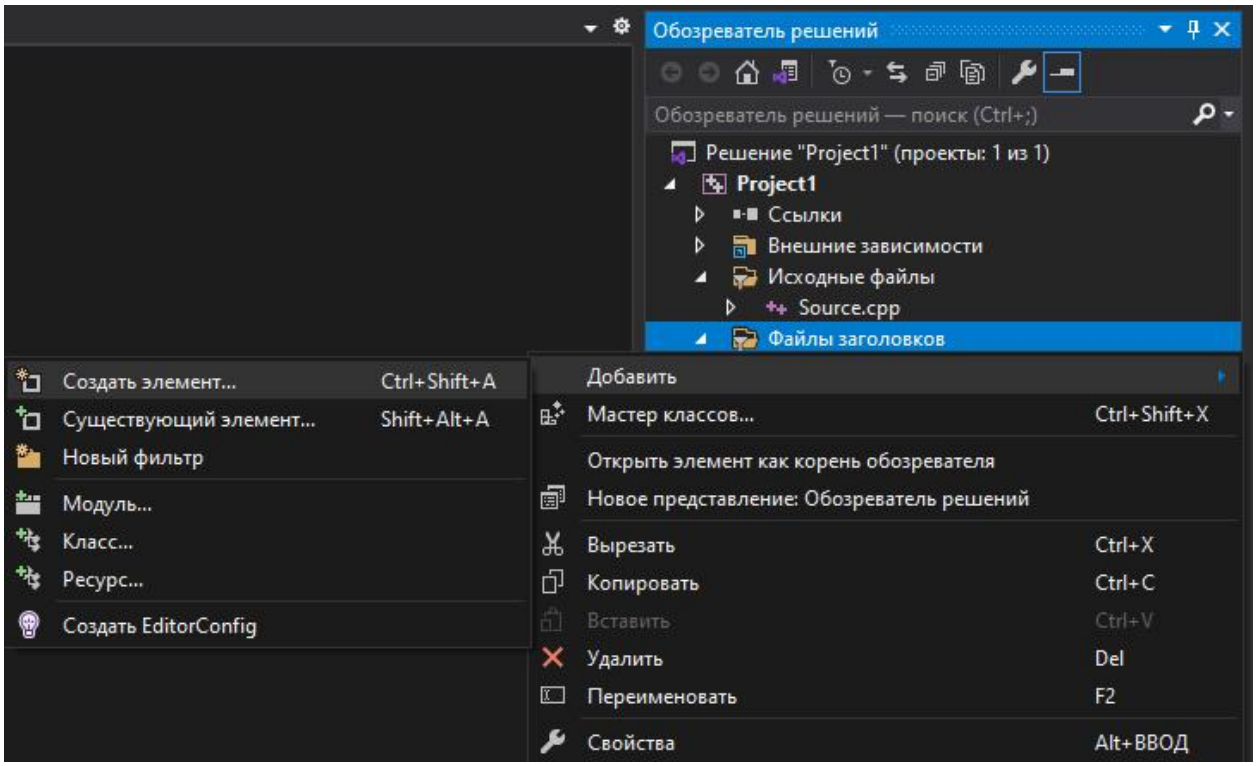
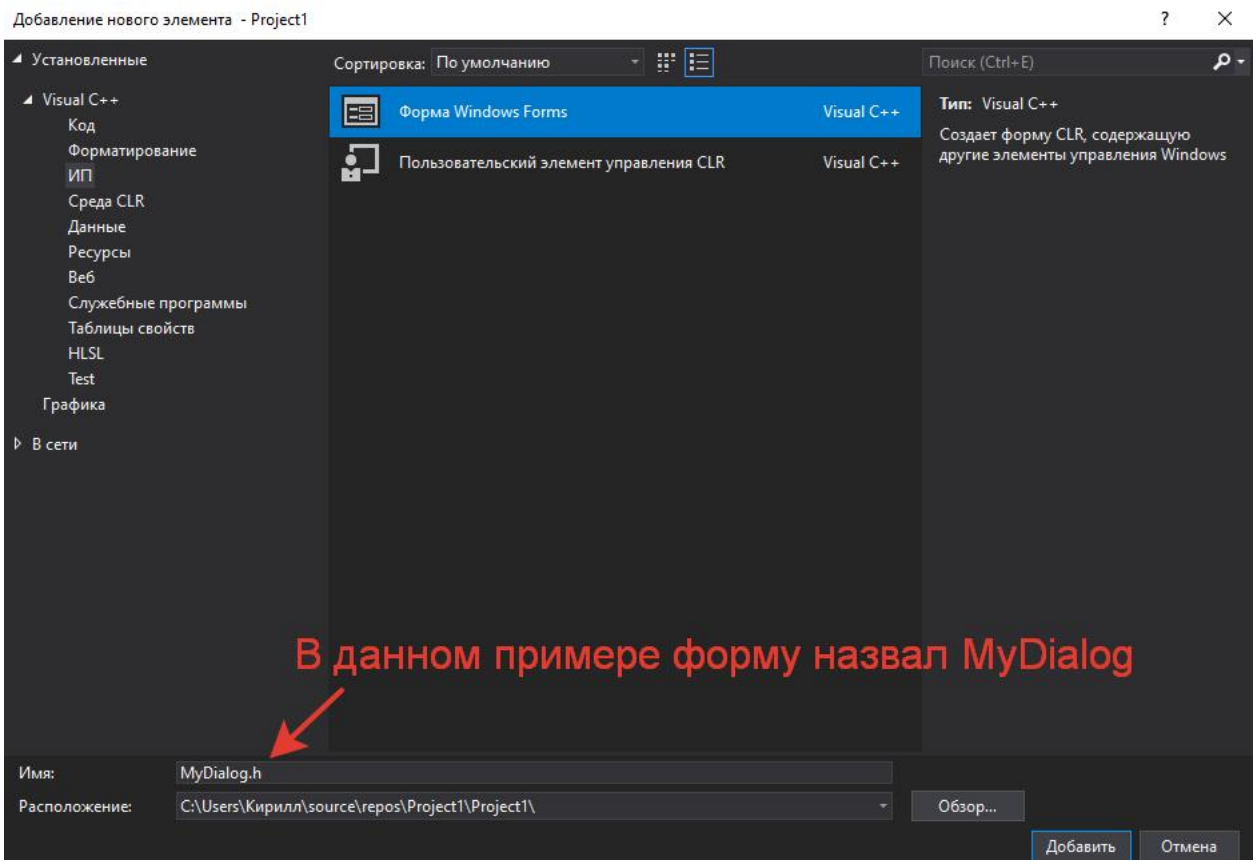
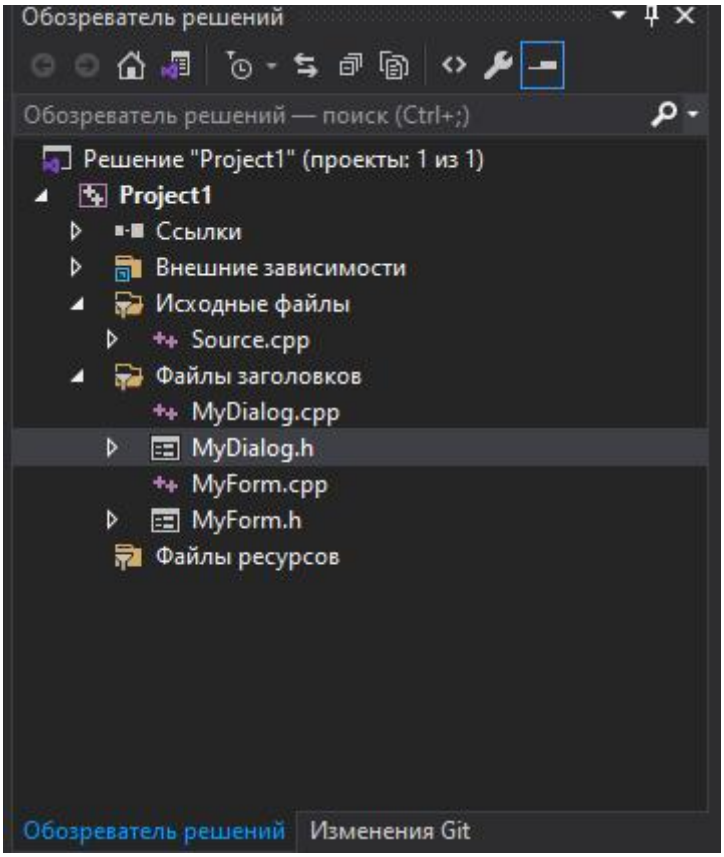
Работа со второй формой (диалоговый режим)

Добавление второй формы в проект выполняется так же, как и добавление первой:



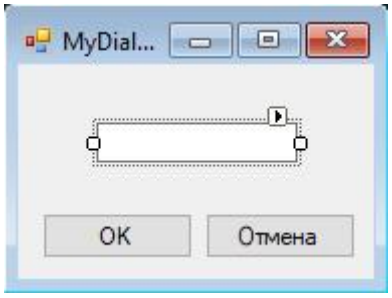


После добавления, вторая форма появится в «Обозревателе решений» рядом с основной:



Если новая Студия после этих действий не может нарисовать новую форму в «Конструкоре» и выдаёт ошибку, помогает перезапуск Студии.

В качестве примера я добавил на вторую форму две кнопки и один TextBox:



Свойства «Name» кнопок я изменил на btnOK и btnCancel, свойство «Name» TextBox’a – на tbMessage. Дальнейший код пишется с учётом этих имён переменных.

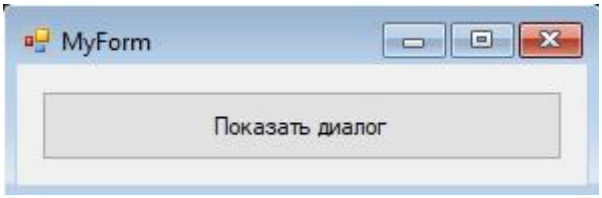
В обработчиках нажатия кнопок нужно написать следующее:

```
private: System::Void btnOK_Click(System::Object^ sender, System::EventArgs^ e) {
    this->DialogResult = Windows::Forms::DialogResult::OK;
}
private: System::Void btnCancel_Click(System::Object^ sender, System::EventArgs^ e) {
    this->DialogResult = Windows::Forms::DialogResult::Cancel;
}
```

`this->DialogResult` – это результат, который должна вернуть функция ShowDialog() этой формы (см. ниже) (как с OpenFileDialog из прошлой лабораторной). «`this->`» можно опустить.

Присвоение значения свойству `DialogResult` сразу закрывает форму, если она была показана с помощью ShowDialog(), при этом функция ShowDialog() в качестве результата вернёт то значение, которое было присвоено свойству `DialogResult`.

Основная форма в примере выглядит так:



В файл с кодом основной формы необходимо подключить файл с кодом второй формы, чтобы класс второй формы был виден в коде первой:

```
#pragma once

#include "MyDialog.h"

namespace Project1 {

    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;

    /// <summary>
    /// Сводка для MyForm
    /// </summary>
    public ref class MyForm : public System::Windows::Forms::Form
    {
    public:
        MyForm(void)
        {
            InitializeComponent();
            //
```

Следующим шагом нужно объявить в классе основной формы переменную-ссылку на вторую форму:

```
            // MyForm
            //
            this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
            this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
            this->ClientSize = System::Drawing::Size(284, 60);
            this->Controls->Add(this->btnDialog);
            this->Name = L"MyForm";
            this->Text = L"MyForm";
            this->ResumeLayout(false);

        }
#pragma endregion
        MyDialog^ dlg;
```

После этого нужно создать экземпляр класса второй формы и сохранить ссылку на него в объявленную переменную. Это можно сделать в конструкторе основной формы:

```
using namespace System;
using namespace System::ComponentModel;
using namespace System::Collections;
using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;

/// <summary>
/// Сводка для MyForm
/// </summary>
public ref class MyForm : public System::Windows::Forms::Form
{
public:
    MyForm(void)
    {
        InitializeComponent();
        //
        //TODO: добавьте код конструктора
        //
        dlg = gcnew MyDialog();
    }
```

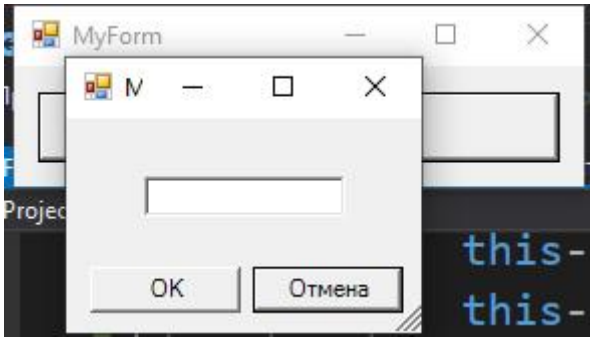
Другой вариант – это можно сделать в обработчике события FormLoad (но первый вариант правильнее).

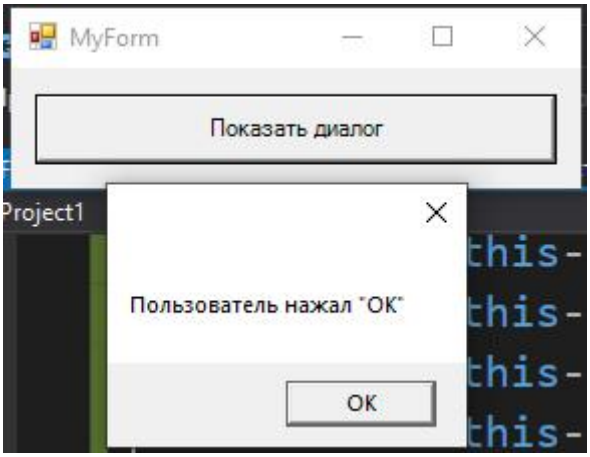
Теперь в основной форме есть переменная “dlg”, содержащая ссылку на экземпляр второй формы. Вся дальнейшая работа со второй формой будет выполняться через эту переменную.

Показ второй формы в режиме диалога теперь осуществляется так же, как показ OpenFileDialog из предыдущей лабораторной. Пример (код - в обработчике кнопки с надписью «Показать диалог» основной формы (см. выше)):

```
private: System::Void btnDialog_Click(System::Object^ sender, System::EventArgs^ e) {
    if (dlg->ShowDialog() == Windows::Forms::DialogResult::OK) {
        MessageBox::Show("Пользователь нажал \"OK\"");
    }
}
```

Результат (при нажатии на «OK»):





При нажатии на «Отмена» ничего не происходит, поскольку в коде написано, чтобы MessageBox::Show срабатывал, только если функция ShowDialog() второй формы вернёт OK в качестве результата.

Обмен данными между формами

Во второй форме был размещён TextBox, названный tbMessage (см. выше). Соответствующая ему переменная появилась в коде второй формы с модификатором **private:**

```
private: System::Windows::Forms::Button^ btnOK;
private: System::Windows::Forms::Button^ btnCancel;
private: System::Windows::Forms::TextBox^ tbMessage;
```

(Этот код сгенерирован автоматически при добавлении кнопок и TextBox'а и смены их имён. Не нужно его писать самостоятельно!)

Поскольку модификатор – private, извне класса второй формы получить доступ к tbMessage не получится (и это правильно!). Однако основной форме может понадобиться узнать, что написано в этом TextBox'е, или изменить эту надпись. Правильный подход – добавить для этих целей в код второй формы

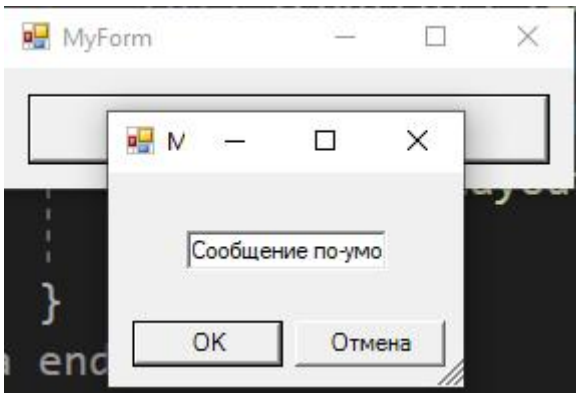
две функции с модификатором **public**, которые будут возвращать и изменять свойство Text TextBox’a “tbMessage”:

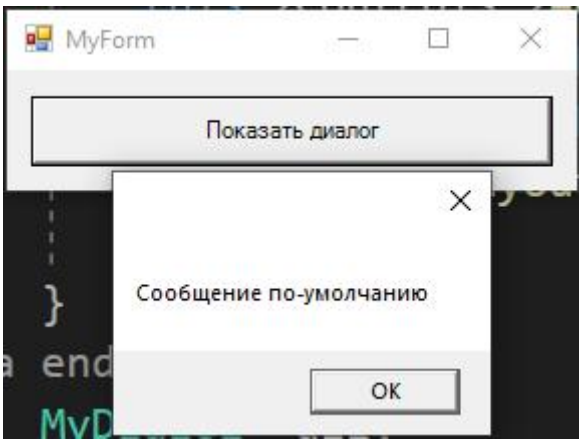
```
            this->Name = L"MyDialog";
            this->Text = L"MyDialog";
            this->ResumeLayout(false);
            this->PerformLayout();

        }
#pragma endregion
    public:
        void SetMessage(String^ msg) {
            tbMessage->Text = msg;
        }

        String^ GetMessage() {
            return tbMessage->Text;
        }
```

Теперь основная форма, пользуясь этими функциями, сможет узнать или изменить то, что написано в свойстве Text TextBox’a tbMessage. Изменим функцию обработки нажатия кнопки «Показать диалог» следующим образом:

```
private: System::Void btnDialog_Click(System::Object^ sender, System::EventArgs^ e) {
    dlg->SetMessage("Сообщение по-умолчанию");
    if (dlg->ShowDialog() == Windows::Forms::DialogResult::OK) {
        MessageBox::Show(dlg->GetMessage());
    }
}
```

Результат:





С помощью написанной функции SetMessage в tbMessage->Text заносится значение «Сообщение по-умолчанию», затем вторая форма показывается в режиме диалога и, если пользователь нажал в ней «OK», выводится сообщение с текущим содержимым tbMessage->Text, получаемым функцией GetMessage.

Обратите внимание, все функции второй формы вызываются через переменную “dlg”, которая содержит ссылку на экземпляр второй формы (см. выше).

Задание на лабораторную работу:

В лабораторной на таблицы («Музыкальные композиции») реализовать

- добавление композиции (в конец),
- вставку композиции,
- редактирование композиции

через показ второй формы в режиме диалога. То есть, если нужно добавить или вставить композицию, появляется вторая форма с пустыми полями (например, TextBox’ами) для ввода всей необходимой информации о композиции. Пользователь заполняет поля, и **если** нажимает «Принять», то данные из этих полей переносятся в добавляемую/вставляемую строку таблицы на основной форме.

Если же нужно редактировать композицию – то появляется та же самая вторая форма в режиме диалога, однако все поля в ней должны изначально быть заполнены редактируемыми значениями (чтобы, например, если нужно

изменить одну цифру в длительности композиции, пользователю не приходилось заново вводить её название и исполнителя).