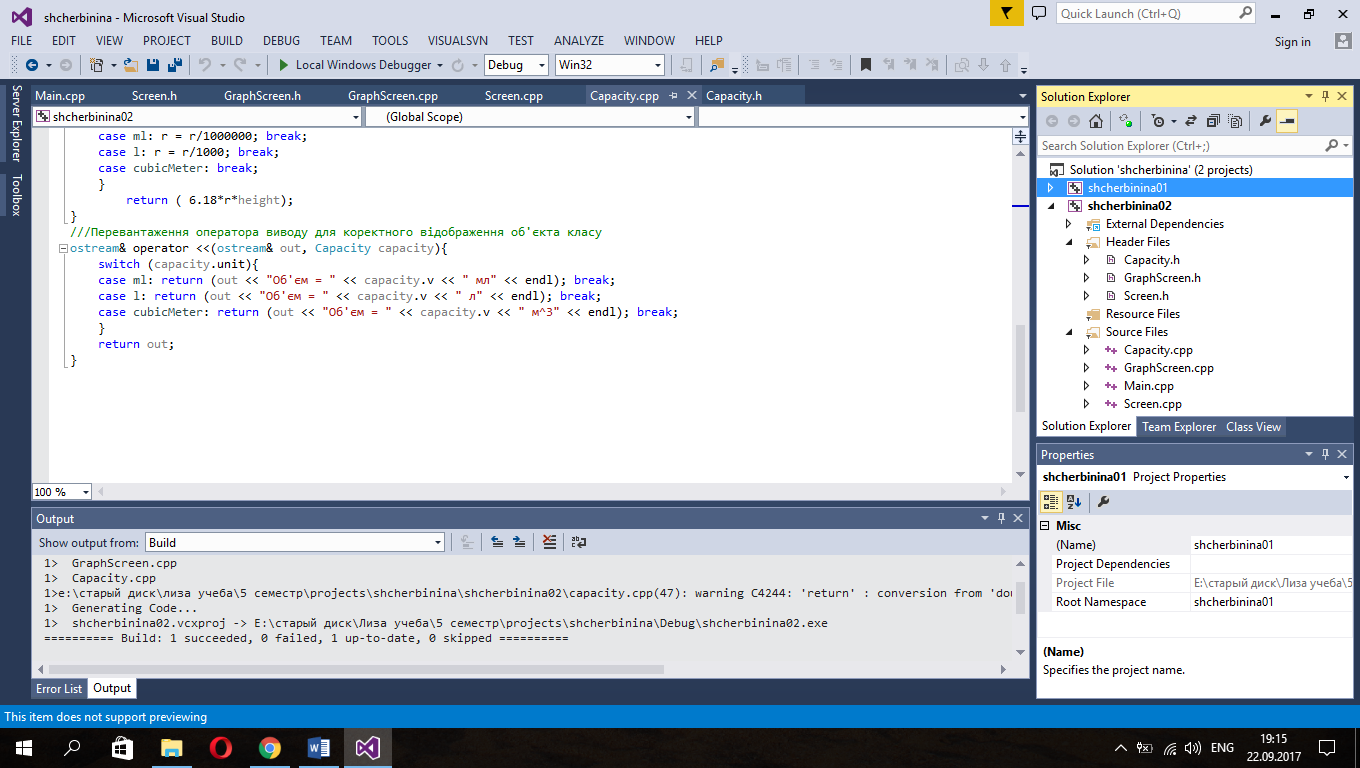


Рисунок 2.3 – Структура проекту

Призначення спроектованих класів наведено на рис.2.4

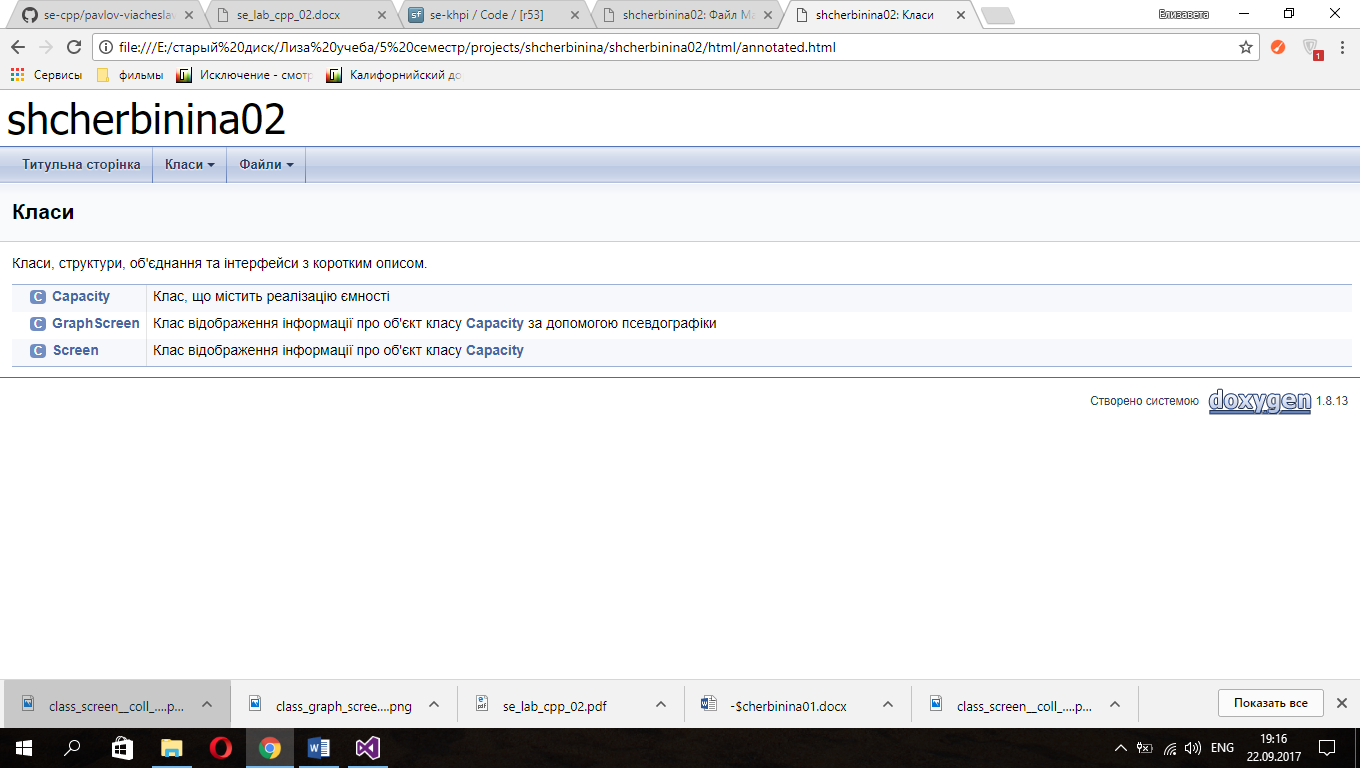


Рисунок 2.4 – Призначення класів

1. Важливі фрагменти програми

Функція відображення об’єкта за допомогою псевдографіки:

///Функція відображення даних

void GraphScreen::printData(const Capacity& capacity){

Screen view(capacity);

view.view();

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

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

cout << " | | | |\n";

cout << " | | | |\n";

cout << " | ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^|^^|\n";

cout << " |^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^| |\n";

cout << " | | | |\n";

cout << " | | | |\n";

cout << " | | | |\n";

cout << " | | | |\n";

cout << " | | | |\n";

cout << " | | | |\n";

cout << " | |\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ |\_ |\n";

cout << " |/ | /\n";

cout << " |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|/\n";

}

Функція підрахунку площі тари циліндричної форми:

///Функція для знаходження площі циліндричної тари

const float Capacity::square(float height){

double r = 0;

r = sqrt(v/(3.14\*height));

switch (unit){

case ml: r = r/1000000; break;

case l: r = r/1000; break;

case cubicMeter: break;

}

return ( 6.18\*r\*height);

}

Функція main():

/\*\*

\* Точка входу в програму

\*/

int main() {

setlocale(LC\_ALL, "Russian");

Capacity capacity(l, 100);

GraphScreen view;

view.setDataSource(&capacity);

view.printData();

system("pause");

return 0;

}

1. РЕЗУЛЬТАТИ РОБОТИ

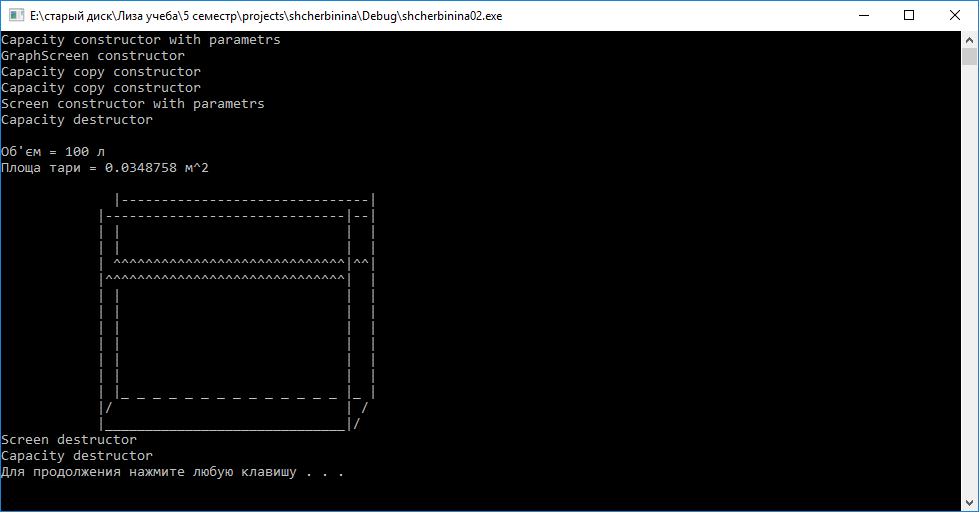


Рисунок 3.1 – Приклад роботи програми

ВИСНОВКИ

В розробленій програмі я отримала навички створення програми із застосуванням прав доступу та соnst-модифікаторів, а також використання покажчиків та посилань на об’єкти.