Лабораторна робота №2

ПРАВА ДОСТУПУ, const, ПОКАЗЧИКИ, ПОСИЛАННЯ.

***Мета роботи***: Отримати навики при передаванні об’єктів у класи із застосуванням прав доступу та const-методів.

**1 ІНДИВІДУАЛЬНЕ ЗАВДАННЯ**

Розподілити в <Data1> права доступу private, public.

Реалізувати клас <View2> основна задача котрого полягає у більш

багатому відображенню даних <Data1> із застосуванням псевдографіки для наочного відображення пов’язаного об’єкта.

<View2> повинен містити поля згідно опису в індивідуальному завданні

та наступні обов’язкові методи:

- SetDataSource() – для зміни об’єкта-джерела даних. Сигнатура:

void SetDataSource(const CMyData\* aData);

- PrintData() – виводитиме на екран інформацію про отриманий об’єкт у

якості аргументу. Сигнатура:

void <View2>::PrintData(const CMyData& aData);

Оновити <View1> для збереження функціональності цього класу при

роботі із оновленим <Data1>.

Згідно індивідуального завдання реалізувати нову програму, базуючись

на класах роботи No1, та вибравши своє завдання за варіантами.

Завдання вибирається згідно із номером у журналі за формулою

TaskNumber = (No mod 10) +1;

**2 ОПИС ПРОГРАМИ**

**2.1 Ієрархія та структура класів**

**class Data1 –** клас, який містить атрибути **data\_field1 data\_field2.**

**class View –** клас, який містить метод, що виводить значення полів **data\_field1 data\_field2**.

**class View2** – клас, який мітить методи info()

**2.2 Опис програми**

На рисунку 2.1 наведена структура розробленої програми

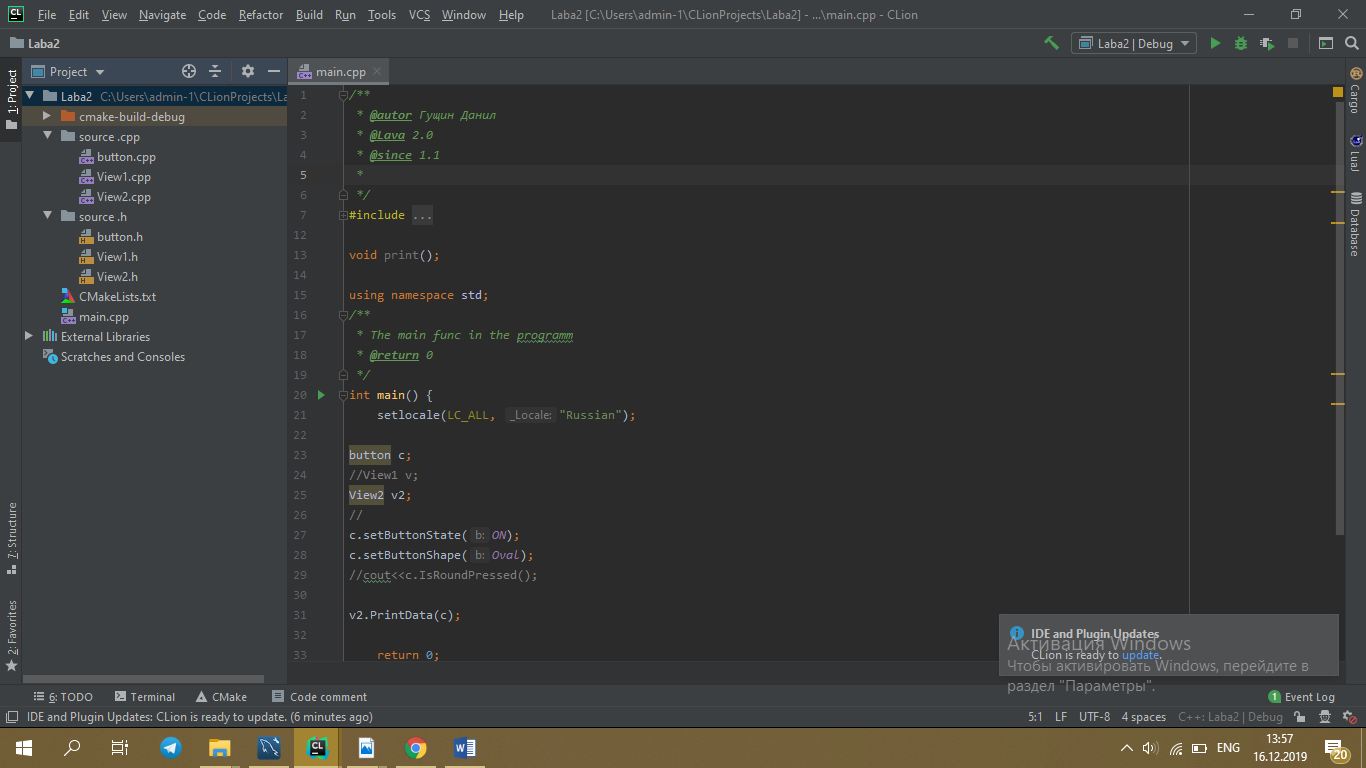


Рис 2.1 – структура програми

**2.3 Важливі фрагменти програми**

2.3.1 Файл*main.cpp*

/\*\*  
 \* @autor Гущин Данил  
 \* @Lava 2.0  
 \* @since 1.1  
 \*  
 \*/  
#include <iostream>  
#include "source .cpp/button.cpp"  
#include "source .h/button.h"  
#include "source .cpp/View1.cpp"  
#include "source .cpp/View2.cpp"  
  
void print();  
  
using namespace std;  
/\*\*  
 \* The main func in the programm  
 \* @return 0  
 \*/  
int main() {  
 setlocale(LC\_ALL, "Russian");  
  
button c;  
//View1 v;  
View2 v2;  
//  
c.setButtonState(*ON*);  
c.setButtonShape(*Oval*);  
//cout<<c.IsRoundPressed();  
  
v2.PrintData(c);  
  
 return 0;  
}

2.3.1 Файл*Data1.cpp*

//  
// Created by admin-1 on 06.09.2019.  
//  
#pragma once  
#include "../source .h/button.h"  
#include <iostream>  
 /\*\*  
 \* Setter  
 \* @param b Value for initialization field bt  
 \*/  
 void button::setButtonState(ButtonState b) {  
 bt = b;  
 }  
/\*\*  
 \* Getter  
 \* @return bt  
 \*/  
ButtonState button::getButtonState() const {  
 return this->bt;  
}  
  
  
/\*\*  
 \*  
 \* @return info when button is round pressed true  
 \* or false if button is don't pressed  
 \*/  
bool button::IsRoundPressed() {  
if(getButtonShape()==0)  
{  
 return true;  
}else  
 {  
 return false;  
 }  
}  
/\*\*  
 \* Setter  
 \* @param b Value for initialization field bs  
 \*/  
void button::setButtonShape(ButtonShape b) {  
bs=b;  
}  
/\*\*  
 \* Getter  
 \* @return bs  
 \*/  
ButtonShape button::getButtonShape() const {  
 return this->bs;  
}

2.3.1 Файл*Data1.h*

/\*\*  
 \* @autor Гущин Данил  
 \* @Lava 2.0  
 \* @since 1.1  
 \*  
 \*/  
#pragma once  
#include <string>  
  
using namespace std;  
/\*\*  
 \* Created enum for class <Data>  
 \* which description state of the button  
 \*/  
enum ButtonState  
{  
 *ON*,  
 *OFF*};  
/\*\*  
 \* Created enum for class <Data>  
 \* which description shape of the button  
 \*/  
enum ButtonShape  
 {  
*Oval*,  
*Rectangular* };  
/\*\*  
 \*Class button(<Data>)  
 \* the main class data  
 \* which have fields shape and state of button  
 \* and his setters and getters  
 \*/  
 class button{  
 private:  
 ButtonState bt;  
 ButtonShape bs;  
  
 public:  
  
/\*\*  
 \* Setter  
 \* @param b Value for initialization field bt  
 \*/  
 void setButtonState(ButtonState b);  
 /\*\*  
 \* Setter  
 \* @param b Value for initialization field bt  
 \*/  
 void setButtonShape(ButtonShape b);  
 /\*\*  
 \* Getter  
 \* @return bs  
 \*/  
 ButtonShape getButtonShape() const;  
 /\*\*  
 \* Getter  
 \* @return bt  
 \*/  
 ButtonState getButtonState() const;  
/\*\*  
 \*  
 \* @return info when button is round pressed true  
 \* or false if button is don't pressed  
 \*/  
 bool IsRoundPressed();  
};

2.3.2 Файл*View.cpp*

#include "../source .h/View1.h"  
#include "../source .h/button.h"  
#include "iostream"  
using namespace std;  
/\*\*  
 \* This function give info about object b  
 \* of class button  
 \* @param b  
 \*/  
void View1::INFO(button b) {  
 cout<<"State of button is ";  
 cout <<b.getButtonState()<<endl;  
  
 cout<<"Shape of button is ";  
 cout <<b.getButtonShape()<<endl;  
  
  
}

2.3.3 Файл*View.h*

#pragma once  
/\*\*  
 \* Class View1 which show info about  
 \* data class(<Data>)  
 \*/  
class View1 {  
public:  
/\*\*  
 \* This function give info about object b  
 \* of class button  
 \* @param b  
 \*/  
 void INFO(button b);  
};

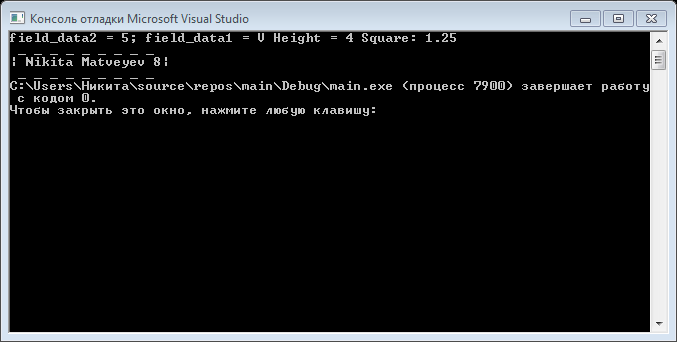
//  
// Created by admin-1 on 07.09.2019.  
//  
#pragma once  
#include "../source .h/View2.h"  
#include <iostream>  
#include "../source .h/button.h"  
#include "../source .cpp/button.cpp"  
/\*\*  
 \* Setter for object b of class button  
 \* which change his Data Source  
 \* @param b  
 \*/  
void View2::setDataSource(const button &b)  
{  
 c=b;  
}  
/\*\*  
\* This function give info about object b  
\* of class button  
\* @param b  
\*/  
void View2::PrintData(const button& b)  
{  
  
 int a=176;  
  
 cout<<"Button State is ";  
cout <<b.getButtonState()<<endl;  
 cout<<"Button Shape is ";  
cout <<b.getButtonShape()<<endl;  
  
/\*\*  
 \* This is an algorithm for description info about object  
 \* of class button in graphical interface (table)  
 \*/  
  
  
 int len = 31;//this size weight of table  
  
  
 /\*\*  
 \* made length for two columns of table  
 \*/  
 int len\_str1 = 25 + strlen("Kit-27B");  
 int len\_str2 = strlen("Gushin") + strlen("Danil") + strlen("Sergeyevich") + 6;  
  
  
  
/\*\*  
 \* the top bord of the table  
 \*/  
 cout << "\n\t\t\t";  
 for (int counter = 0; counter < len; counter++ )  
 cout << "\*";  
/\*\*  
 \* the first column  
 \*/  
 cout << "\n\t\t\t\* Laba #2 ";  
 for (int counter = 0; counter < len - 25; counter++ )  
 cout << " ";  
  
 /\*\*  
 \* The second column  
 \*/  
 cout << "\*\n\t\t\t\* Completed by: st. gr." << "Kit-27B";  
 for (int counter = 0; counter <= (len - len\_str1); counter++ )  
 cout << " ";  
/\*\*  
 \* Third column  
 \*/  
 cout << "\*\n\t\t\t\* " << "Gushin" << " " << "Danil" << " " << "Sergeyevich";  
 for (int counter = 0; counter <= (len - len\_str2); counter++ )  
 cout << " ";  
  
/\*\*  
 \* fourth column  
 \*/  
 cout << "\*\n\t\t\t\* " << "Button State ";  
 cout<<b.getButtonState()<<" ";  
  
 for (int counter = 0; counter <= (len - len\_str2); counter++ )  
 c cout << " ";  
  
/\*\*  
 \* fifth column  
 \*/  
  
 cout << "\*\n\t\t\t\* " << "Button Shape ";  
 cout<<b.getButtonShape()<<" ";  
  
 for (int counter = 0; counter <= (len - len\_str2); counter++ )  
 cout << " ";  
 cout << "\*";  
  
 /\*\*  
 \* down border of the table  
 \*/  
 cout << "\n\t\t\t";  
 for (int counter = 0; counter < len; counter++ )  
 cout << "\*";  
 out << endl;  
  
  
  
 }

2.3.5 Файл*View2.h*

/\*

#include <iostream>  
using namespace std;  
  
/\*\*  
 \* Class View2 which show more detailed info about  
 \* data class(<Data>) with graphical interface  
 \*/  
class View2  
{  
  
public:  
 button c;  
  
/\*\*  
 \* Setter for object b of class button  
 \* which change his Data Source  
 \* @param b  
 \*/  
 void setDataSource(const button& b);  
 /\*\*  
 \* This function give info about object b  
 \* of class button  
 \* @param b  
 \*/  
 void PrintData(const button& b );  
  
};

**3 РЕЗУЛЬТАТИ**

****

**ВИСНОВКИ**

Отримали навики при передаванні об’єктів у класи із застосуванням прав доступу та const-методів. Закріпили навичку з програмування на С++.