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

Лабораторна робота №1

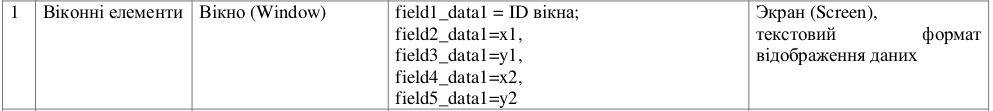
Мета:

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

1 ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

Створити клас даних Window та клас відображення даних — Screen. Обєкт відображення конструюється на стеку функції main() обєктом даних, що заздалегідь створений на стеку. Передавати Window як значення.

Window має всі публічні поля та методи. Лише виконує відображення даних у форматі назва поля=значення; всі його методи та атрибути публічні.



2 РОЗРОБКА ПРОГРАМИ

Для реалізації програми було розроблено два класи: Window та Screen.

2.1 Ієрархія та структура класів

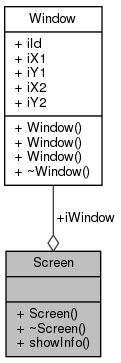
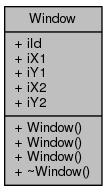


Рисунок 2.1 — Ієрархія класів

2.2 Опис програми

На рис.2.2 наведена структура розробленого проекту

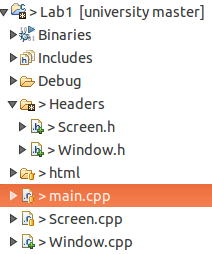


Рисунок 2.2 — Структура проекту

Призначення спроектованих класів наведено на рис. 2.3.



Рисунок 2.3 — Призначення класів, створене за допомогою Javadoc

2.3 Фрагменти програми

2.3.1 Файл Window.h

/\*\*

\* @file Window.h

\* Declaration of class Window.

\* @author Яшенко Олександр

\* @version 0.0.1

\* @date 2017.09.15

\*/

**#ifndef** WINDOW\_H\_

**#define** WINDOW\_H\_

/\*\*

\* Storing information about window

\*/

**class** Window {

**public**:

/\*\*

\* Constructor without parameters

\*/

**Window**(**void**);

/\*\*

\* Constructor

\* @param aId sets Window::iId

\* @param aX1 sets Window::iX1

\* @param aY1 sets Window::iY1

\* @param aX2 sets Window::iX2

\* @param aY2 sets Window::iY2

\*/

**Window**(**int** aId, **int** aX1, **int** aY1, **int** aX2, **int** aY2);

/\*\*

\* Copying contrustor

\* @param window object with data to copy

\*/

**Window**(**const** Window& window);

/\*\*

\* Empty destructor

\*/

**~Window**();

**int** iId; ///< Window identification number

**int** iX1; ///< X1 coordinate

**int** iY1; ///< Y1 coordinate

**int** iX2; ///< X2 coordinate

**int** iY2; ///< Y2 coordinate

};

**#endif** // WINDOW\_H\_

2.3.2 Файл Window.cpp

/\*\*

\* @file Window.cpp

\* Implementation of Window class.

\* @author Яшенко Олександр

\* @version 0.0.1

\* @date 2017.09.15

\*/

**#include** "Window.h"

**Window::Window**() :

iId(0), iX1(0), iY1(0), iX2(0), iY2(0) {

}

**Window::Window**(**int** aId, **int** aX1, **int** aY1, **int** aX2, **int** aY2) :

iId(aId), iX1(aX1), iY1(aY1), iX2(aX2), iY2(aY2) {

}

**Window::Window**(**const** Window& window) :

iId(window.iId), iX1(window.iX1), iY1(window.iY1), iX2(window.iX2), iY2(

window.iY2) {

}

**Window::~Window**() {

}

2.3.3 Файл Screen.h

/\*\*

\* @file Screen.h

\* Declaration of class Screen.

\* @author Яшенко Олександр

\* @version 0.0.1

\* @date 2017.09.15

\*/

**#ifndef** SCREEN\_H\_

**#define** SCREEN\_H\_

**#include** "Window.h"

/\*\*

\* Shows information about window

\*/

**class** Screen {

**public**:

/\*\*

\* Constructor

\* @param window sets Screen::iWindow

\*/

**Screen**(Window window);

/\*\*

\* Empty destructor

\*/

**~Screen**();

Window iWindow; ///< Information about window

/\*\*

\* Shows data from Screen::iWindow

\*/

**void** **showInfo**();

};

**#endif** // SCREEN\_H\_

2.3.4 Файл Screen.cpp

/\*\*

\* @file Screen.cpp

\* Implementation of class Screen

\* @author Яшенко Олександр

\* @version 0.0.1

\* @date 2017.09.15

\*/

**#include** "Screen.h"

**#include** <iostream>

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

**Screen::Screen**(Window window) :

iWindow(window) {

}

**Screen::~Screen**() {

}

**void** **Screen::showInfo**() {

cout << "ID=" << iWindow.iId << **endl**;

cout << "X1=" << iWindow.iX1 << **endl**;

cout << "Y1=" << iWindow.iY1 << **endl**;

cout << "X2=" << iWindow.iX2 << **endl**;

cout << "Y2=" << iWindow.iY2 << **endl**;

}

2.3.5 Файл main.cpp

/\*\*

\* @file main.cpp

\* Implementation of main() function

\* @author Яшенко Олександр

\* @version 0.0.1

\* @date 2017.09.15

\*/

**#include** "Window.h"

**#include** "Screen.h"

/\*\*

\* Entry point.

\* @param argc number of command line parameters

\* @param argv array of command line parameters

\* @return exit code

\*/

**int** **main**(**int** argc, **char**\*\* argv) {

Window window(1, 10, 20, 30, 40);

Screen view2(window);

view2.showInfo();

**return** 0;

}

3 РЕЗУЛЬТАТИ РОБОТИ

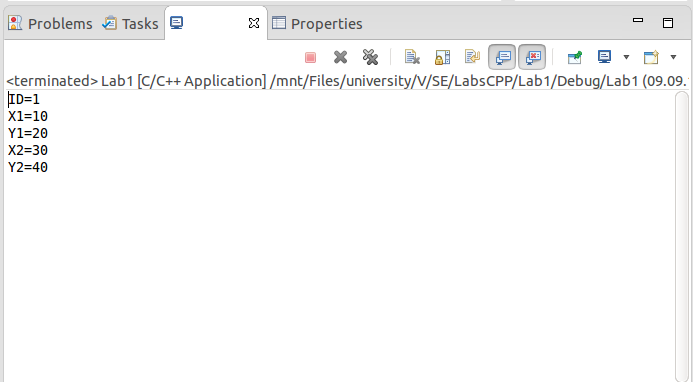


Рисунок 3.1 - Приклад роботи програми

ВИСНОВКИ

В розробленій програмі реалізовано класи Window та Screen. Window — клас для збереження інформації про вікна, Screen — для відображення інформації про вікна.