Частное учреждение образования
«Колледж бизнеса и права»

УТВЕРЖДАЮ
Заведующий методическим кабинетом
_____Е.В.Фалей
«_____»_____2018 г.

Специальность: 2-40 01 01 «Программное обеспечение информационных технологий»	Дисциплина: «Основы алгоритмизации и программирование»

Лабораторная работа № 29
Инструкционно-технологическая карта

Тема: Построение графиков на плоскости и в трёхмерном пространстве

Цель: Научиться строить графики на плоскости и в трёхмерном пространстве с помощью компонентов

Время выполнения: 2 часа

1. Краткие теоретические сведения. Пример выполнения программы

Элемент Chart предназначен для вывода в экранную форму графика (диаграммы). Очень удобно строить этот график по табличным данным, представленным в виде объекта класса DataTable.

В данной лабораторной работе решается следующая задача. Даны сведения об объемах продаж за пять месяцев. Требуется наглядно визуализировать эти данные на графике в виде гистограммы.

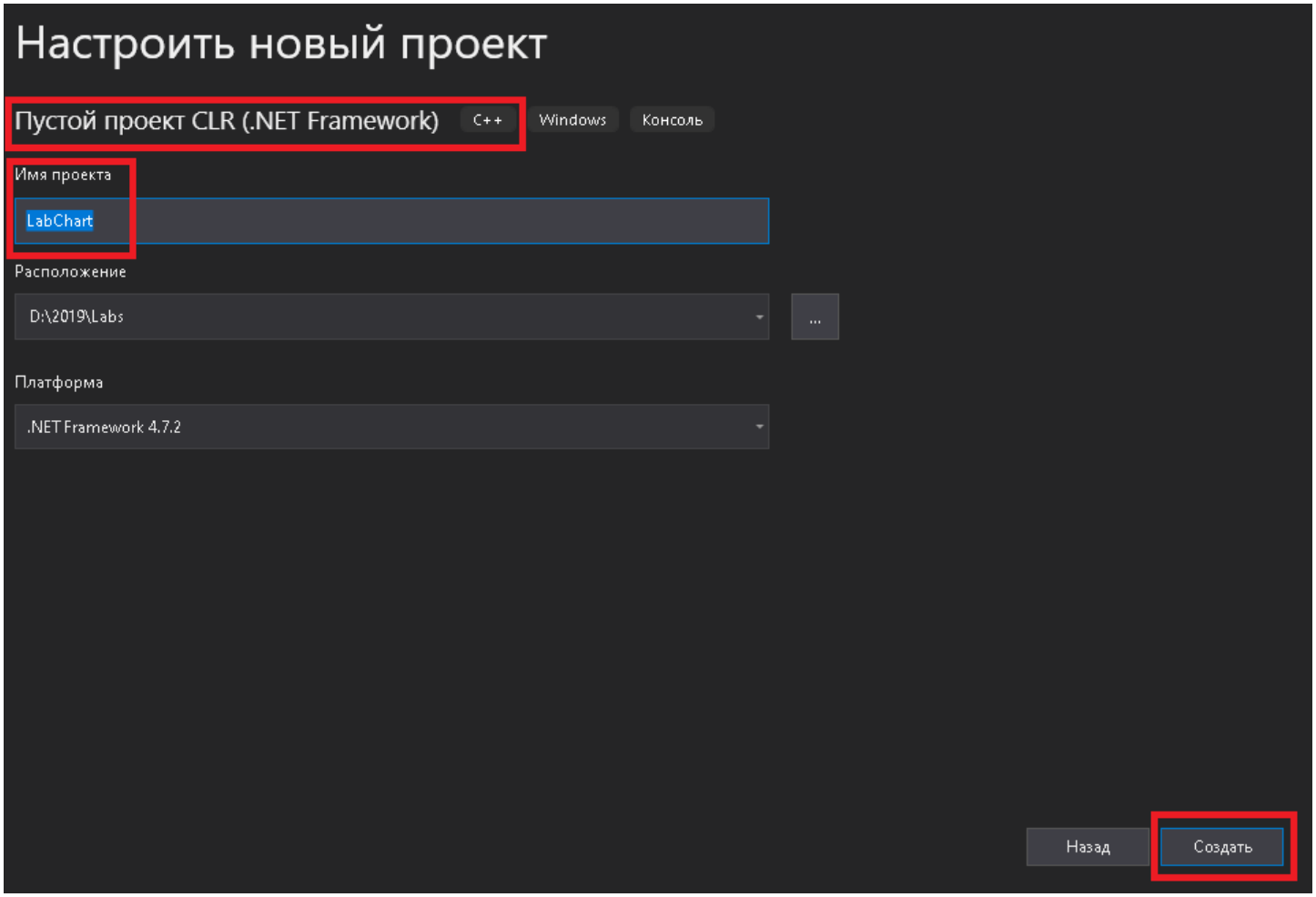
В программном коде при обработке события загрузки формы объявите объектТаблица класса DataTable. Этот объект представляет одну таблицу данных в оперативной памяти. Чтобы визуализировать эту таблицу на экране, используют элемент — сетка данных DataGridView. Объект класса DataTable используют в качестве исходных данных и для сетки данных DataGridView, и для диаграммы Chart. В таблице DataTable определите ее схему, заказывая две колонки «Месяц»и «Объем продаж». А далее заполните таблицу по ее строкам (рядам), используя метод Add. Заполненную пятью строками таблицу укажите в качестве источника данных для элементов Chart и DataGridView. Далее оформите внешний вид диаграмм.

Мы создадим оконное приложение с таблицей и столбчатой диаграммой, данные в которых связаны, синхронно обновляются и редактируются пользователем как в плане изменения данных (их добавления, редактирования и удаления), так и в плане изменения внешнего вида диаграммы (изменения внешнего вида отрисовки столбчатой диаграммы с двумерного на трехмерное).

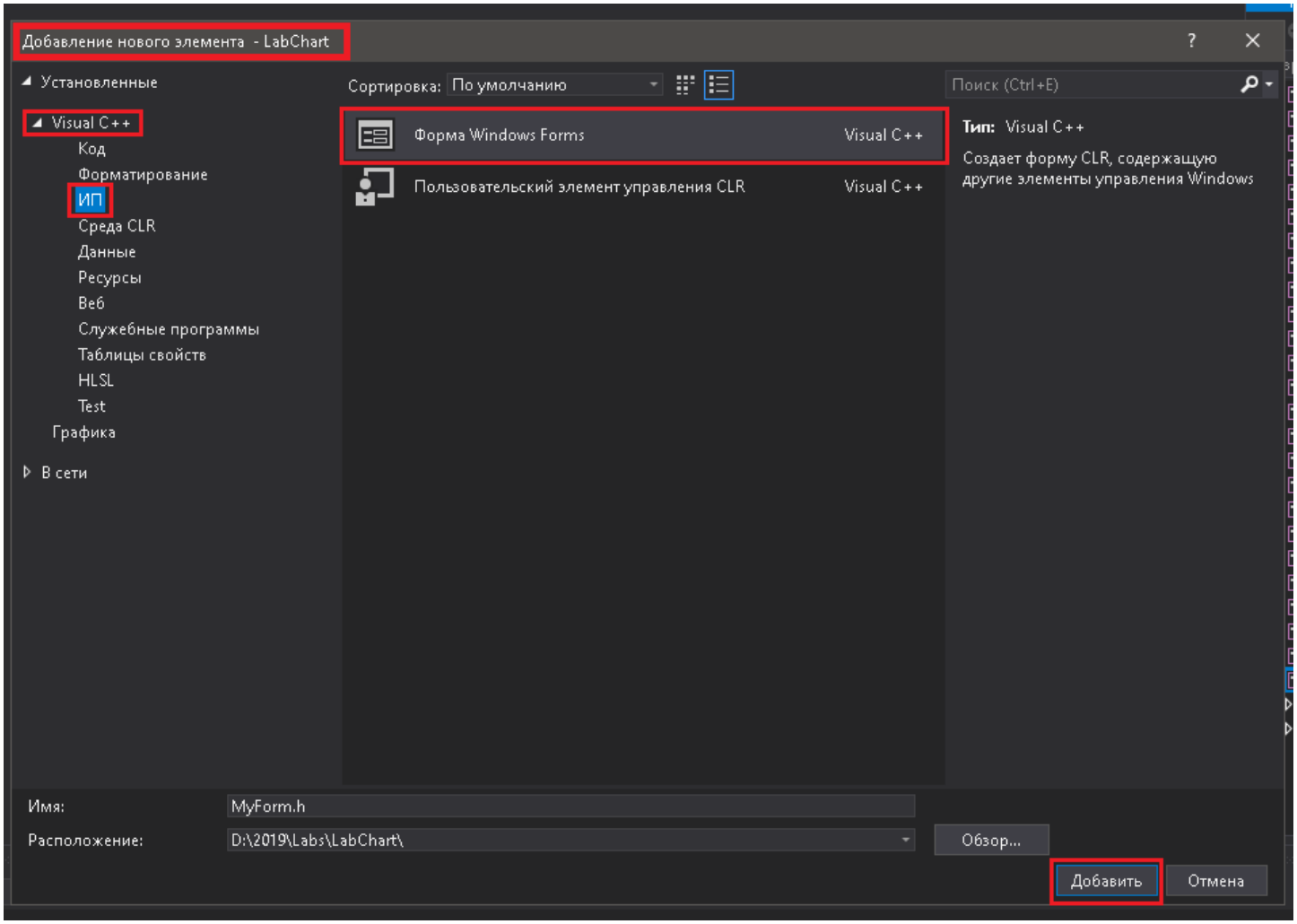
Также в нашем приложении по нажатию кнопки будет появляться дополнительное окно с данными, которые вводились пользователем и были известны только в окне с кнопкой.

Задание 1 (выполнить)

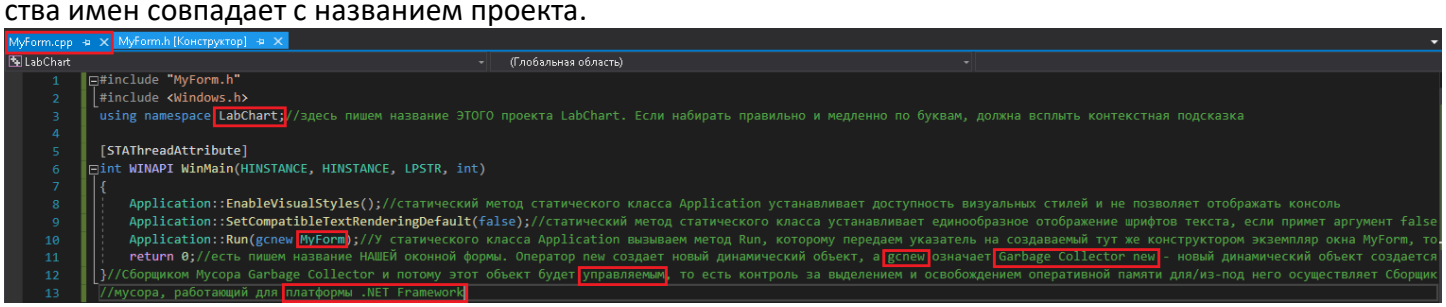
Сначала создаем как в 28-й лабораторной работе новый проект типа «Пустой проект CLR» с названием LabChart.



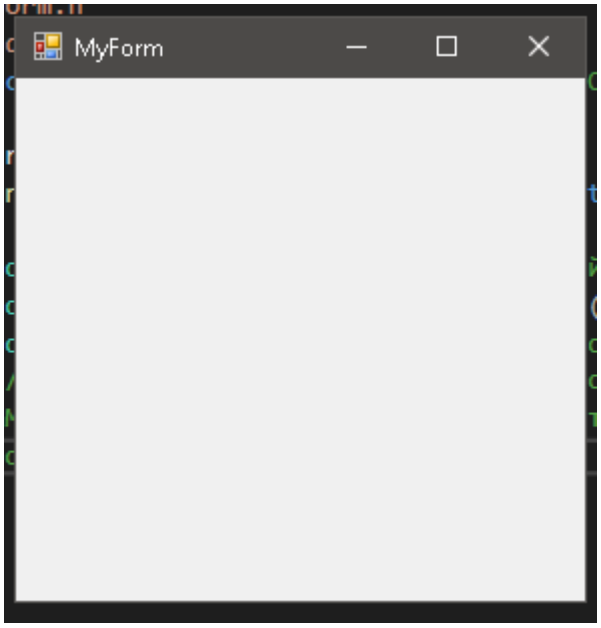
В проекте создаем главную оконную форму, нажимая правой кнопкой мыши по проекту / Добавить / Создать элемент / Visual C++ / ИП / Форма Windows Forms / Добавить. **ИП** – это аббревиатура от «Интерфейс пользователя», то есть user interface или **UI**.



Открываем и дописываем MyForm.cpp файл, помня, что в каждом проекте название пространства имен совпадает с названием проекта.



Сохраняем, компилируем, запускаем проект. Если нет ошибок, то отработает оконная форма вашего нового приложения с базовым функционалом. В ином случае надо перепроверить код, а также, возможно, может потребоваться сохранить проект, закрыть MS VS 2019, открыть ее снова и перекомпилировать проект.



Пусть у нас по нажатии на кнопку типа класса Button запускается вспомогательное окно приложения, которое обладает бОльшими возможностями, чем окно сообщения MessageBox, но запускается только после отображения главной оконной формы и нажатия на соответствующую кнопку на главной оконной форме. Нам нужно:

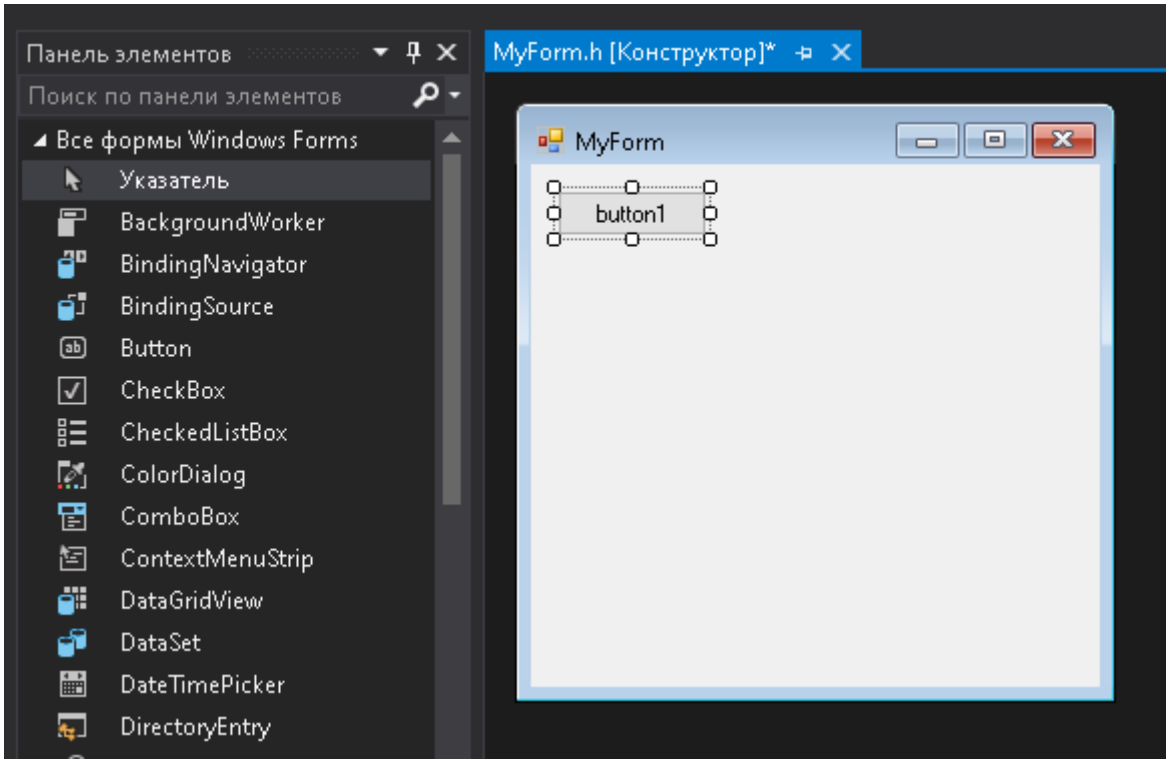
в режиме графического Конструктора расположить на главной оконной форме кнопку button1,

создать обработчик нажатия левой кнопкой мыши по этой кнопке,

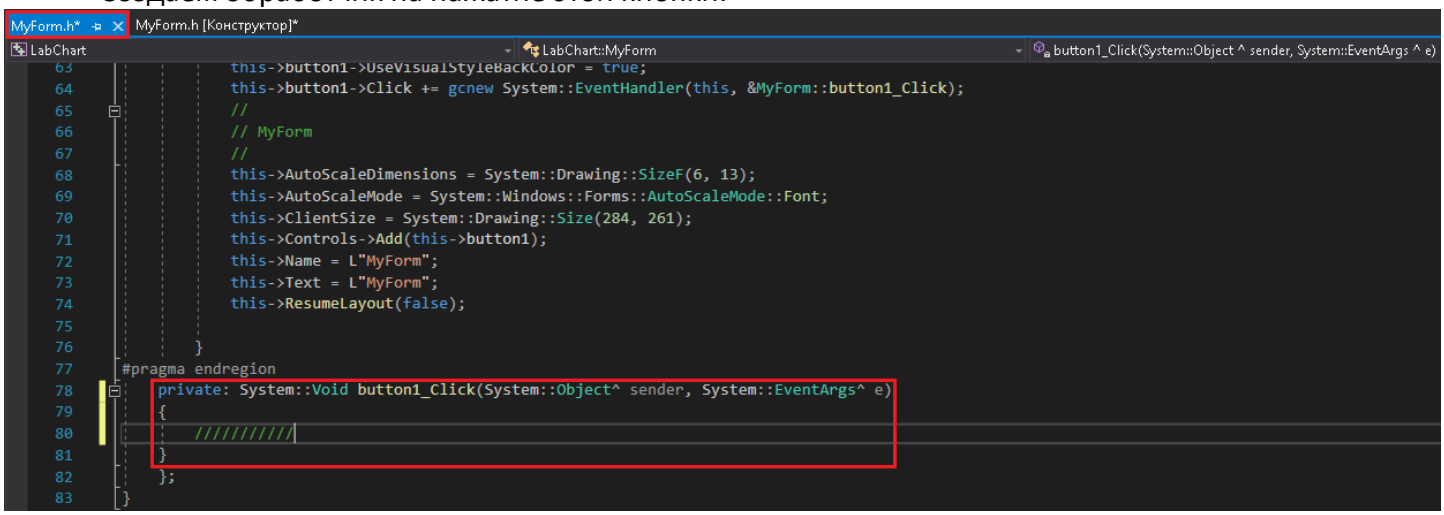
создать вторую оконную форму в нашем приложении,

дописать обработчик и MyForm.h файл основного окна, чтобы отображалось вспомогательное окно. Вспомогательное окно у нас будет **НЕ**модальным, то есть его отображение не будет блокировать наши действия с главным окном, а закрывая главное окно, мы закрываем приложение и тем самым уничтожаются все производные окна нашего приложения. **Модальное окно** – окно в приложении, которое в случае своего отображения на экране ПК блокирует действия пользователя с главным окном до тех пор, пока не будет закрыто это модальное окно, то есть есть без закрытия этого вспомогательного окна наши действия по главному окну не будут обрабатываться нашим приложением до тех пор, пока мы не закроем модальное окно.

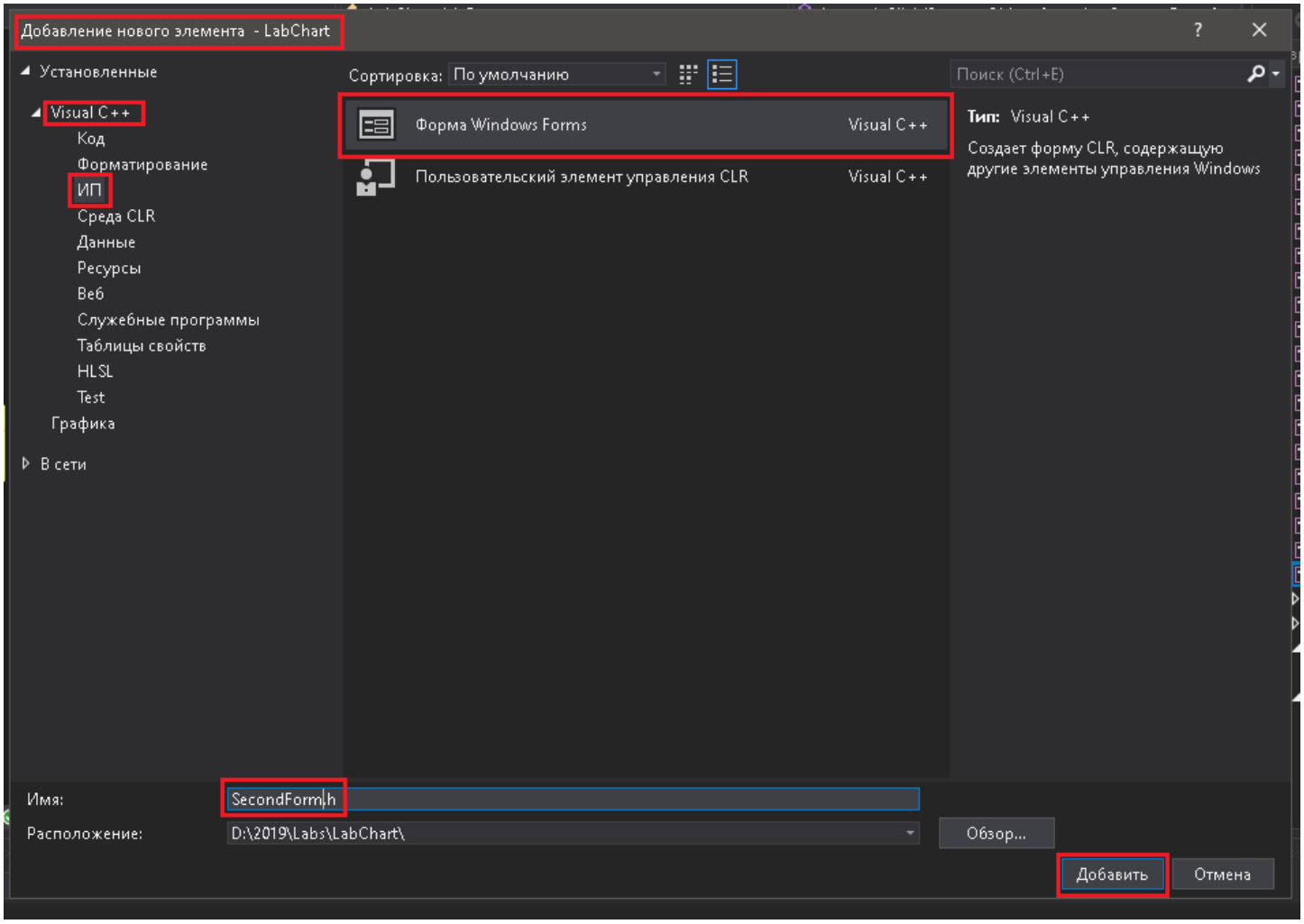
Сначала располагаем кнопку на макете главной оконной формы. Если не получается сразу открыть MyForm.h файл в режиме графического Конструктора, то закройте и сохраните все файлы проекта, закройте MS VS 2019 и откройте снова, нажав в Обозревателе решений на файл MyForm.h правой кнопкой мыши, во всплывающем контекстном меню вызовите пункт Открыть в Конструкторе.



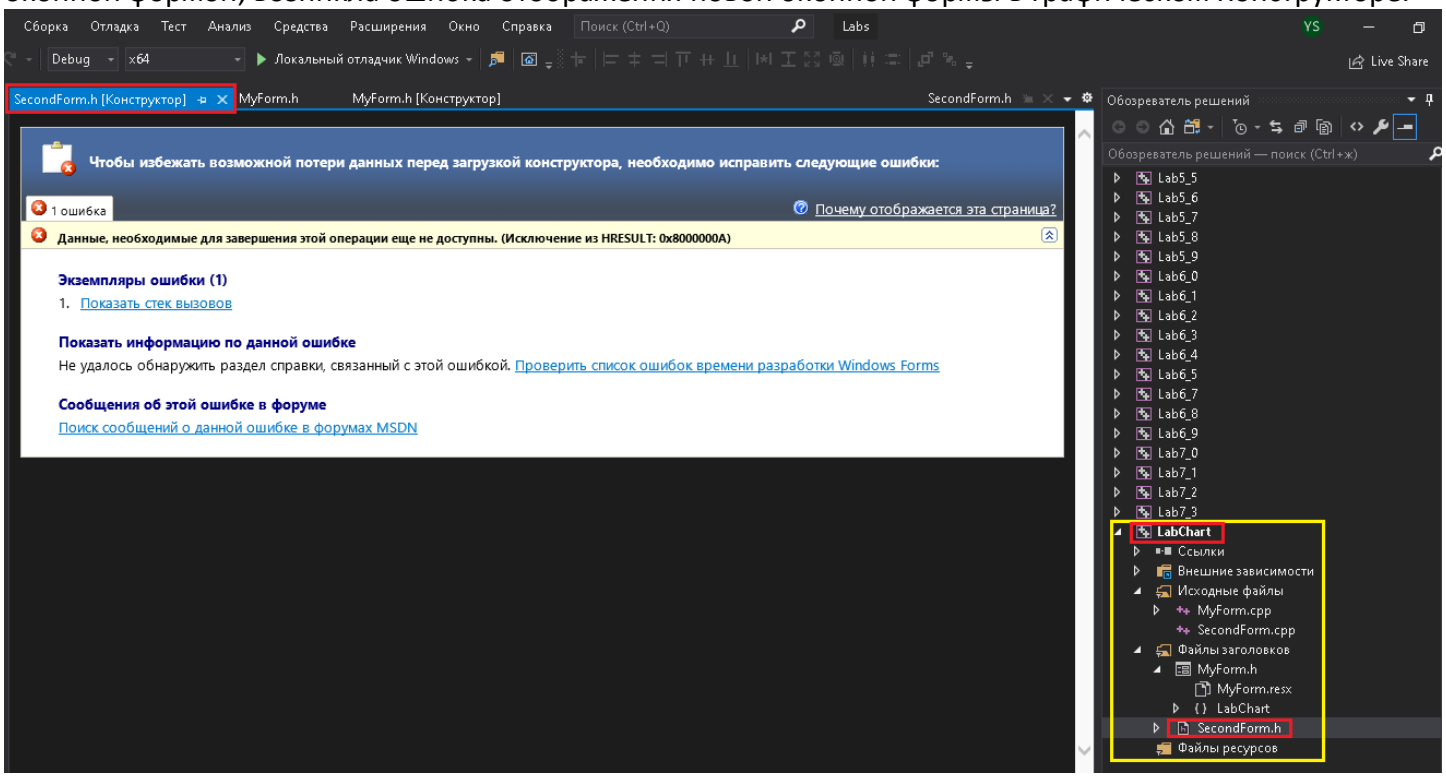
Создаём обработчик на нажатие этой кнопки:



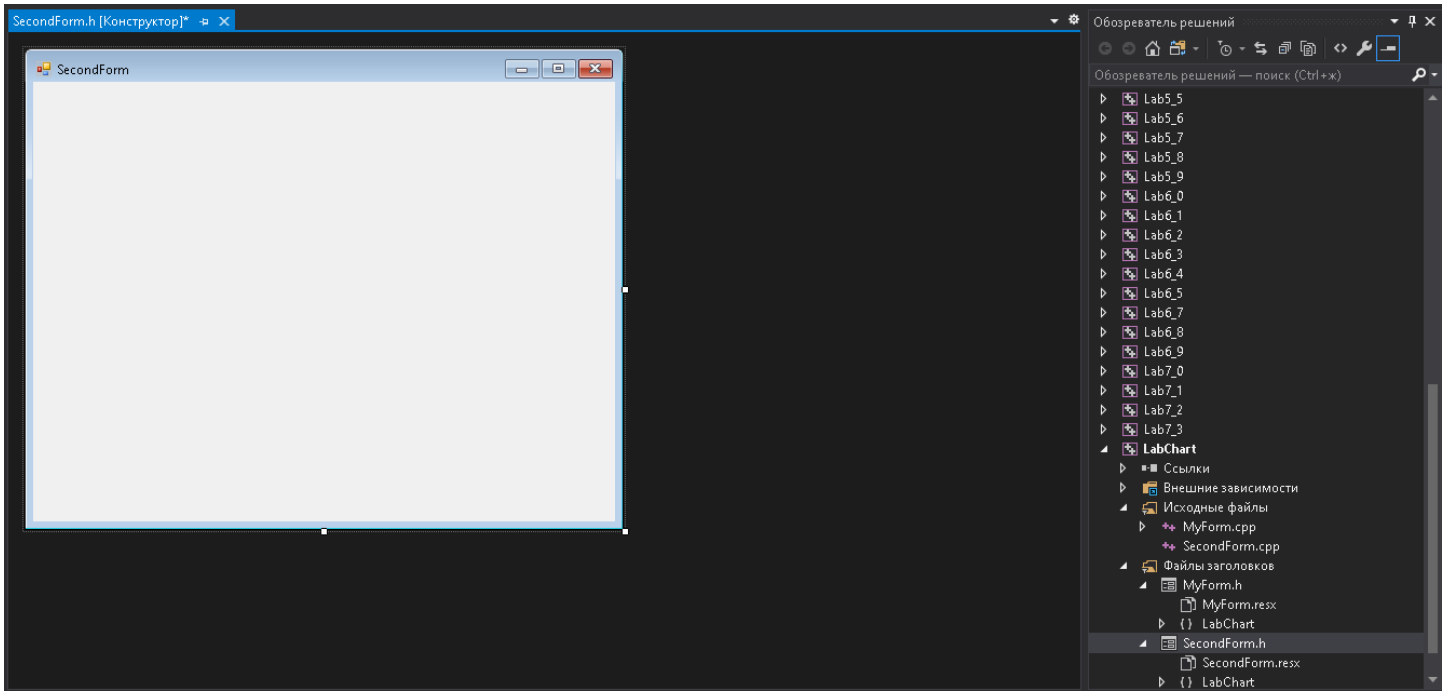
Создаем вторую оконную форму в нашем приложении, для чего нажимаем на имя проекта в окне Обозревателя решений, жмем правой кнопкой мыши по имени проекта LabChart и выбираем в контекстном меню пункт Добавить / Создать элемент / Visual C++ / ИП / Форма Windows Forms / Вводим имя: SecondForm / Добавить. **ИП** – это аббревиатура от «Интерфейс пользователя», то есть user interface или **UI**. Как видите, создание новой оконной формы совпадает с созданием главной оконной формы, ведь они являются экземплярами одного класса Form. Таким образом можно создать много оконных форм в одном приложении, но у этих экземпляров окон должны быть разные имена (точнее, разные имена указателей на них). Но пока вы не пропишете в коде MyForm.h файла приложения вызов конкретных окон, они будут в проекте, но не будут отображаться в приложении при его работе.



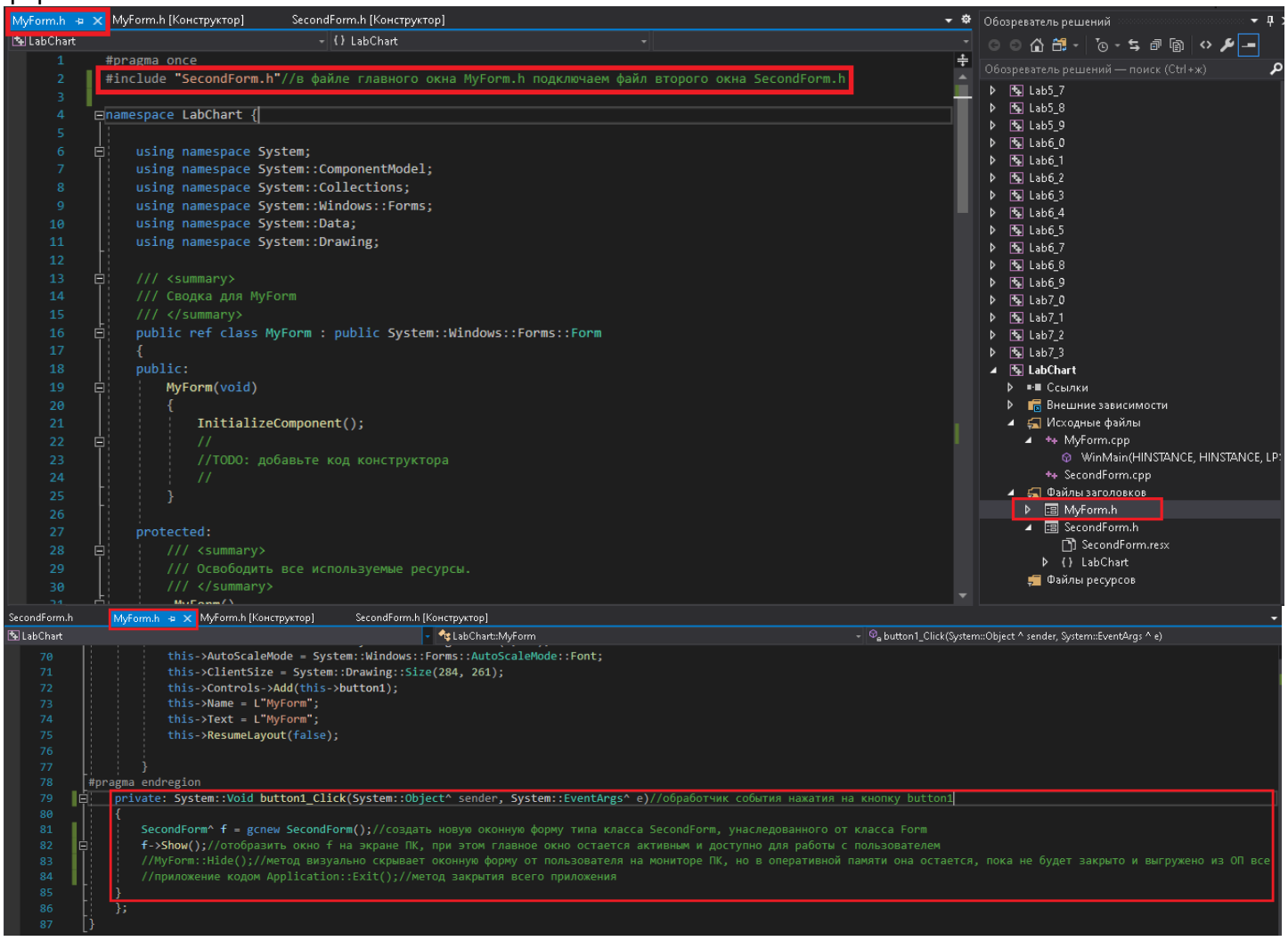
Оконная форма и сам файл SecondForm.h создался в проекте, но, как и в случае с предыдущей оконной формой, возникла ошибка отображения новой оконной формы в графическом Конструкторе.



Если не получается сразу открыть MyForm.h файл в режиме графического Конструктора, то закройте и сохраните все файлы проекта, закройте MS VS 2019 и откройте снова, нажав в Обозревателе решений на файл SecondForm.h правой кнопкой мыши, во всплывающем контекстном меню вызовите пункт Открыть в Конструкторе.

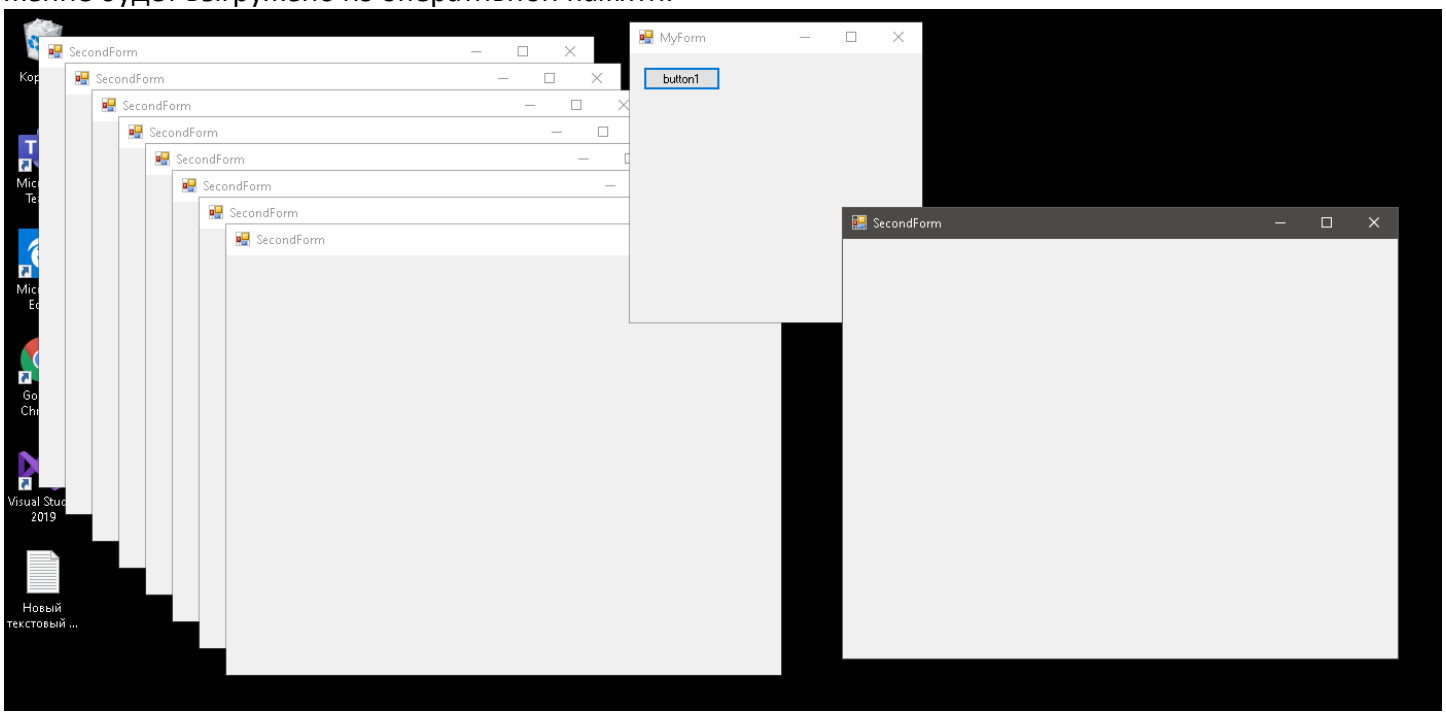


Дописываем код файла MyForm.h, поскольку наше вспомогательное окно вызывается по нажатию кнопки на окне MyForm, код которого содержится в файле MyForm.h. Файл MyForm.h дописывается в двух местах: во-первых, нужно подключить файл оконной формы SecondForm.h, а во-вторых, нам нужно дописать обработчик события нажатия на кнопку, чтобы отобразилась вспомогательная оконная форма.

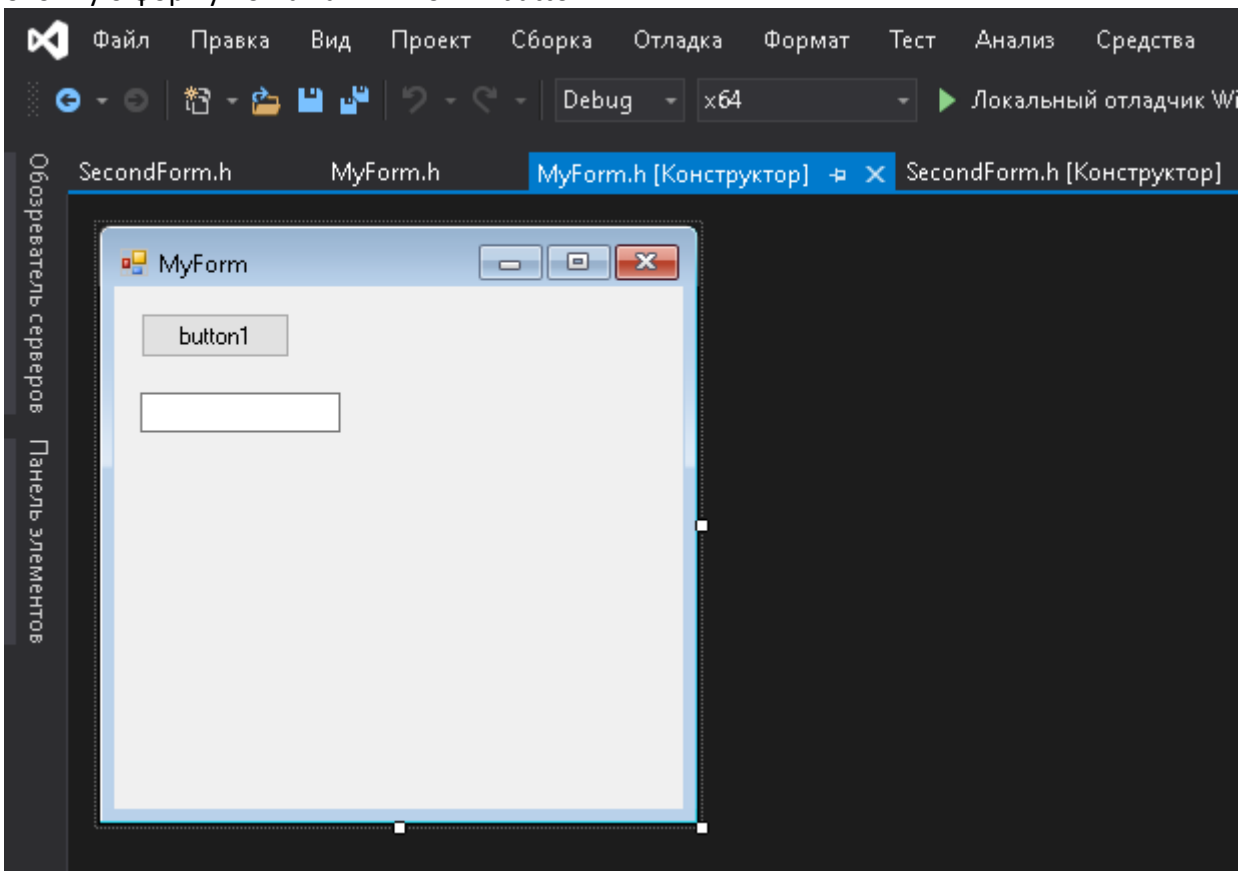


Тестируем. Видим, что главная оконная форма доступна пользователю, а значит пользователь может нажимать на кнопку создания вспомогательных оконных форм типа SecondForm, которые создаются. Приложение будет работать до тех пор, пока пользователь не закроет главную оконную форму. Если пользователь закроет главную оконную форму при запущенных вспомогательных формах, то

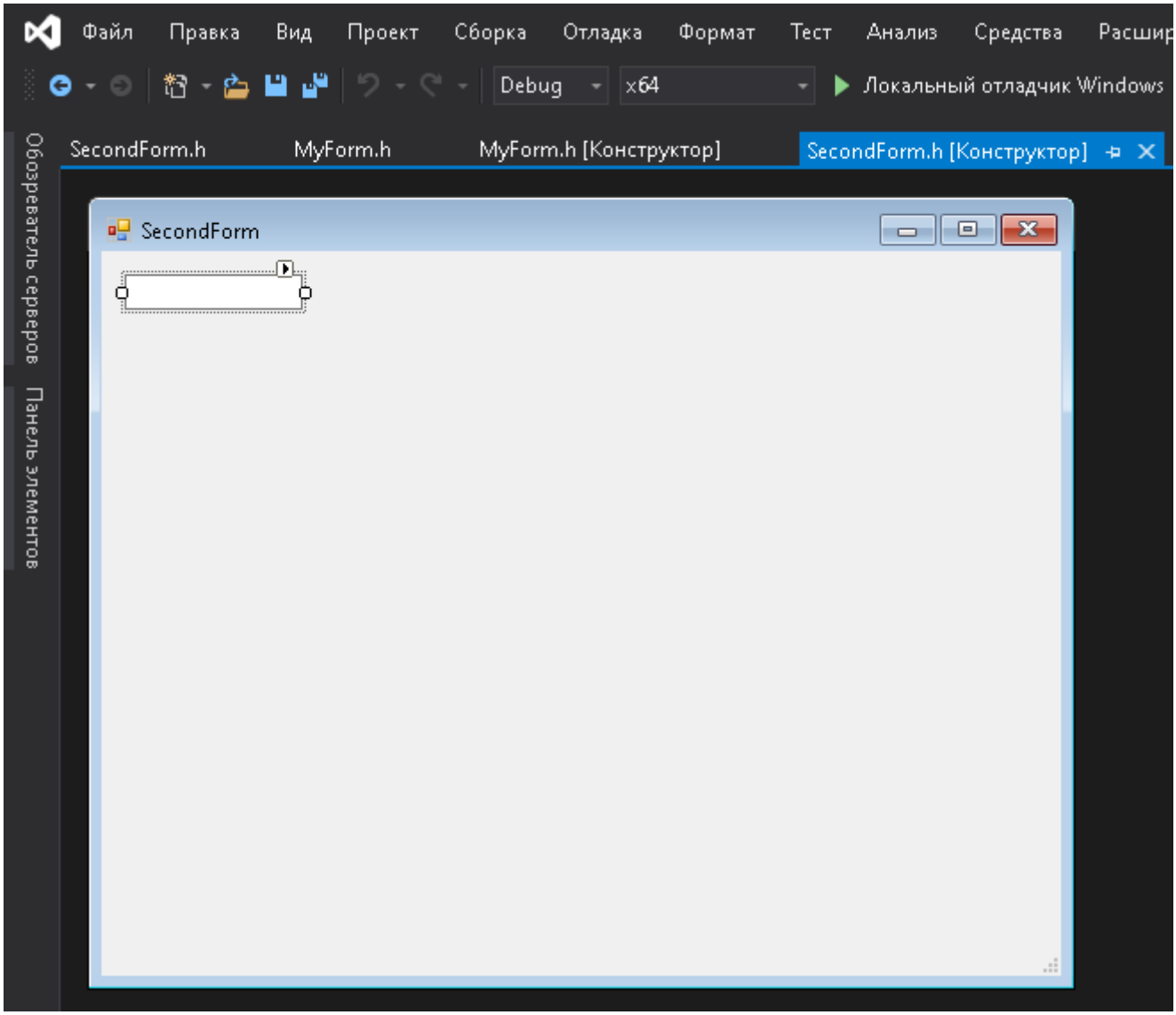
они будут уничтожены, скрыты с монитора ПК и последним закроется главная оконная форма и приложение будет выгружено из оперативной памяти.



Попробуем из начальной оконной формы передать данные (например, вещественное значение) во вспомогательную оконную форму. Для этого создадим на главной оконной форме текстБокс1, в который пользователь будет вводить вещественное число, а оно будет передаваться во вспомогательную оконную форму по нажатии кнопки button1.



В окне вспомогательной оконной формы создадим «местный» текстБокс1, в котором уже при открытии вспомогательного окна должно отобразиться значение, которое в него передается из главного окна. Для этого окна указатель this указывает на само данное вспомогательное окно, то есть оконную форму SecondForm^ f.



Значение во вспомогательную оконную форму будем передавать в конструкторе вызова. По умолчанию у нас создан роботом конструктор без параметров:

```
public ref class SecondForm : public System::Windows::Forms::Form
{
public:
	SecondForm(void)
	{
		InitializeComponent();
		//
		//TODO: добавьте код конструктора
		//
	}
```

А мы создадим рядом по этому образцу еще один конструктор с параметром:

```
public ref class SecondForm : public System::Windows::Forms::Form
{
public:
	SecondForm(void)//конструктор ничего не принимает входным аргументом
	{
		InitializeComponent();//обязательный вызов метода инициализации компонентов оконной формы
		//
		//TODO: добавьте код конструктора
		//
	}
```

SecondForm(double x)//пишем свой конструктор, принимающий один вещественный параметр
{
InitializeComponent();//обязательный вызов метода инициализации компонентов оконной формы
this->textBox1->Text = x.ToString();//входное вещественное значение переменной x приводим к строковому (чаровский массив) виду и присваиваем его свойству Text у текстБокса1 этой оконной формы
}//поскольку код конструктора срабатывает при создании объекта данного класса, то наше значение сразу поместится в текстБокс1, как только отработает метод InitializeComponent();

Код файла SecondForm.h примет вид:

```
#pragma once

namespace LabChart {

	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 SecondForm
	/// </summary>
	public ref class SecondForm : public System::Windows::Forms::Form
	{
	public:
		SecondForm(void)//конструктор ничего не принимает входным аргументом
		{
			InitializeComponent();//обязательный вызов метода инициализации компонентов оконной формы
			//
			//TODO: добавьте код конструктора
			//
		}
		SecondForm(double x)//пишем свой конструктор, принимающий один вещественный параметр
		{
			InitializeComponent();//обязательный вызов метода инициализации компонентов оконной формы
			this->textBox1->Text = x.ToString();//входное вещественное значение переменной x приводим к строковому (чаровский массив) виду и присваиваем его свойству Text у текстБокса1 этой оконной формы
		}//поскольку код конструктора срабатывает при создании объекта данного класса, то наше значение сразу поместится в текстБокс1, как только отработает метод InitializeComponent();
	protected:
		/// <summary>
		/// Освободить все используемые ресурсы.
		/// </summary>
		~SecondForm()
		{
			if (components)
```

```
		{
			delete components;
		}
	}
private: System::Windows::Forms::TextBox^ textBox1;
protected:
private:
	/// <summary>
	/// Обязательная переменная конструктора.
	/// </summary>
	System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
	/// <summary>
	/// Требуемый метод для поддержки конструктора — не изменяйте
	/// содержимое этого метода с помощью редактора кода.
	/// </summary>
	void InitializeComponent(void)
	{
		System::ComponentModel::ComponentResourceManager^ resources = (gcnew System::ComponentModel::ComponentResourceManager(SecondForm::typeid));
		this->textBox1 = (gcnew System::Windows::Forms::TextBox());
		this->SuspendLayout();
		//
		// textBox1
		//
		resources->ApplyResources(this->textBox1, L"textBox1");
		this->textBox1->Name = L"textBox1";
		//
		// SecondForm
		//
		resources->ApplyResources(this, L"$this");
		this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
		this->AutoValidate = System::Windows::Forms::AutoValidate::EnableAllowFocusChange;
		this->CausesValidation = false;
		this->Controls->Add(this->textBox1);
		this->DoubleBuffered = true;
		this->HelpButton = true;
		this->Name = L"SecondForm";
		this->SizeGripStyle = System::Windows::Forms::SizeGripStyle::Show;
		this->ResumeLayout(false);
		this->PerformLayout();
	}
#pragma endregion
	};
}
```

В коде файла MyForm.h главного окна приложения надо поменять код. Мы создадим вспомогательную оконную форму с помощью конструктора с вещественным параметром, а не конструктором без параметров. Но сначала нам надо взять значение из текстБокса1 главной оконной формы, конвертировать находящийся в нем текст в значение вещественного типа (в ходе чего может возникнуть ошибка, если в текстБоксе1 будут буквы или не будет вообще никаких символов) и передать это вещественное значение конструктору с параметром, который создаст вспомогательную оконную форму. Изменяем код в обработчике button1_Click() на такой:

private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e)//обработчик события нажатия на кнопку button1

{

double a;

try//в блоке кода try{} отслеживаем возможные ошибки трех типов

{

a = Double::Parse(this->textBox1->Text);//статический класс Double содержит метод Parse(), который принимает значение из текстБокса1, конвертирует его в вещественной значение и возвращает результат, который мы помещаем в вещественную переменную a. Если конвертируемое значение не походит на вещественное число, например, это слова, буквы, нечисловые символы, отсутствие любых символов, то произойдет ошибка конвертирования. Разделителем целой и дробной частей должна выступать ЗАПЯТАЯ в нашем языковом регионе

SecondForm^ f = gcnew SecondForm(a);//создать новую оконную форму типа класса SecondForm конструктором с вещественным параметром, куда передаем значение из переменной a. Имя оконной формы ^f - это указатель на оконную форму

f->Show();//отобразить НЕмодальное окно f на экране ПК, при этом главное окно остается активным и доступно для работы с пользователем

//MyForm::Hide();//метод визуально скрывает главную оконную форму от пользователя на мониторе ПК, но в оперативной памяти она остается, пока не будет закрыто и выгружено из ОП все приложение кодом Application::Exit();//метод закрытия всего приложения

//this->Enabled = false;//это выражение делает главное окно приложение НЕреагирующим на дествия мыши пользователя. появившееся вспомогательное окно при этом доступно для действий мыши пользователя, ведь его свойство по умолчанию this->Enabled = true;

}

catch (System::ArgumentNullException^ e)//отлавливаем недопустимую пустую (непроинициализированную) ссылку

{//сообщаем пользователю информацию в мини-окне MessageBox'a, который является модальным окном (пока он существует, основное окно заблокировано для действий пользователя)

MessageBox::Show(this, "Пустая ссылка.", "Внимание", MessageBoxButtons::OK, MessageBoxIcon::Error);//окну сообщения можно присваивать заголовок, текст, назначать кнопки, иконку

}

catch (System::FormatException^ e)//отлавливаем ошибку некорректного формата данных

{

MessageBox::Show(this, "Введите вещественное число.", "Внимание", MessageBoxButtons::OK, MessageBoxIcon::Warning);

}

catch (System::OverflowException^ e)//отлавливаем ошибку переполнения значения переменной некоторого типа данных

{

MessageBox::Show(this, "ВВедите число подходящего размера.", "Слишком большое или маленькое число", MessageBoxButtons::OK, MessageBoxIcon::Stop);

}

}

Весь код файла MyForm.h:

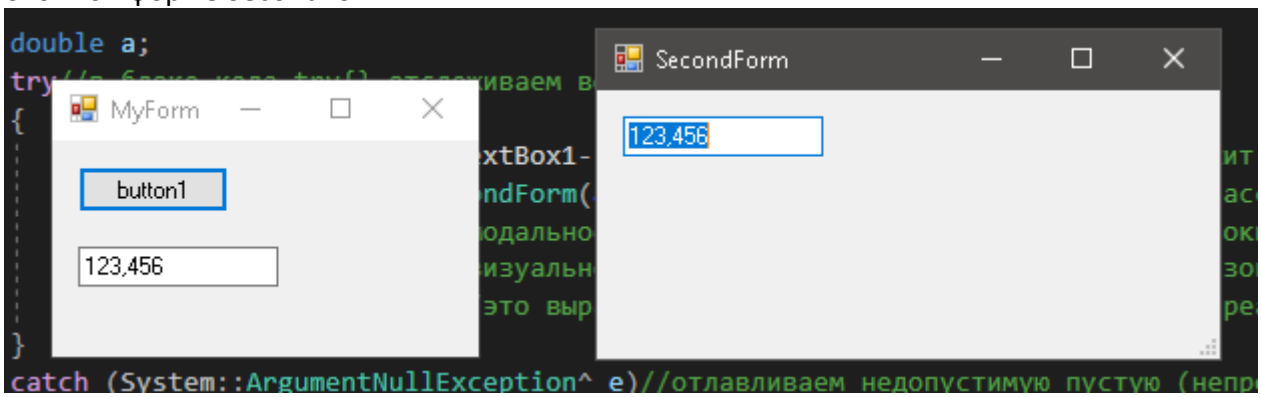
```
#pragma once
#include "SecondForm.h"//в файле главного окна MyForm.h подключаем файл второго окна SecondForm.h

namespace LabChart {

	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
		//
	}

protected:
	/// <summary>
	/// Освободить все используемые ресурсы.
	/// </summary>
	~MyForm()
	{
		if (components)
		{
			delete components;
		}
	}
protected: System::Windows::Forms::Button^ button1;
private: System::Windows::Forms::TextBox^ textBox1;
protected:
private:
	/// <summary>
	/// Обязательная переменная конструктора.
	/// </summary>
	System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
	/// <summary>
	/// Требуемый метод для поддержки конструктора — не изменяйте
	/// содержимое этого метода с помощью редактора кода.
	/// </summary>
	void InitializeComponent(void)
	{
		this->button1 = (gcnew System::Windows::Forms::Button());
		this->textBox1 = (gcnew System::Windows::Forms::TextBox());
		this->SuspendLayout();
		//
		// button1
		//
		this->button1->Location = System::Drawing::Point(13, 13);
		this->button1->Name = L"button1";
```

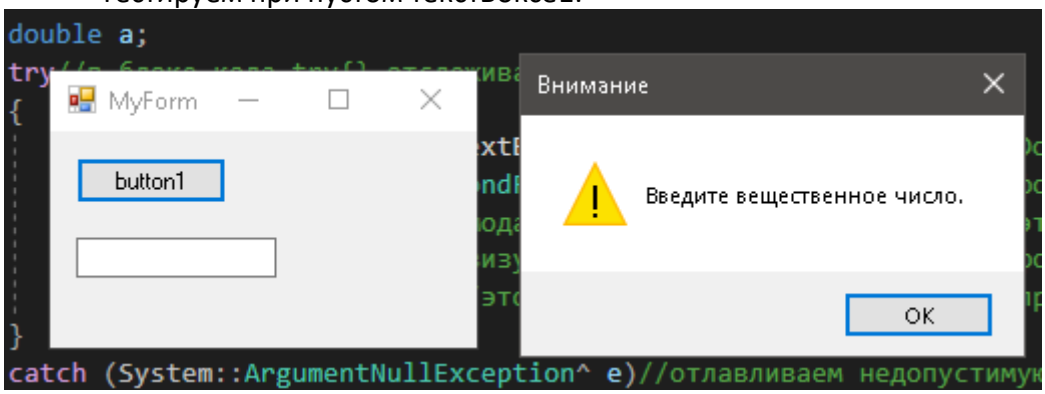
```
            this->button1->Size = System::Drawing::Size(75, 23);
            this->button1->TabIndex = 0;
            this->button1->Text = L"button1";
            this->button1->UseVisualStyleBackColor = true;
            this->button1->Click += gcnew System::EventHandler(this, &MyForm::button1_Click);
            //
            // textBox1
            //
            this->textBox1->Location = System::Drawing::Point(13, 53);
            this->textBox1->Name = L"textBox1";
            this->textBox1->Size = System::Drawing::Size(100, 20);
            this->textBox1->TabIndex = 1;
            //
            // MyForm
            //
            this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
            this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
            this->ClientSize = System::Drawing::Size(284, 261);
            this->Controls->Add(this->textBox1);
            this->Controls->Add(this->button1);
            this->Name = L"MyForm";
            this->Text = L"MyForm";
            this->ResumeLayout(false);
            this->PerformLayout();

        }
#pragma endregion
    private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e)//обработчик события нажатия на кнопку button1
    {
        double a;
        try//в блоке кода try{} отслеживаем возможные ошибки трех типов
        {
            a = Double::Parse(this->textBox1->Text);//статический класс Double содержит метод Parse(), который принимает значение из текстБокса1, конвертирует его в вещественной значение и возвращает результат, который мы помещаем в вещественную переменную a. Если конвертируемое значение не походит на вещественное число, например, это слова, буквы, нечисловые символы, отсутствие любых символов, то произойдет ошибка конвертирования. Разделителем целой и дробной частей должна выступать ЗАПЯТАЯ в нашем языковом регионе
            SecondForm^ f = gcnew SecondForm(a);//создать новую оконную форму типа класса SecondForm конструктором с вещественным параметром, куда передаем значение из переменной a. Имя оконной формы ^f - это указатель на оконную форму
            f->Show();//отобразить НЕмодальное окно f на экране ПК, при этом главное окно остается активным и доступно для работы с пользователем
            //MyForm::Hide();//метод визуально скрывает главную оконную форму от пользователя на мониторе ПК, но в оперативной памяти она остается, пока не будет закрыто и выгружено из ОП все приложение кодом Application::Exit();//метод закрытия всего приложения
            //this->Enabled = false;//это выражение делает главное окно приложение НЕреагирующим на дествия мыши пользователя. появившееся вспомогательное окно при этом доступно для действий мыши пользователя, ведь его свойство по умолчанию this->Enabled = true;
        }
        catch (System::ArgumentNullException^ e)//отлавливаем недопустимую пустую (непроинициализированную) ссылку
```

```
                {//сообщаем пользователю информацию в мини-окне MessageBox'a, который является модальным окном (пока он существует, основное окно заблокировано для действий пользователя)
                    MessageBox::Show(this, "Пустая ссылка.", "Внимание", MessageBoxButtons::OK, MessageBoxIcon::Error);//окну сообщения можно присваивать заголовок, текст, назначать кнопки, иконку
                }
                catch (System::FormatException^ e)//отлавливаем ошибку некорректного формата данных
                {
                    MessageBox::Show(this, "Введите вещественное число.", "Внимание", MessageBoxButtons::OK, MessageBoxIcon::Warning);
                }
                catch (System::OverflowException^ e)//отлавливаем ошибку переполнения значения переменной некоторого типа данных. Есть ли разница, в каком порядке располагать отлавливатели ошибок?
                {
                    MessageBox::Show(this, "ВВедите число подходящего размера.", "Слишком большое или маленькое число", MessageBoxButtons::OK, MessageBoxIcon::Stop);
                }
            }
};
}
```

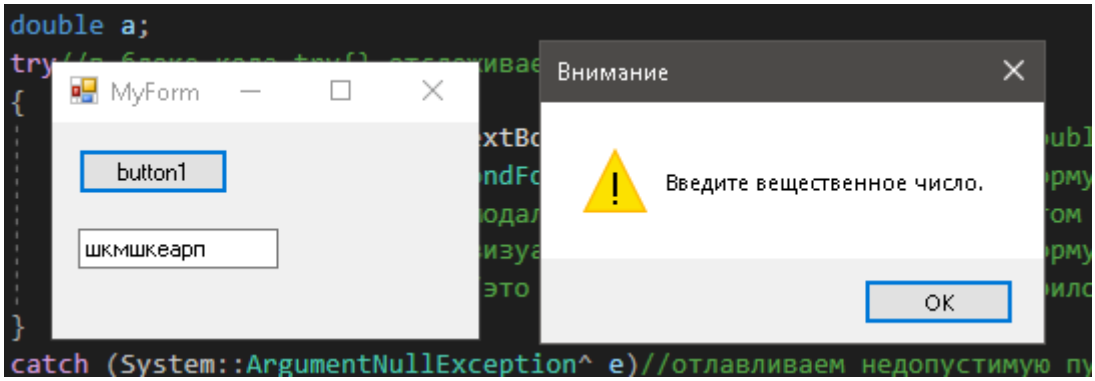
Тестируем передачу значения из главной во вспомогательную оконную форму. В текстБокс1 окна MyForm я ввел число 123,456, которое по нажатию на кнопку button1 отображалось в появившейся оконной форме SecondForm:



Тестируем при пустом текстБоксе1:



Тестируем на буквенных значениях вместо числовых:

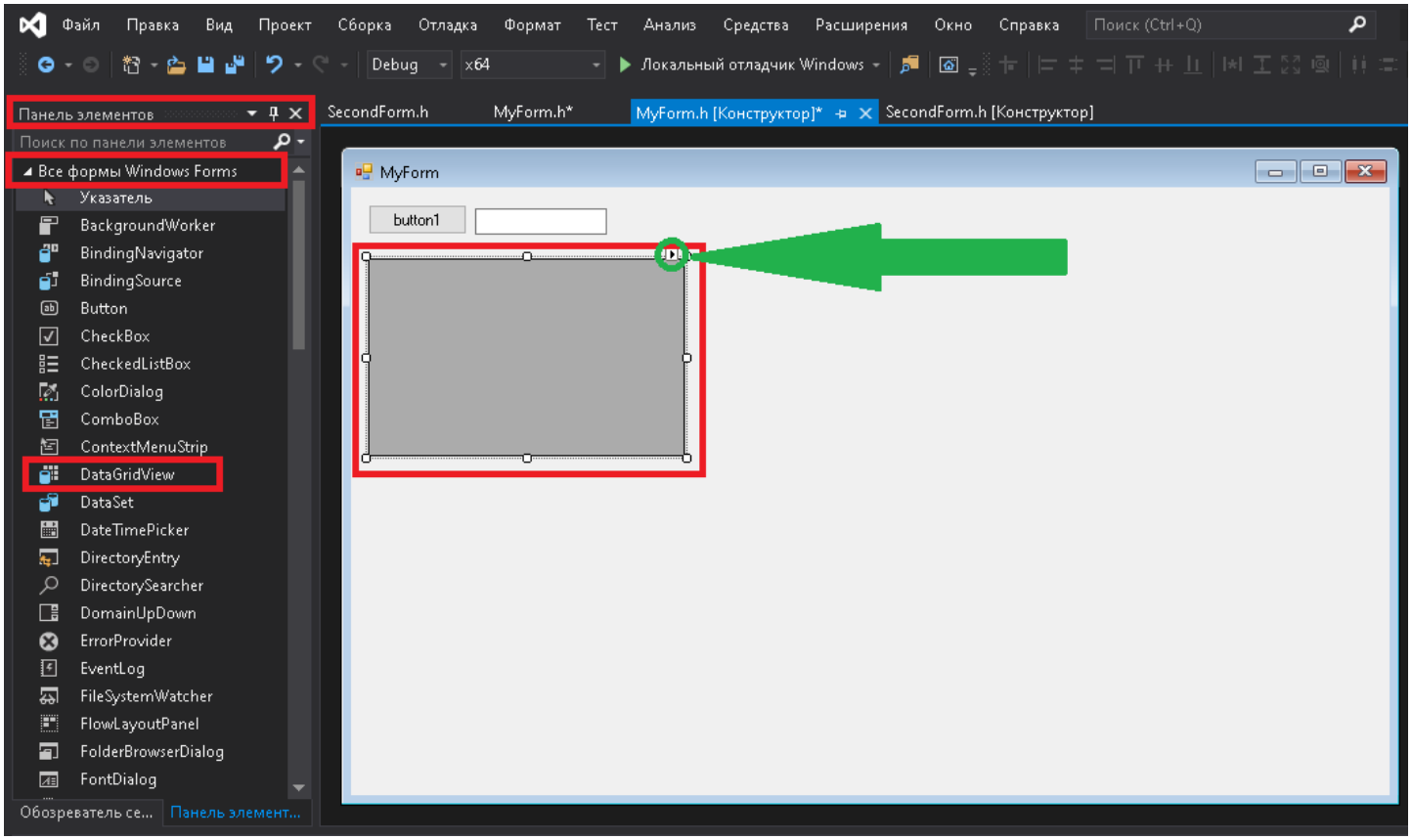


Элемент Chart предназначен для вывода в экранную форму графика (диаграммы). Удобно строить этот график по табличным данным, представленным в виде объекта класса DataTable (таблицы данных).

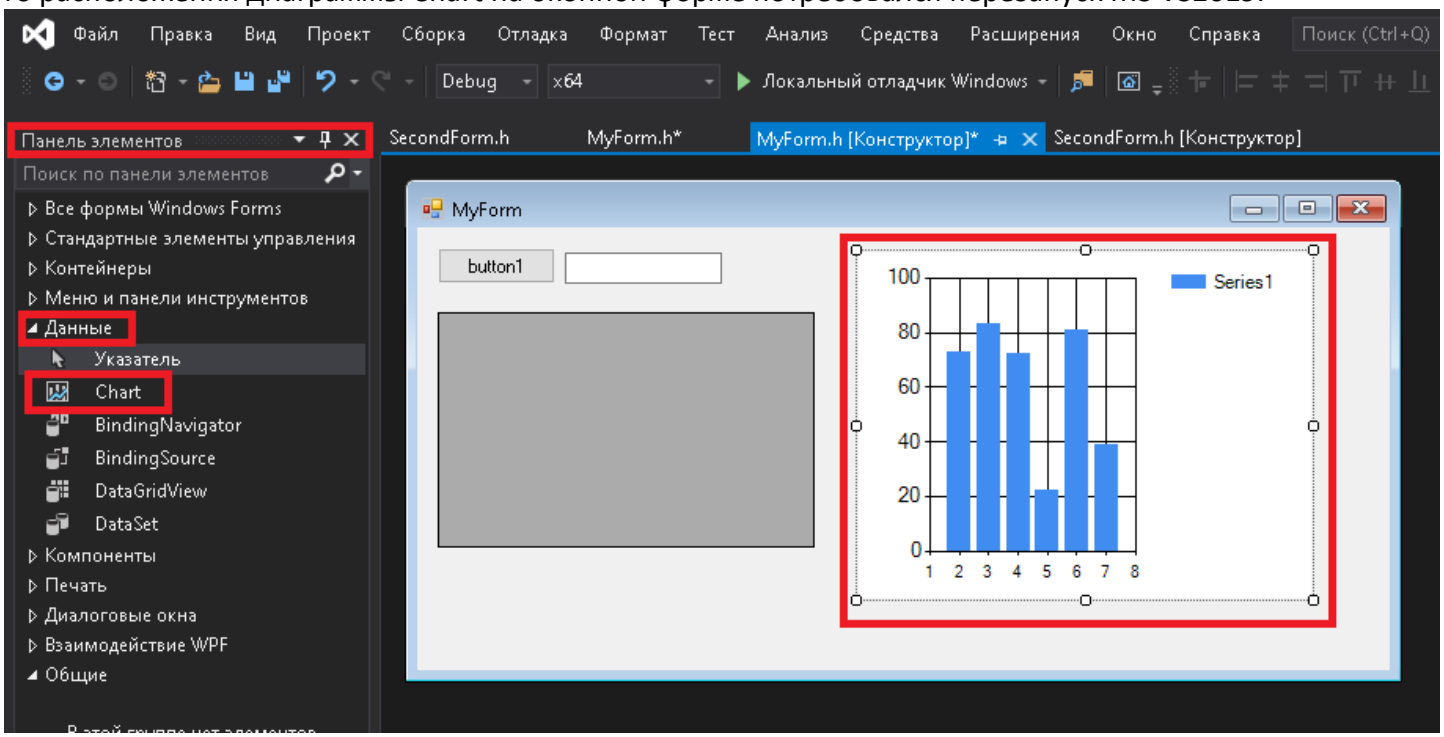
Пусть даны сведения об объемах продаж за пять месяцев. Предусмотреть ввод пользователем новых данных, возможность редактирования старых данных и удаления данных за любой месяц. Требуется наглядно визуализировать эти данные на графике в виде гистограммы.

В объекте класса DataGridView данные отображаются на мониторе пользователя, но реально данные хранятся в объекте класса DataTable. Этот объект представляет собой одну таблицу данных в оперативной памяти. Чтобы визуализировать эту таблицу на экране, используют элемент — сетку данных (визуальную таблицу) DataGridView. Объект класса DataTable используют в качестве исходных данных и для сетки данных DataGridView, и для диаграммы Chart (например, столбчатая диаграмма). Таблица DataTable определяется в виде схемы, состоящей из колонок, которые имеют свои названия. В процессе эксплуатации таблица DataTable заполняется по строкам (рядам) методом Add(). Заполненную строками таблицу нужно указать в качестве источника данных для элементов Chart и DataGridView. Можно менять внешний вид диаграмм, используя вид по умолчанию (простой двумерный), трехмерный или иной.

Чтобы перетащить на наше окно два крупных элемента, сначала за уголок расширим поле главного окна MyForm нашего приложения (но технически можно реализовать и во вспомогательном окне SecondForm), открытого в режиме Дизайнера. Далее, на вкладке Панель элементов (ToolBox) из подпункта Все формы Windows Forms (All Windows Forms) выбираю мышью элемент DataGridView и кликаю мышью по полю окна формы, чтобы разместить его на ней. Мне отображаются настройки (если нажать мышью по верхнему правому треугольнику у данного элемента), что этот элемент поддерживает вставку, редактирование и удаление данных из него, а источник, откуда берутся данные для заполнения таблицы, пока отсутствует. DataGridView – это элемент в виде таблицы, предназначенный для хранения, отображения, ввода и удаления данных из него. На основе его данных будет строиться диаграмма в моем приложении.



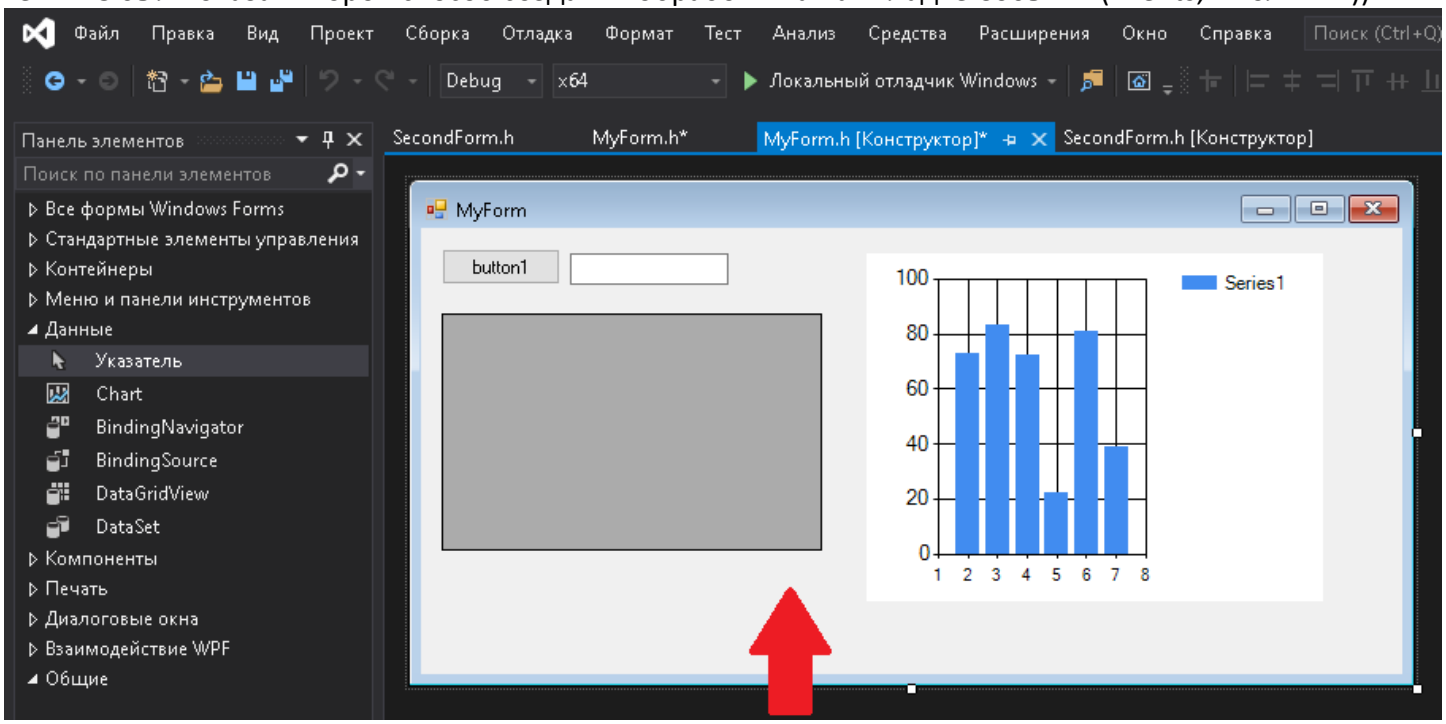
Поместим на оконное приложение элемент Диаграмму Chart. Chart – это элемент в виде диаграммы (столбчатой диаграммы, гистограммы и т.д.), который принимает данные и может строить по ним диаграммы двумерные и трехмерные. В моем случае элемента Chart нет во вкладке подпункта All Windows Forms Панели элементов, но он есть в том окне в подпункте Данные. На вкладке ToolBox из подпункта Data выбираю мышью элемент Chart и кликаю мышью по полю окна формы, чтобы разместить его на ней или растягиваю этот элемент мышью на оконной форме. В моем случае для корректного расположения диаграммы Chart на оконной форме потребовался перезапуск MS VS2019.



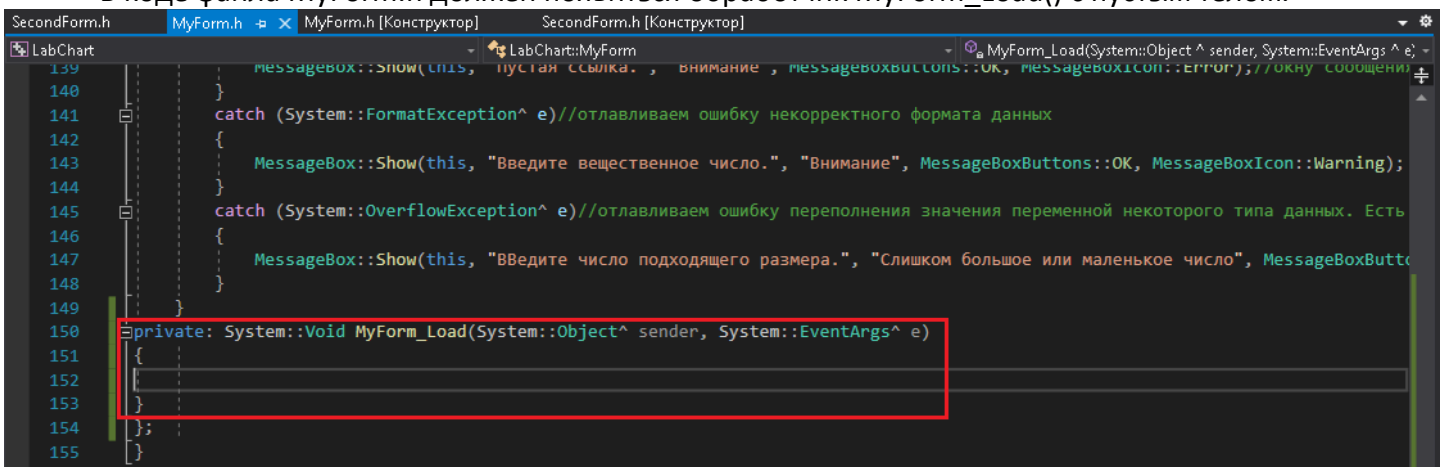
На нашей оконной форме есть два элемента – таблица и диаграмма, но они не имеют поведения и данных. Данные для таблицы должны быть уже до момента ее отображения на оконной форме при запуске приложения, поскольку иначе таблица будет пустой. А из таблицы данные берет элемент Диаграмма и рисует по полученным данным столбчатую диаграмму. Данные в таблице можно редактировать, но диаграмма будет иметь прежний вид, построенный на «старой» версии данных из таблицы.

Сделаем так, чтобы при клике мышью по диаграмме столбцы в ней перерисовывались в соответствии с теми данными, которые она в этот момент возьмет из таблицы.

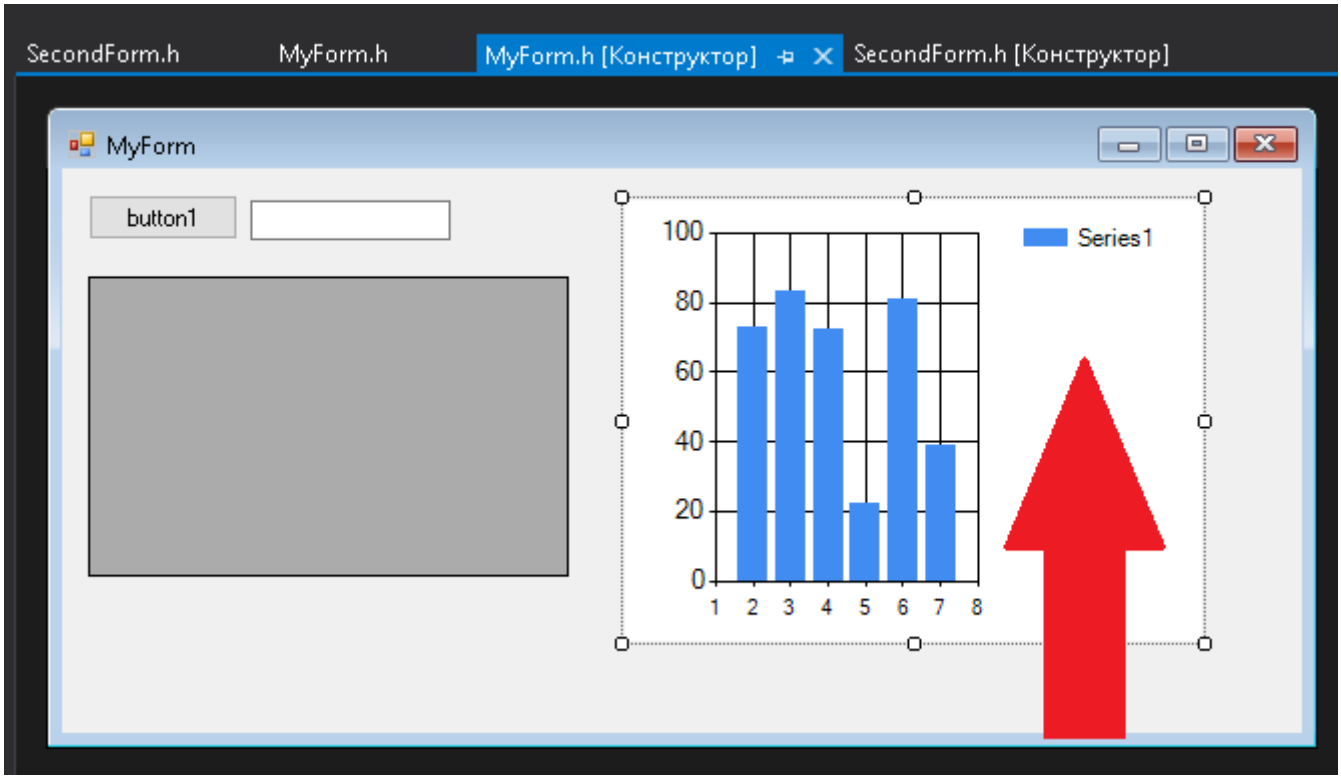
Значит, нам нужен обработчик события загрузки оконной формы, поскольку он отработает раньше отрисовки таблицы DataGridView, а значит именно он сможет дать таблице DataGridView начальные данные. Для создания обработчика события загрузки оконной формы всего приложения дважды кликаем мышью по свободному от уже расположенных графических элементов, рамки, заголовка окна, на макете оконной формы месту в режиме графического Дизайнера (в лабораторной работе № 28 был показан второй способ создания обработчика на вкладке События (Events, «молния»)).



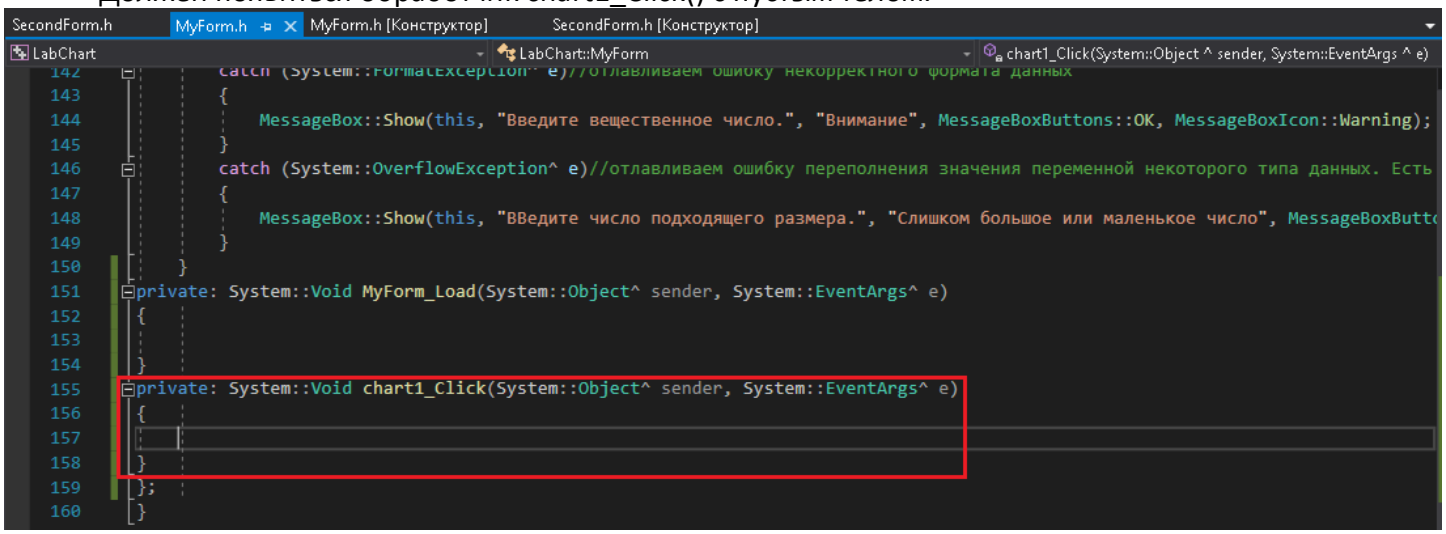
В коде файла MyForm.h должен появиться обработчик MyForm_Load() с пустым телом.



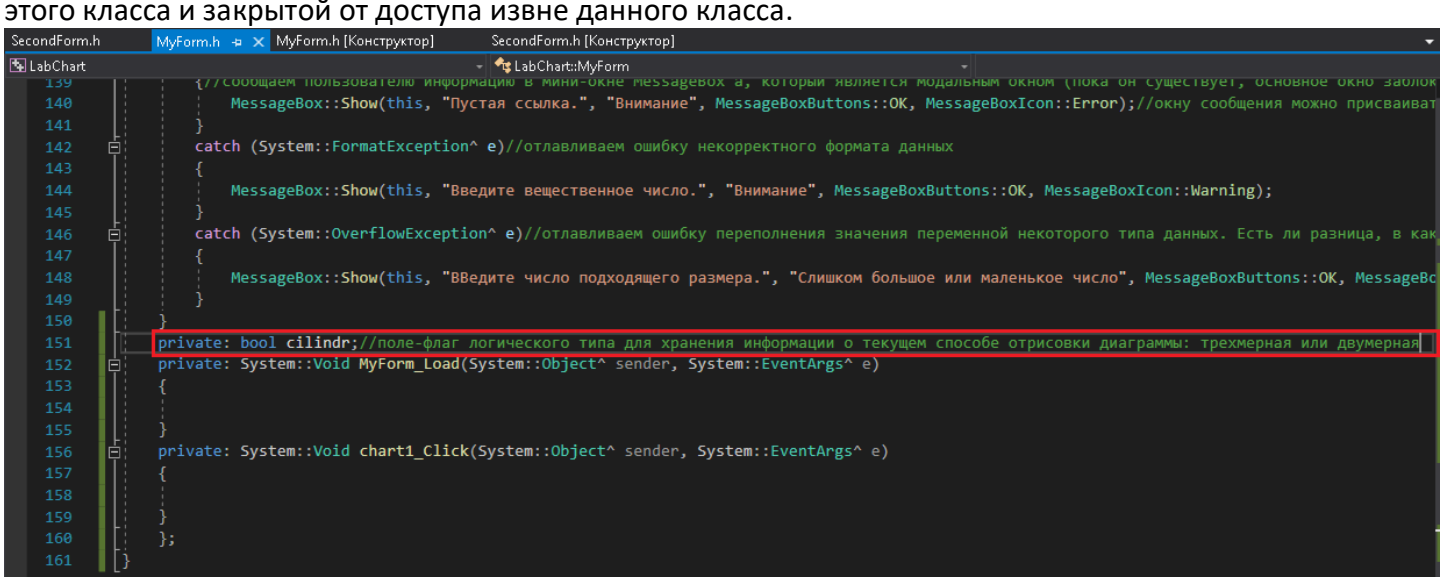
Нужен еще обработчик нажатия мышью по элементу Диаграмма, в теле которого мы напишем код по изменению отображения диаграммы с двумерного на трехмерное и зададим перерисовку диаграммы в соответствии с текущими данными, запрашиваемыми диаграммой у таблицы в момент клика мышью по диаграмме. Для создания такого обработчика дважды нажмем мышью по элементу Диаграмма в окне нашего приложения, отображаемого в режиме Дизайнера.



Должен появиться обработчик chart1_Click() с пустым телом.



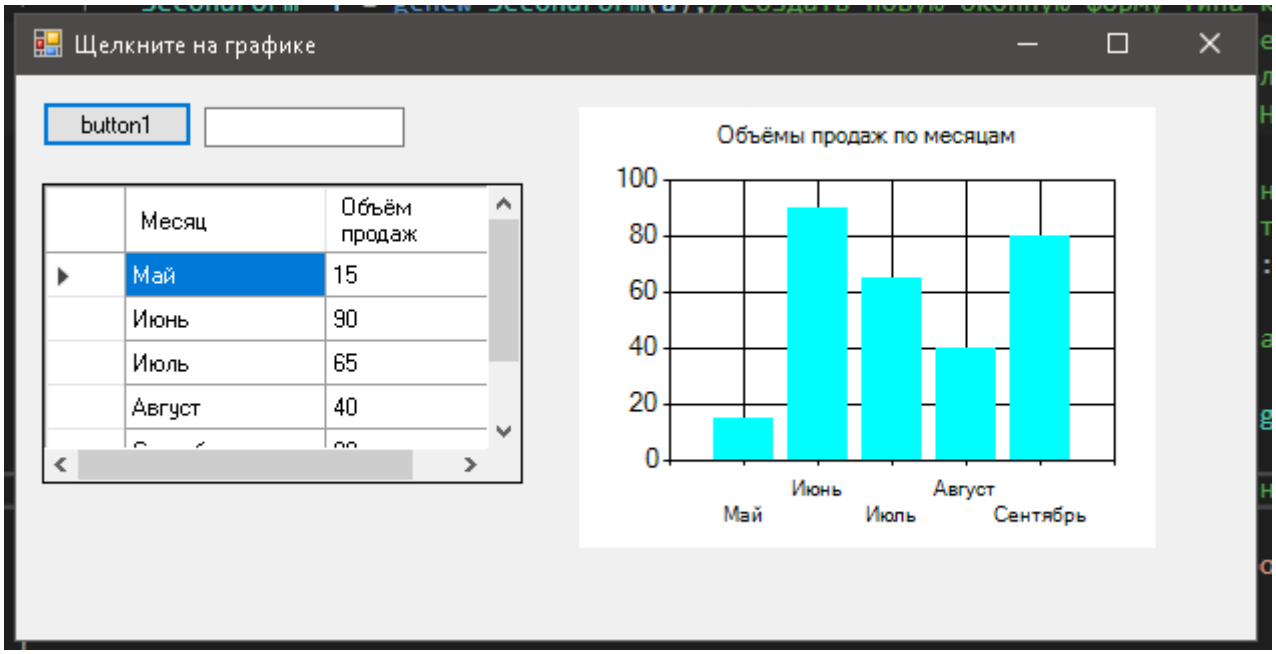
Дописываем код файла MyForm.h. Нам нужно хранить информацию, о текущем способе отображения диаграммы (двумерна или трехмерная), для чего создадим булевскую переменную-флаг. Она должна быть видна всем обработчикам, поэтому создадим ее в классе, но вне методов-обработчиков этого класса и закрытой от доступа извне данного класса.



Заполним обработчики MyForm_Load() и chart1_Click() кодом. Сделаем «стартовую» таблицу в оперативной памяти, которую программа будет отображать в виде визуальной таблицы и графика, но эти данные можно будет редактировать:

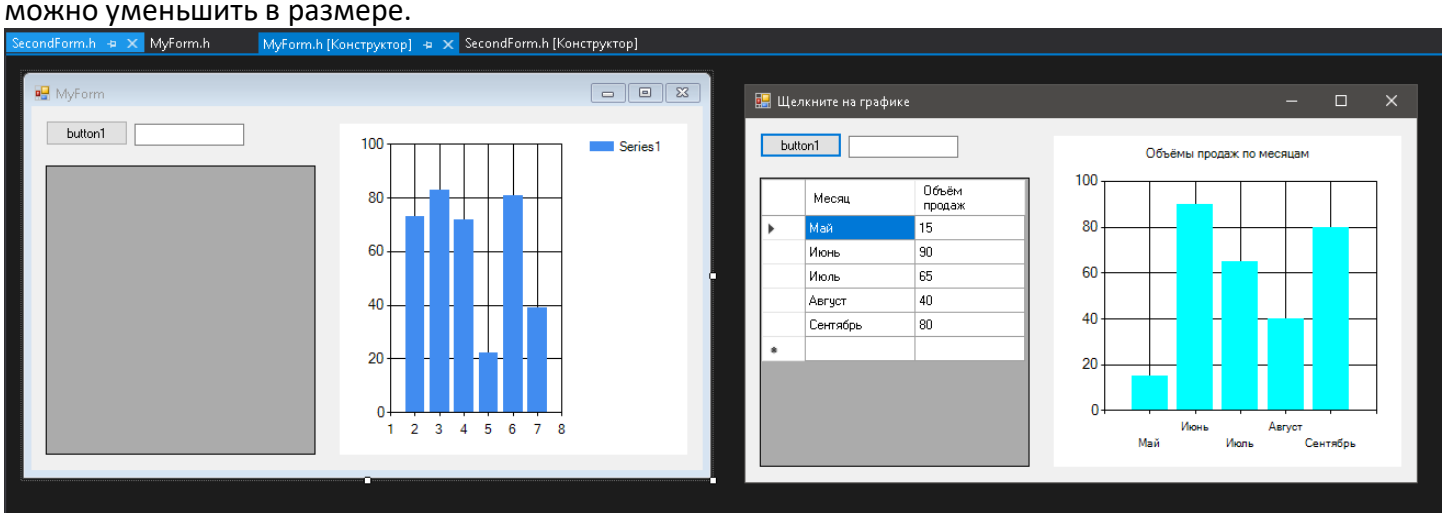
```
private: bool cilindr;//поле-флаг логического типа для хранения информации о текущем способе отрисовки диаграммы: трехмерная или двумерная
private: System::Void MyForm_Load(System::Object^ sender, System::EventArgs^ e)//метод-обработчик события загрузки оконной формы MyForm
{
    cilindr = false;//при загрузке оконной формы диаграмма принимает двумерный "плоский" вид
    this->Text = "Щелкните на графике";//надпись в заголовке окна
    DataTable^ table = gcnew DataTable();//создать управляемый объект типа ТаблицаДанных для хранения данных в оперативной памяти ПК
    table->Columns->Add("Месяц", String::typeid);//добавить в таблицу table столбец строкового типа с названием "Месяц"
    table->Columns->Add("Объём продаж", long::typeid);//добавить в таблицу table столбец длинного целочисленного типа с названием "Объём продаж"
    DataRow^ row = table->NewRow();//создать новую СтрокуДанных row в нашей таблице table
    row["Месяц"] = "Май";//в ячейку "Месяц" нашей строки row поместить константный массив символов "Май"
    row["Объём продаж"] = 15;//в ячейку "Объём продаж" нашей строки row поместить числовое значение 15
    table->Rows->Add(row);//в массив (коллекцию) строк таблицы table добавить строку row
    row = table->NewRow();//в таблице table создать новую строку, присвоив ее адрес указателю row, который уже свободен
    row["Месяц"] = "Июнь";
    row["Объём продаж"] = 90;
    table->Rows->Add(row);//в массив (коллекцию) строк таблицы table добавить еще строку row
    row = table->NewRow();//в таблице table создать новую строку, присвоив ее адрес указателю row, который уже свободен
    row["Месяц"] = "Июль";
    row["Объём продаж"] = 65;
    table->Rows->Add(row);//в массив (коллекцию) строк таблицы table добавить еще строку row
    row = table->NewRow();//в таблице table создать новую строку, присвоив ее адрес указателю row, который уже свободен
    row["Месяц"] = "Август";
    row["Объём продаж"] = 40;
    table->Rows->Add(row);//в массив (коллекцию) строк таблицы table добавить еще строку row
    row = table->NewRow();//в таблице table создать новую строку, присвоив ее адрес указателю row, который уже свободен
    row["Месяц"] = "Сентябрь";
    row["Объём продаж"] = 80;
    table->Rows->Add(row);//в массив (коллекцию) строк таблицы table добавить еще строку row
    chart1->DataSource = table;//заполненную данными таблицу table указываем в качестве источника данных для диаграммы chart1
    //на одном графике можно указать несколько зависимостей. Например, указывать месячные объёмы продаж за несколько лет, причем одинаковые по названию месяцы разных лет будут
    //стоять по одной линии, столбцы образуют трехмерный "лес"
    chart1->Series["Series1"]->XValueMember = "Месяц";//по горизонтальной оси откладываем названия месяцев
    chart1->Series["Series1"]->YValueMembers = "Объём продаж";//по вертикальной оси откладываем объёмы продаж
    chart1->Titles->Add("Объёмы продаж по месяцам");//инициализируем заголовок диаграммы
    chart1->Series["Series1"]->ChartType = System::Windows::Forms::DataVisualization::Charting::SeriesChartType::Column;//тип диаграммы - столбчатая гистограмма
    chart1->Series["Series1"]->Color = Color::Aqua;//цвет столбцов - Aqua из набора цветов статического класса Color
    chart1->Series["Series1"]->IsVisibleInLegend = false;//легенду на графике не отображать
    chart1->DataBind();//привязать диаграмму chart1 к источнику данных для нее (это наша таблица table)
    dataGridView1->DataSource = table;//источник данных для графической таблицы в оконной форме тоже таблица table из оперативной памяти
}
private: System::Void chart1_Click(System::Object^ sender, System::EventArgs^ e)//метод-обработчик события клика мышью по диаграмме chart1 в окне MyForm
{
    cilindr = !cilindr;//если пользователь нажал на диаграмму, то она должна изменить внешность, и значение переменной-флага надо изменить на противоположное
    if (cilindr == true)//если диаграмма отображается в виде двумерных линиц
    {
        chart1->Series["Series1"]["DrawingStyle"] = "Cylinder";//то присвоить ей трехмерный цилиндрический стиль
    }
    else//иначе
    {
        chart1->Series["Series1"]["DrawingStyle"] = "Default";//присвоить диаграмме стиль двумерных полос
    }
    chart1->DataBind();//обновить данные диаграммы в соответсвии с последними данными из таблицы table
}
};
}
```

Компилируем и тестируем. Приложение работает, несмотря на то, что в моем случае модуль подсказок IntelliSense показывает наличие двух ошибок, которые не влияют на работоспособность приложения.



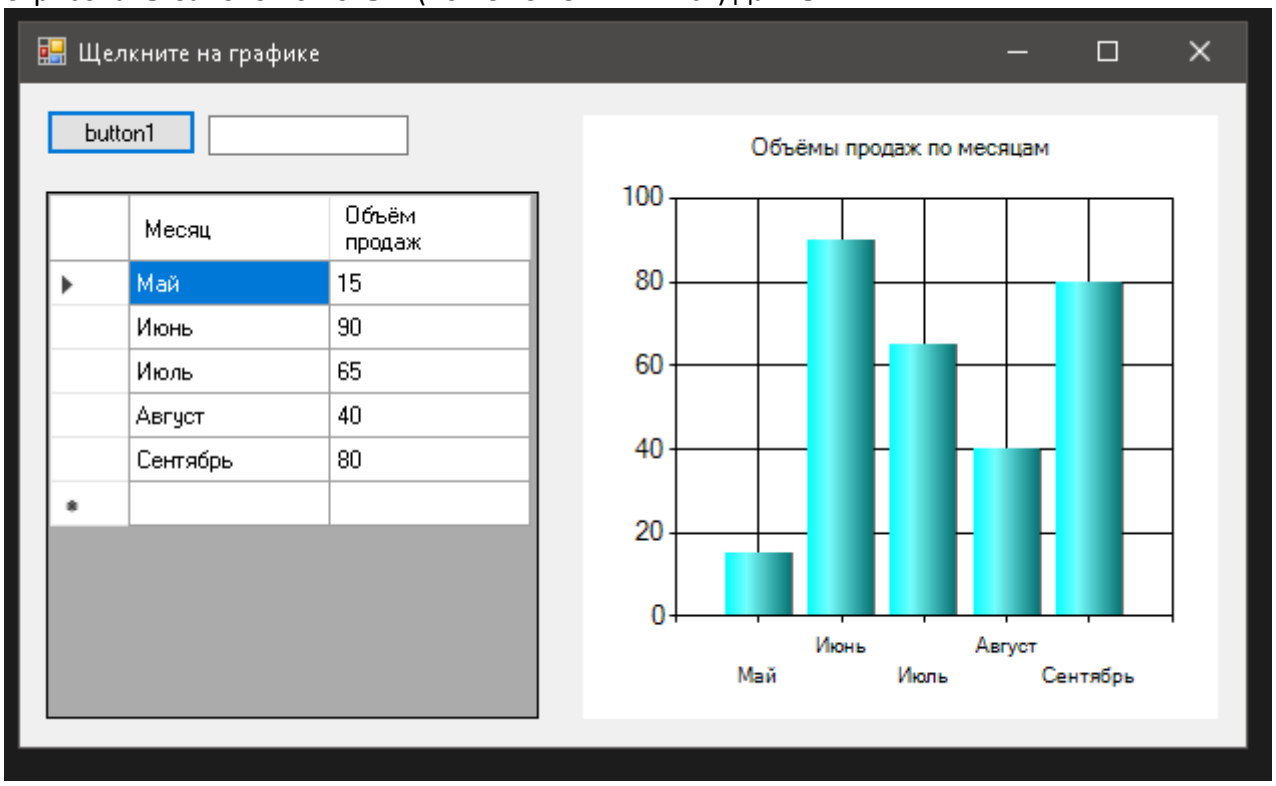
Окно приложения работает, но элемент DataGridView имеет по умолчанию недостаточную для нас площадь отображения, поэтому его окошко имеет специальные полосы («ползунки») прокрутки снизу и справа. В режиме графического Дизайнера расширим поле отображения элемента DataGridView

и сдвинем оба элемента, чтобы рационально использовать площадь окна нашего приложения, которое можно уменьшить в размере.

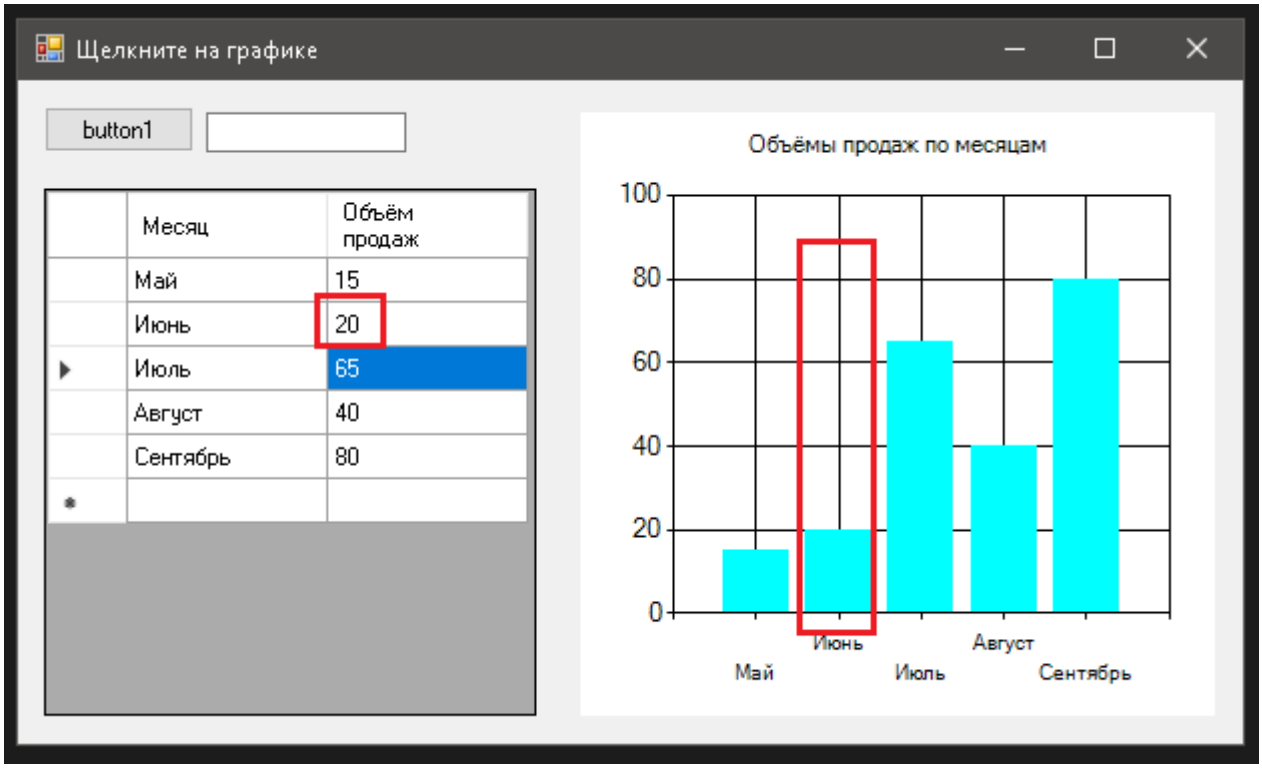


Итак, отображается таблица и график, построенный на основе данных таблицы. График имеет название, подписи элементов по оси иксов и автосгенерированные единицы измерения по оси игреков, причем для краткости отображения минимальный шаг по оси игреков составляет 20 единиц. Высота столбцов диаграммы соответствует данным, введенным нами в коде программы. Диаграмма отображается в простом двумерном виде.

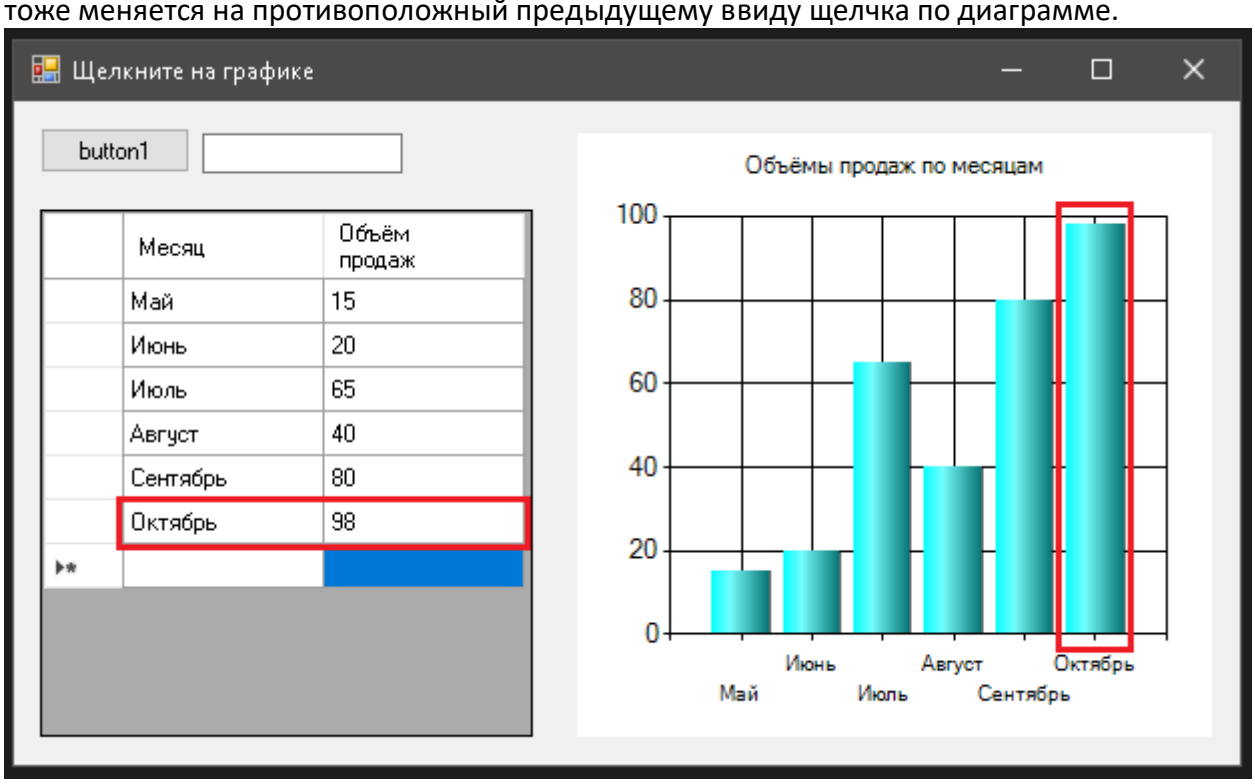
Кликнем по диаграмме мышью – визуальное отображение диаграммы должно измениться на трехмерное, столбцы становятся цилиндрическими, меняют цвет и получают градиентную окраску. Поскольку данные в таблице не изменились к данному моменту, то сама высота столбиков диаграммы и их количество остались прежними, хотя на самом деле был сделан запрос к таблице данных и столбики отрисованы заново по новым (но не изменившимся) данным.



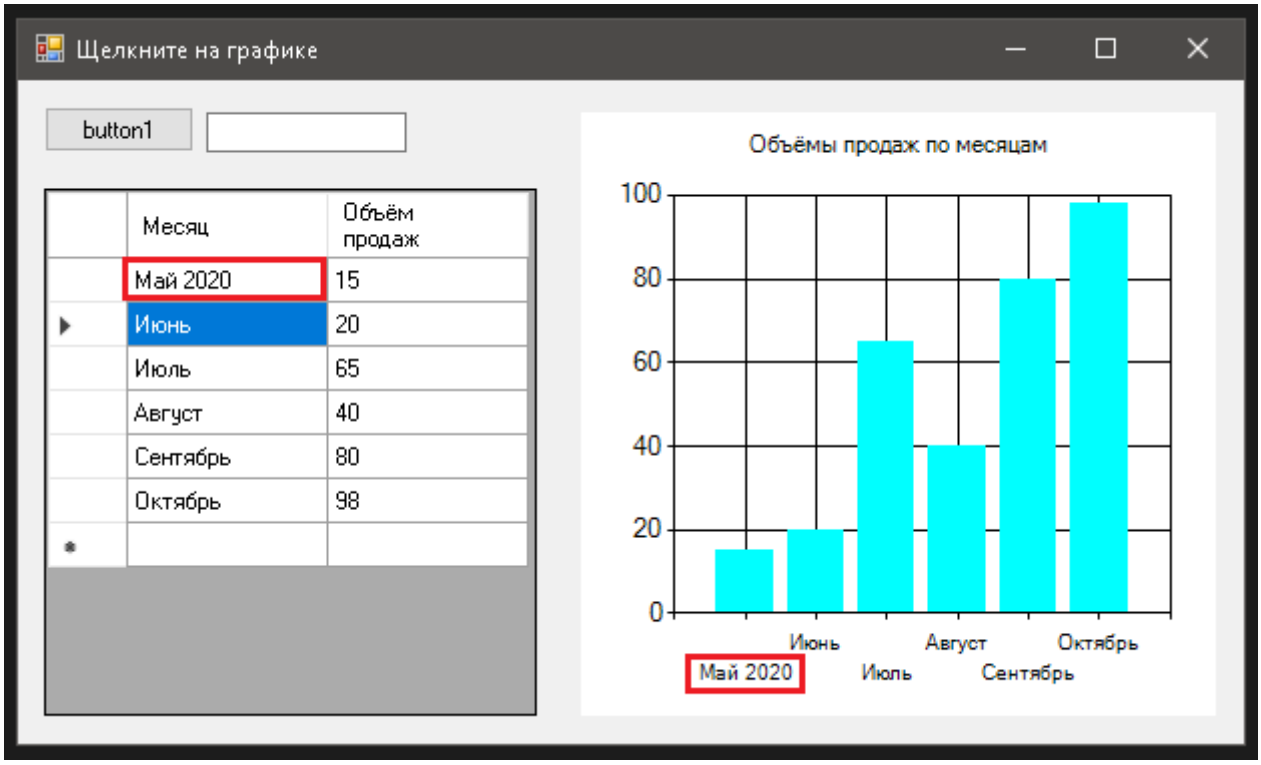
Поставим курсор в таблицу DataGridView и поменяем данные за июнь месяц с 90 на 20 и нажмем кнопку Enter на клавиатуре для вступления изменений в силу. После этого кликнем мышью по диаграмме. Изменились данные и в таблице, и в соответствующем столбце диаграммы, а поскольку мы кликали по диаграмме, то она снова поменяла свой стиль на изначальный двумерный.



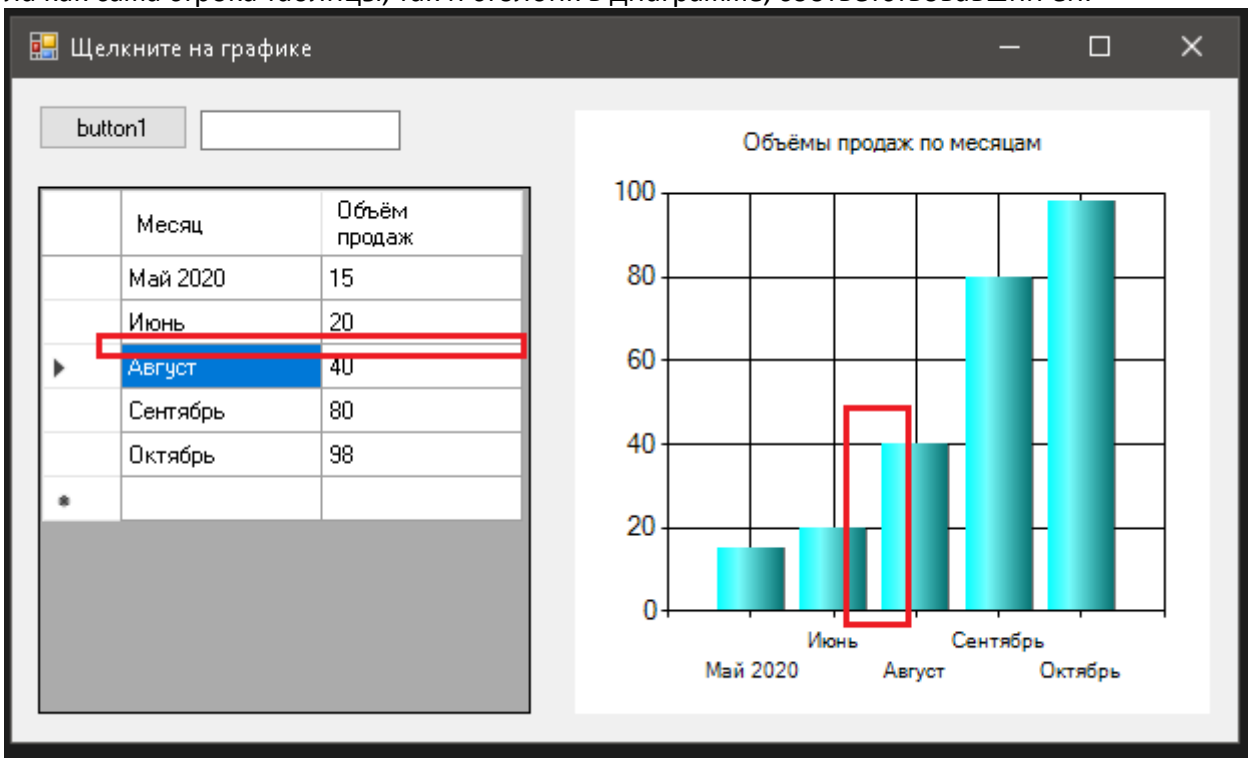
Поставим курсор в последнюю строку таблицы и добавим еще месяц Октябрь и его продажи 98 единиц. Нажмем Enter для вступления изменений в силу (данные теперь вносятся в таблицу table, находящуюся в оперативной памяти ПК) и кликнем по диаграмме для запроса данных и перерисовки диаграммы. Появляется новый столбец «Октябрь» высотой 98 единиц. Стиль оформления диаграммы тоже меняется на противоположный предыдущему ввиду щелчка по диаграмме.



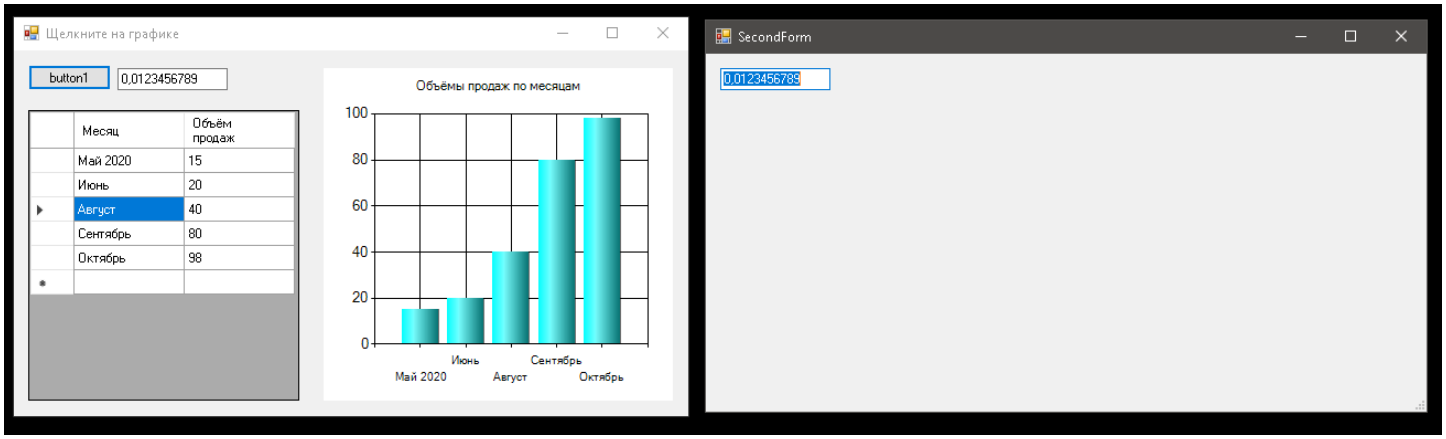
Изменим название месяца с «Май» на «Май 2020», нажмем Enter для вступления изменений в силу и кликнем по диаграмме для перерисовки. Изменения отображаются корректно.



Удалим из таблицы строку с данными за месяц Июль. Для этого поставим курсор слева от ячейки «Июль», чтобы выделилась вся строка, нажмем кнопку Delete и кликнем мышью по диаграмме. Исчезла как сама строка таблицы, так и столбик в диаграмме, соответствовавший ей.



Функционал отображения дополнительного окна, разумеется, должен сохраниться в нашем приложении:



2.Порядок выполнения работы

1. Изучить теоретические сведения к лабораторной работе.
2. Разработать на языке C++ программу вывода на экран решения задачи в соответствии с вариантом индивидуального задания, указанным преподавателем.
3. Отлаженную, работающую программу сдать преподавателю. Работу программы показать с помощью самостоятельно разработанных тестов.
4. Ответить на контрольные вопросы.

Задание 2

Выполнить задания по вариантам.

Создать новый проект со вспомогательным окном, в котором будет отображаться таблица с двумя столбцами и диаграммой, строящейся по данным из таблицы. Диаграмма должна перерисовываться по клику по ней. Обеспечить возможность ввода пользователем данных в таблицу, изменение уже существующих данных либо удаление строк из таблицы и перерисовку диаграммы по новым данным таблицы по клику на диаграмме или кнопке. Диаграмма должна иметь свое название. При запуске приложения в таблице должно быть не менее 4-х строк данных и построенная по ним диаграмма с возможностью редактирования. Предусмотреть отображение фамилии разработчика программы.

№	Описание столбцов таблицы
1	Таблица состоит из столбцов: Месяц (символьный массив) и Средняя температура, в градусах (вещественное число).
2	Таблица состоит из столбцов: Название автомобиля (символьный массив) и Средний расход топлива, в литрах на километр (вещественное число).
3	Таблица состоит из столбцов: Название процессора (символьный массив) и его пиковая Тактовая частота, в гигагерцах (вещественное число).
4	Таблица состоит из столбцов: Название химического вещества (символьный массив) и его Плотность, килограмм на метр кубический (вещественное число).
5	Таблица состоит из столбцов: Название автомобиля (символьный массив) и его Масса в тоннах (вещественное число).
6	Таблица состоит из столбцов: Год (целое число) и Количество вооруженных конфликтов в течение его (в тысячах) (вещественное число).
7	Таблица состоит из столбцов: Название растения (символьный массив) и его Максимальная высота в метрах (вещественное число).
8	Таблица состоит из столбцов: Название металла (символьный массив) и его Цена за тонну (в долларах) (вещественное число).
9	Таблица состоит из столбцов: Название продукта питания (символьный массив) и его Цена в рублях (вещественное число).
10	Таблица состоит из столбцов: Адрес квартиры (символьный массив) и ее Площадь в метрах (вещественное число).
11	Таблица состоит из столбцов: Название горы (символьный массив) и ее Высота в километрах (вещественное число).
12	Таблица состоит из столбцов: ФИО человека (символьный массив) и его Рост в метрах (вещественное число).
13	Таблица состоит из столбцов: Должность (символьный массив) и Ставка оплаты труда в рублях (вещественное число).

14	Таблица состоит из столбцов: Название болезни (символьный массив) и Среднее время выздоровления (в сутках) (вещественное число).
15	Таблица состоит из столбцов: Страна (символьный массив) и Количество населения в миллионах человек (вещественное число).
16	Таблица состоит из столбцов: Название самолета (символьный массив) и его Цена в миллионах долларов (вещественное число).
17	Таблица состоит из столбцов: Название страны (символьный массив) и ее Валовой внутренний продукт на душу населения (в долларах) (вещественное число).
18	Таблица состоит из столбцов: ФИО человека (символьный массив) и его Вес в килограммах (вещественное число).
19	Таблица состоит из столбцов: Название песни (символьный массив) и ее Продолжительность в минутах (вещественное число).
20	Таблица состоит из столбцов: Название ноутбука (символьный массив) и Длина его диагонали в дюймах (вещественное число).
21	Таблица состоит из столбцов: Название фирмы (символьный массив) и ее Годовой оборот в миллионах долларов (вещественное число).
22	Таблица состоит из столбцов: Название животного (символьный массив) и его Максимальная скорость в беге (километров в час) (вещественное число).
23	Таблица состоит из столбцов: Название смартфона (символьный массив) и его Вес в килограммах (вещественное число).
24	Таблица состоит из столбцов: ФИО студента (символьный массив) и его Средний балл за семестр (вещественное число).
25	Таблица состоит из столбцов: Название страны (символьный массив) и ее Среднее количество детей в семье (вещественное число).
26	Таблица состоит из столбцов: Название города (символьный массив) и Средняя этажность зданий в нем (вещественное число).
27	Таблица состоит из столбцов: Название автомобиля (символьный массив) и Время его разгона до скорости 100 км/час, в секундах (вещественное число).
28	Таблица состоит из столбцов: Название страны (символьный массив) и Средняя продолжительность жизни ее населения, в годах (вещественное число).
29	Таблица состоит из столбцов: Название продукта питания (символьный массив) и его Средний срок хранения, в месяцах (вещественное число).
30	Таблица состоит из столбцов: Название группы студентов (символьный массив) и Средний балл студентов группы (вещественное число).

Задание 3

(выполнить свое задание из 30-и вариантов)

№	*Формула f(x)*	*Диапазон для x*	№	*Формула f(x)*	*Диапазон для x*
1	$\frac{4x^2+5}{4x+8}$	[-5, 0]	16	$\frac{21-x^2}{7x-9}$	[0, 5]
2	$\frac{17-x^2}{4x-5}$	[0, 5]	17	$\frac{2x^2-7}{\sqrt{3x^2-2}}$	[0, 5]
3	$\frac{x^2-3}{\sqrt{4x^2-3}}$	[0, 5]	18	$\frac{2x^3-3x^2-2x+1}{3x^2-1}$	[0, 5]
4	$\frac{x^3-4x}{3x^2-4}$	[0, 5]	19	$\frac{x^2-11}{4x-3}$	[0, 5]
5	$\frac{4x^3+3x^2-8x-2}{2-3x^2}$	[0, 5]	20	$\frac{2x^2-9}{\sqrt{x^2-1}}$	[1, 6]
6	$\frac{x^2-3}{\sqrt{3x^2-2}}$	[0, 5]	21	$\frac{4x^3+3x^2-2x-2}{x^2-1}$	[0, 5]
7	$\frac{2x^2-6}{x-2}$	[0, 5]	22	$\frac{1-x^2}{\sqrt{16x^2-9}}$	[0, 5]

8	$\frac{x^3+x^2-3x-1}{x^2-1}$	[0, 5]	23	$\frac{2x^2-3x+1}{1-2x}$	[0,5]
9	$\frac{4x^3-3x}{4x^2-1}$	[0, 5]	24	$\frac{4x^3+x^2-2x-1}{2x^2-1}$	[0, 5]
10	$\frac{x^2-6x+4}{3x-2}$	[0, 5]	25	$\frac{5x^2-3}{\sqrt{3x^2-1}}$	[0, 5]
11	$\frac{2-x^2}{\sqrt{9x^2-4}}$	[0, 5]	26	$\frac{4x^3-x}{x^2-1}$	[0, 5]
12	$\frac{x^3+3x^2-2x-2}{2-3x^2}$	[0, 5]	27	$\frac{2x^3-2x+1}{x^2-1}$	[0,5]
13	$\frac{3x^2-7}{2x+1}$	[-5, 0]	28	$\frac{x^2-5}{\sqrt{x^2-2}}$	[2, 7]
14	$\frac{x^2-5}{\sqrt{9x^2-8}}$	[0, 5]	29	$\frac{2x^2-5}{\sqrt{3x^2-4}}$	[0, 5]
15	$\frac{x^2-6x+4}{2-2x}$	[0, 5]	30	$\frac{15-x^3}{2x-1}$	[0, 5]

Задание 4

Найти на панели графических элементов Windows Forms 5 элементов, попробовать применить их в окне своего приложения и на защите объяснить, для чего предназначены эти визуальные пользовательские элементы.

3. Контрольные вопросы

1. Как создать обработчик события?
2. Как добавить новую форму в проект приложения и как её вызывать?
3. Где отображаются свойства формы и каким способом можно определить их назначение?
4. Чем отличаются модальное и немодальное окна в приложении?
5. Как сделать целую оконную форму или отдельный графический элемент в окне недоступным для изменения пользователем посредством мыши и клавиатуры?
6. Есть ли разница, в каком порядке располагать отлавливатели ошибок?

Литература

Зиборов, В. MS Visual C ++2010 в среде NET (Библиотека программиста)/В. Зиборов.— СПб.: Питер, 2012.

Преподаватель Белокопыцкая Ю.А.

Рассмотрено на заседании цикловой комиссии ПОИТ № 10
Протокол № ____ от «___» _____ 2018 г.
Председатель ЦК_____