

Частное учреждение образования
«Колледж бизнеса и права»

УТВЕРЖДАЮ

Заведующий методическим кабинетом

_____ Е.В. Фалей

« _____ » _____ 2017 г.

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

Лабораторная работа № 28
Инструкционно-технологическая карта

Тема: Ввод/вывод информации с использованием текстовых компонентов (окон и элементов Windows Forms).

Цель: Научиться создавать алгоритмы и программы с использованием текстовых компонентов.

Время выполнения: 2 часа

1. Краткие теоретические сведения

До этого мы создавали программы в консоли. Даже в этом случае наша консоль отображалась в окне Windows, поскольку имела рамку, строку заголовка, вверху справа кнопки Свернуть, Развернуть и Закрыть. На самом деле в современной операционной системе MS Windows работает эмуляция консоли, в то время как в первых компьютерах консоль занимала весь экран и не имела рамки, кнопок Закрыть, Свернуть и Развернуть. Консольный и оконный графические интерфейсы взаимодействия с пользователем различаются, а вот алгоритмы в программах в любом GUI (Graphical User Interface – Графический интерфейс пользователя) одинаковые, то есть и консольное, и оконное приложения рассчитывают факториал числа одинаково (можно написать одну функцию или класс с методом, который будет высчитывать факториал целого числа на регистрах процессора), но отображают информацию по-разному. Для быстрого написания и тестирования кода лучше подходит создание консольного приложения (его быстрее создать), а для написания программы для обычного рядового пользователя лучше подойдет оконное приложение, но создавать его будет дольше за счет размещения различных графических элементов, их оформления. Если нужно пересоздать оконное приложение, то придется снова размещать графические элементы, создавать их обработчики (обработчики нажатия по ним мышью пользователя, наведения на них курсора мыши и т.д.) и заполнять их кодом, в то время как консольное приложение требует создания одного или нескольких файлов и копирования в них кода из «старого» проекта.

Windows Forms — это набор средств для создания оконных windows-приложений, выполняющихся в среде CLR (Common Language Runtime — общезыковая исполняющая среда). Форма и используемые с ней элементы управления представлены классом C++/CLI (Common Language Intermediate). Каждый класс обладает набором свойств, которые определяют поведение и внешний вид элемента управления или формы. При создании проекта приложения создается как окно приложения Windows Forms, построенное на основе класса Form (являющееся экземпляром класса Form), так и весь код, обеспечивающий отображение этого окна приложения. После создания проекта Windows Forms разработка приложения сводится к выполнению четырех отдельных операций:

1. Интерактивное создание графического интерфейса пользователя на вкладке Form Конструктор (Конструктор формы), отображаемой в панели Редактор (Редактор), путем выбора элементов управления в окне Панель элементов и их помещения на оконную форму мышкой. На этом этапе можно также создавать дополнительные окна форм, то есть можно сделать так, что, например, при нажатии пользователем по кнопке, появится новое окно приложения с другими

элементами.

2. Изменение свойств элементов управления и форм в окне Свойства в соответствии с потребностями приложения.

3. Обработчики событий щелчков для элементов управления можно создавать, дважды щелкая на элементе управления на вкладке Form Design (Дизайнер формы). В окне Properties (Свойства) элемента управления в качестве его обработчика события можно также определять существующую функцию (метод).

4. Для удовлетворения потребностей приложения можно изменять и расширять классы, автоматически создаваемые в результате взаимодействия с вкладкой Form Design.

Компиляция проекта приводит к созданию новой сборки, код которой относится к пространству имен, совпадающему с именем проекта. Пространство имен позволяет различать типы с одинаковыми именами в различных сборках, поскольку каждое имя типа уточняется именем конкретного пространства имен.

Существуют шесть рекомендаций к использованию пространств имен библиотеки .NET, охватывающих функциональные возможности, которые, скорее всего, потребуются в приложениях (см. таблицу 1).

Таблица 1. Пространство имен фреймворка (библиотеки библиотек) .NET

Пространство имен	Содержимое
System	Содержит классы, которые определяют типы данных, используемые во всех приложениях CLR. Оно содержит также классы событий и обработчиков событий, исключения и классы, поддерживающие функции общего применения.
System::ComponentModel	Содержит классы, которые поддерживают работу компонентов графического интерфейса пользователя в приложениях CLR
System::Collections	Содержит классы коллекций, предназначенные для всевозможной организации данных, в том числе классы определения списков, очередей и стеков
System::Windows::Forms	Содержит классы, которые поддерживают использование в приложении средств Windows Forms
System::Data	Содержит классы, которые поддерживают набор компонентов ADO.NET, используемый для доступа и обновления источников данных
System:: Drawing	Содержит классы, которые поддерживают основные графические операции, подобные рисованию на форме или компоненте

Класс MyForm — производный от класса Form, который определен в пространстве имен System::Windows::Forms. Класс Form представляет окно приложения или диалогового окна, а класс MyForm, который определяет окно для пространства имен Lab73, наследует все члены класса Form.

Раздел в конце класса MyForm содержит определение функции InitializeComponent(). Эта функция вызывается конструктором для определения окна приложения и любых компонентов, добавляемых в форму.

Вначале components класса MyForm унаследован от базового класса, и его задача — отслеживание компонентов, постепенно добавляемых в форму. Первый оператор сохраняет в члене components дескриптор объекта Container, представляющего коллекцию, которая хранит компоненты графического интерфейса пользователя в списке. Каждый новый компонент, добавляемый в форму с помощью средств Form Design, добавляется в этот объект Container.

Остальные операторы функции InitializeComponent() определяют свойства объекта MyForm. Ни одно из этих свойств не следует изменять непосредственно в коде, но их значения можно выбирать посредством окна Свойства (Properties).

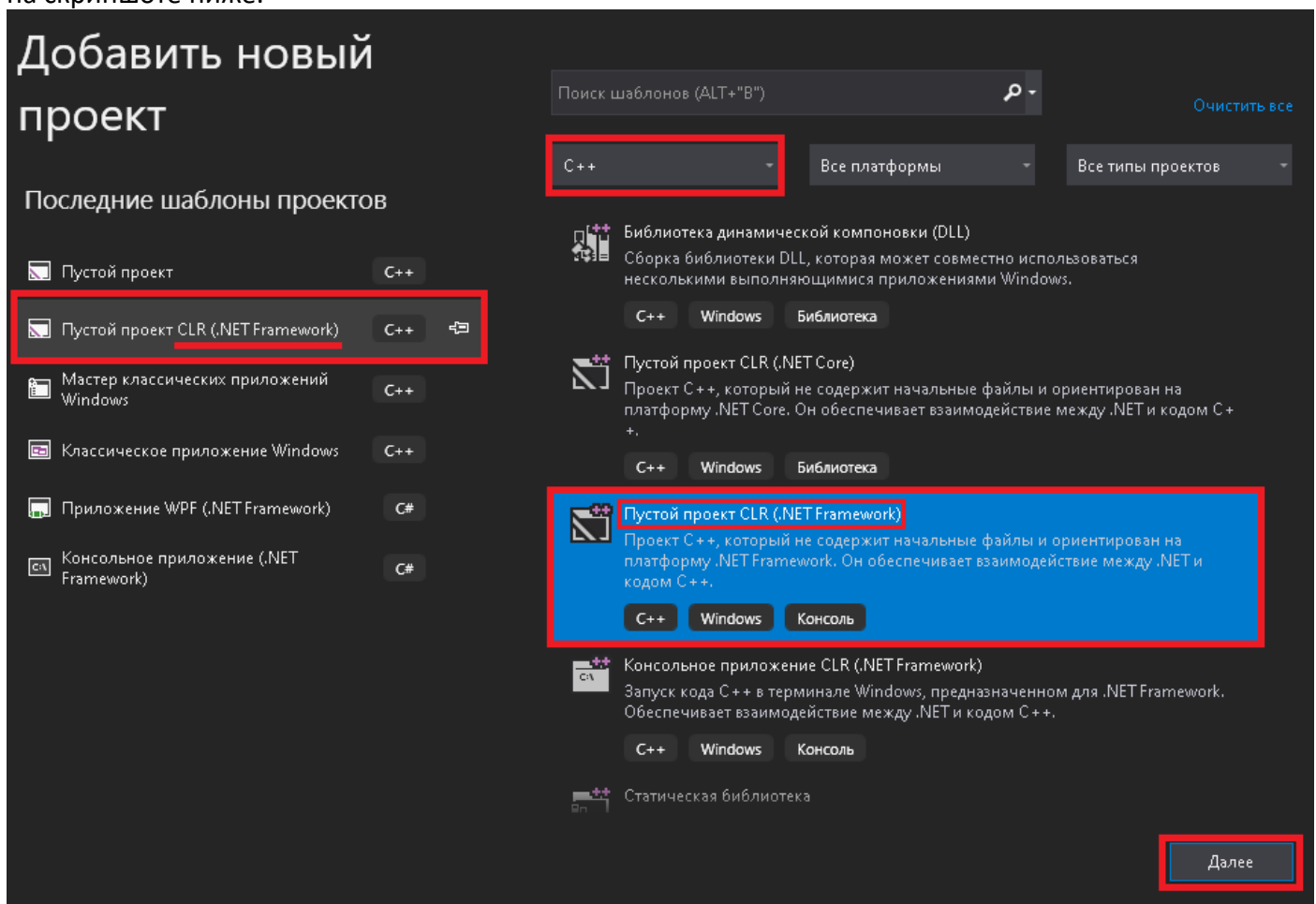
Если переключится обратно на вкладку Form1.h[Конструктор] в окне редактора и щелкнуть правой клавишей мыши, выбрать Свойства, то вы увидите окно свойств формы (один из способов открыть окно Свойств).

Обратите внимание, что здесь активно используются понятия объектно-ориентированного программирования. Указываются классы `Color`, `SystemColors` и используется `this` — указатель на текущий активный объект (в данном случае на форму).

Оконные приложения в различных версиях MS Visual Studio создаются принципиально одинаково, но некоторые действия различаются. Создавать оконные приложения WindowsForms можно начиная с MS Visual Studio 2013, которой требуется минимум ОС MS Windows 7 с установленным Сервисным пакетом SP1. Дальнейшие версии MS Visual Studio также поддерживают создание оконных WindowsForms приложений. Рассмотрим создание оконного приложения в бесплатной среде разработки MS Visual Studio 2019 Community Edition. Нам может понадобиться наличие Интернет-соединения для установки нужных пакетов (библиотек).

Задание 1







Запускаем MS Visual Studio 2019 и создаем новый проект, нажимая правой кнопкой мыши по имени решения в окне Обозреватель решений, далее Добавить/Создать проект/Пустой проект CLR (.NET Framework) [C++, консольный]/Далее. Если у вас уже установлен нужный шаблон, то будет как на скриншоте ниже.



Но если у вас не установлен нужный шаблон, то вы не сможете среди шаблонов C++ найти такого типа проект и вам нужно скачать и установить себе этот шаблон следующим образом. Прокрутите ползунок выбора доступных проектов в самый низ, чтобы увидеть кнопку Установка других средств и компонентов, нажмите ее.

Добавить новый проект

Последние шаблоны проектов

-  Пустой проект C++
-  Пустой проект CLR (.NET Framework) C++
-  Мастер классических приложений Windows C++
-  Классическое приложение Windows C++
-  Приложение WPF (.NET Framework) C#
-  Консольное приложение (.NET Framework) C#

 Очистить все

C++ Все платформы Все типы проектов



Проект машинного модульного теста

Написание модульных тестов C++ с помощью собственной платформы Microsoft CppUnitTest.

C++ Windows Тестирование



Google Test

Write C++ unit tests using Google Test. Includes a copy of the Google Test library for use.

C++ Windows Тестирование



Библиотека классов CLR (.NET Core)

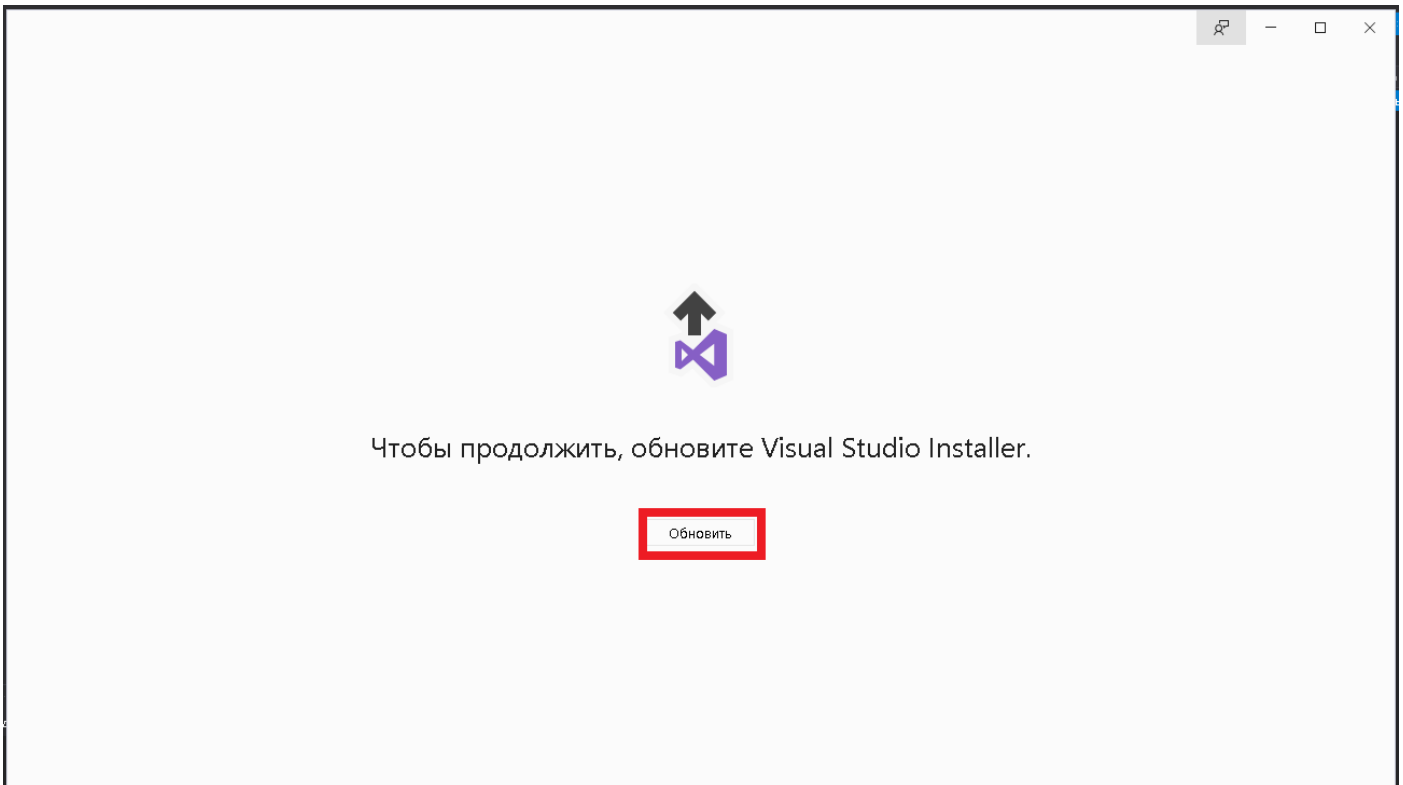
Библиотека C++, предназначенная для .NET Core. Обеспечивает взаимодействие между .NET и кодом C++.

C++ Windows Библиотека

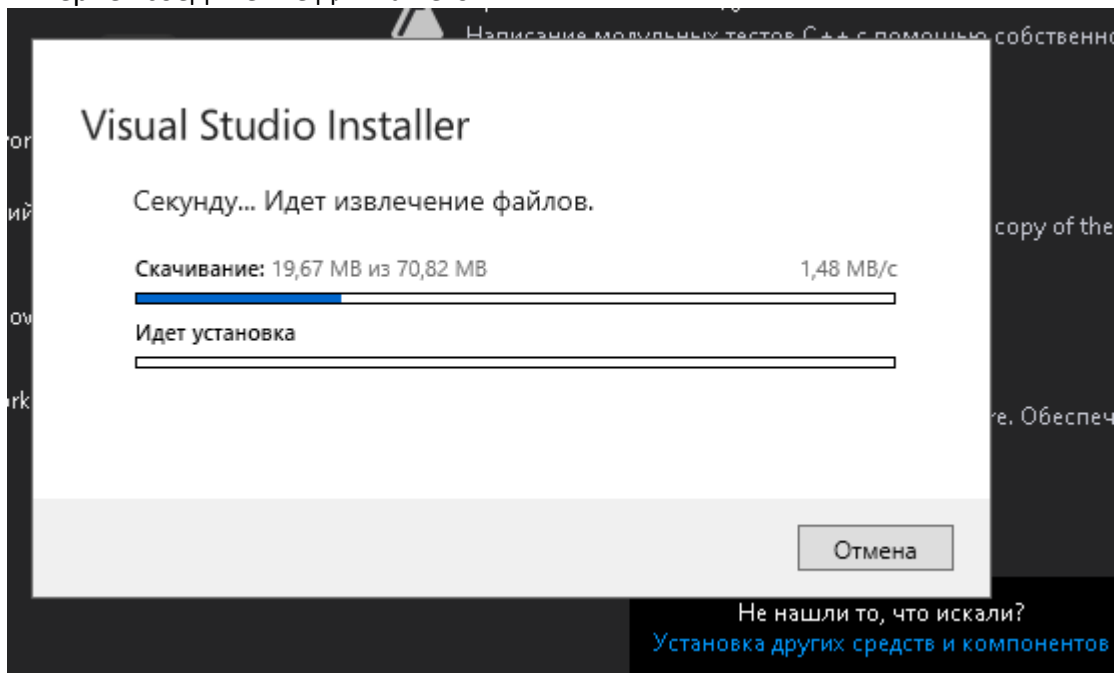
Не нашли то, что искали?
[Установка других средств и компонентов](#)

Далее

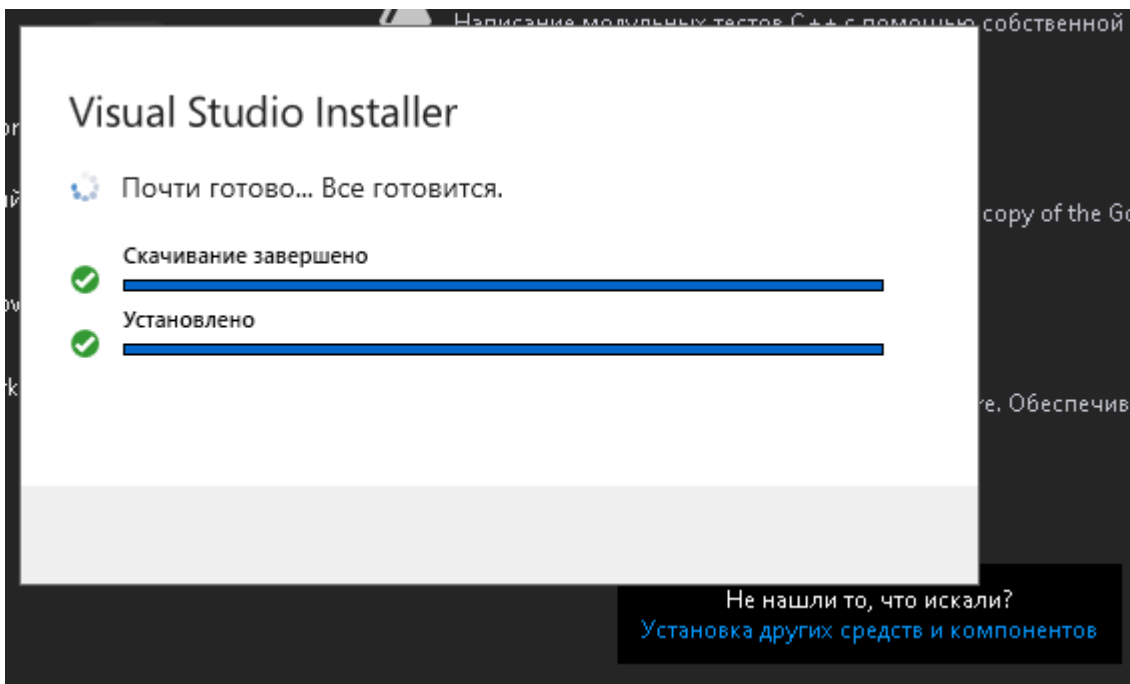
Может появиться окно-переспрос “Разрешить этому приложению вносить изменения на вашем компьютере?”. Проверяем, что это “Проверенный издатель: Microsoft Corporation” (программы ПИШУТСЯ программистами, поэтому фирма-производитель программ является фактически ИЗДАТЕЛЕМ ТЕКСТОВ программ) и жмем Да. Появится окно Visual Studio Installer (Установщик ВизуалСтудии), с помощью которого можно доустанавливать в MS VS новые компоненты, обновлять ее. В моем случае Установщик VS сначала сам хочет обновиться, соглашаемся.



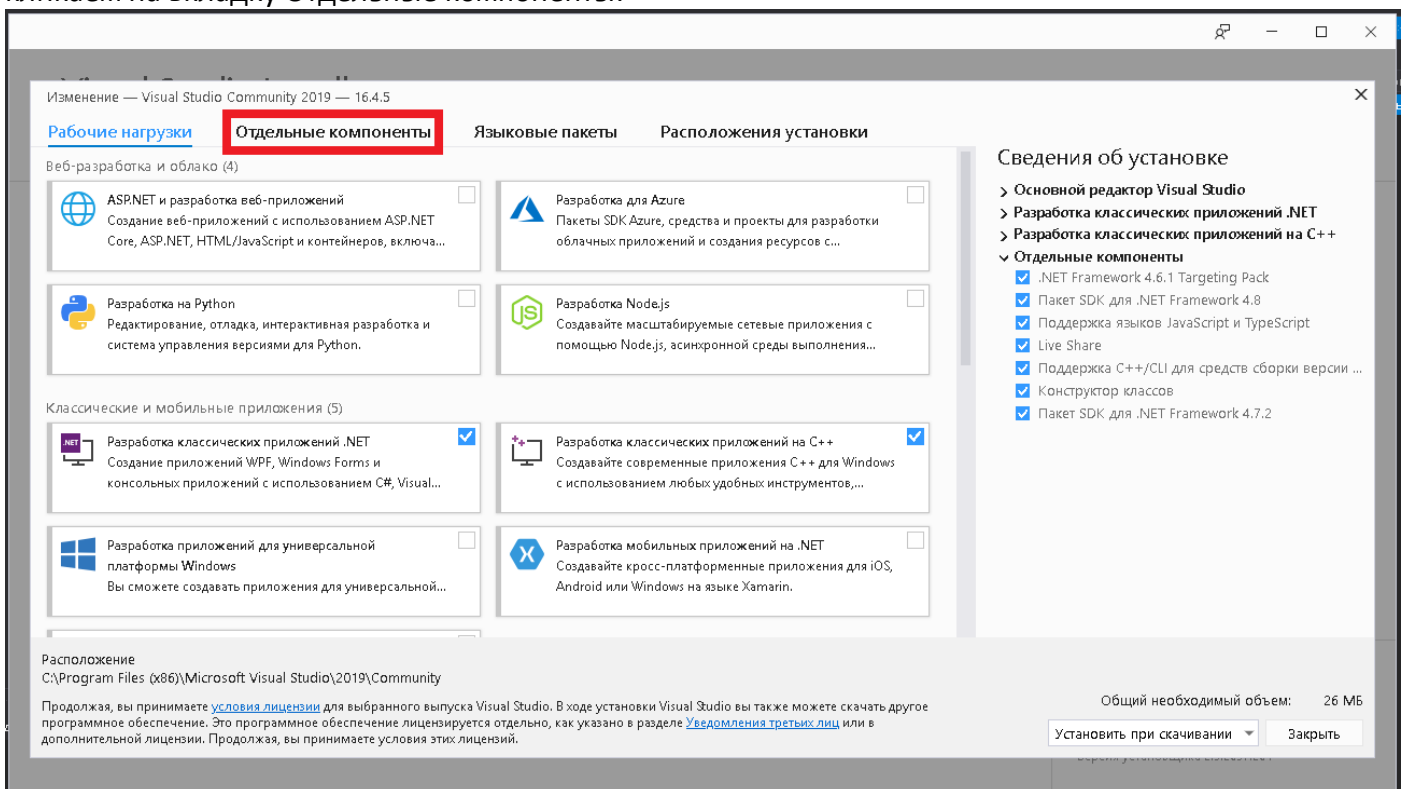
Ждем скачивания и установки компонентов, не прерываем процесс. Уже тут требуется Интернет-соединение для вашего ПК.



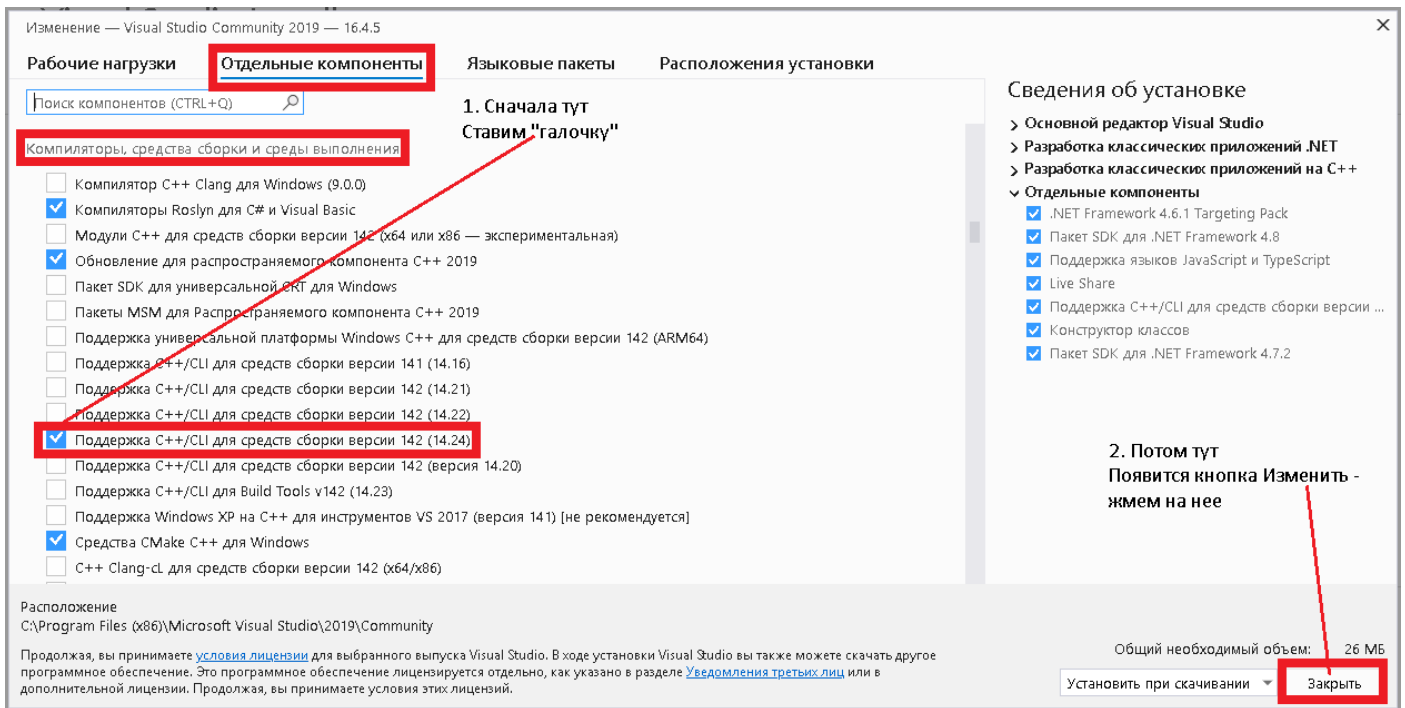
Процесс обычно сам завершается, не прерываем его:



По окончании обновления Установщика VS2019 видим его запущенное окно, в котором кликаем на вкладку Отдельные компоненты:

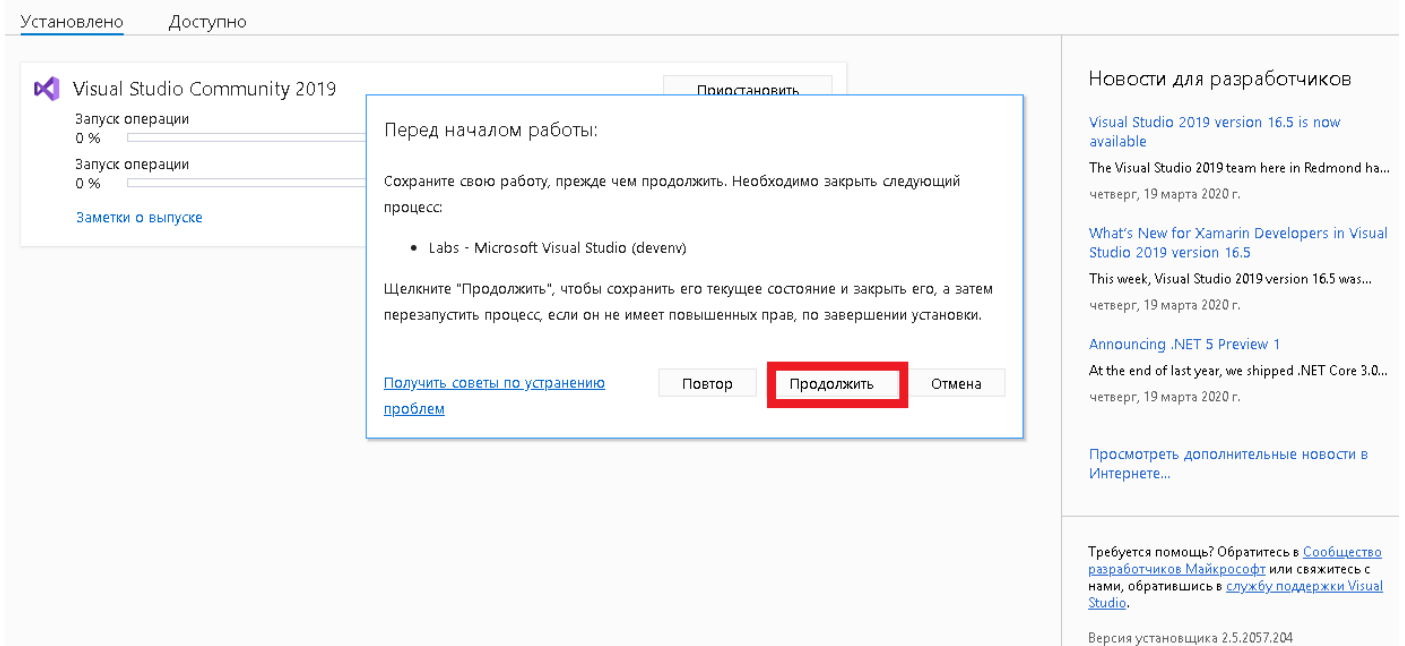


Открывается вкладка Отдельные компоненты, в которой «галочками» отображаются уже установленные компоненты и без галочек – те компоненты, которые можно скачать и установить. Мы ищем раздел «Компиляторы, средства сборки и среды выполнения», в котором ищем компонент «Поддержка C++/CLI для средств сборки...» и ставим перед ним «галочку». Если таких строк будет несколько, выбираем одну самую новую версию (ее номер будет больше остальных). После этого внизу справа кнопка Закрывать должна поменяться на кнопку Изменить. Жмем на кнопку Изменить и ждем скачивания и установки нового компонента.



Если в процессе обновления компонента Установщик будет запрашивать у вас разрешение на скачивание, закрытие MS VS2019 и т.д. – соглашаемся.

Visual Studio Installer



Когда MS VS2019 обновится и перезапустится можно снова создавать Пустой проект CLR (.NET Framework) [C++, консольный], который теперь уже должен быть доступен для создания.

Добавить новый проект

Последние шаблоны проектов

- Пустой проект C++
- Пустой проект CLR (.NET Framework) C++**
- Мастер классических приложений Windows C++
- Классическое приложение Windows C++
- Приложение WPF (.NET Framework) C#
- Консольное приложение (.NET Framework) C#

Поиск шаблонов (ALT+"B")

Очистить все

C++

Все платформы

Все типы проектов



Библиотека динамической компоновки (DLL)

Сборка библиотеки DLL, которая может совместно использоваться несколькими выполняющимися приложениями Windows.

C++

Windows

Библиотека



Пустой проект CLR (.NET Core)

Проект C++, который не содержит начальные файлы и ориентирован на платформу .NET Core. Он обеспечивает взаимодействие между .NET и кодом C++.

C++

Windows

Библиотека



Пустой проект CLR (.NET Framework)

Проект C++, который не содержит начальные файлы и ориентирован на платформу .NET Framework. Он обеспечивает взаимодействие между .NET и кодом C++.

C++

Windows

Консоль



Консольное приложение CLR (.NET Framework)

Запуск кода C++ в терминале Windows, предназначенном для .NET Framework. Обеспечивает взаимодействие между .NET и кодом C++.

C++

Windows

Консоль



Статическая библиотека

Далее

Даем проекту краткое имя латинскими буквами, проверяем место его сохранения (на диске D:\) и жмем кнопку Создать:

Настроить новый проект

Пустой проект CLR (.NET Framework) C++ Windows Консоль

Имя проекта

Lab7_3

Расположение

D:\2019\Labs

Платформа

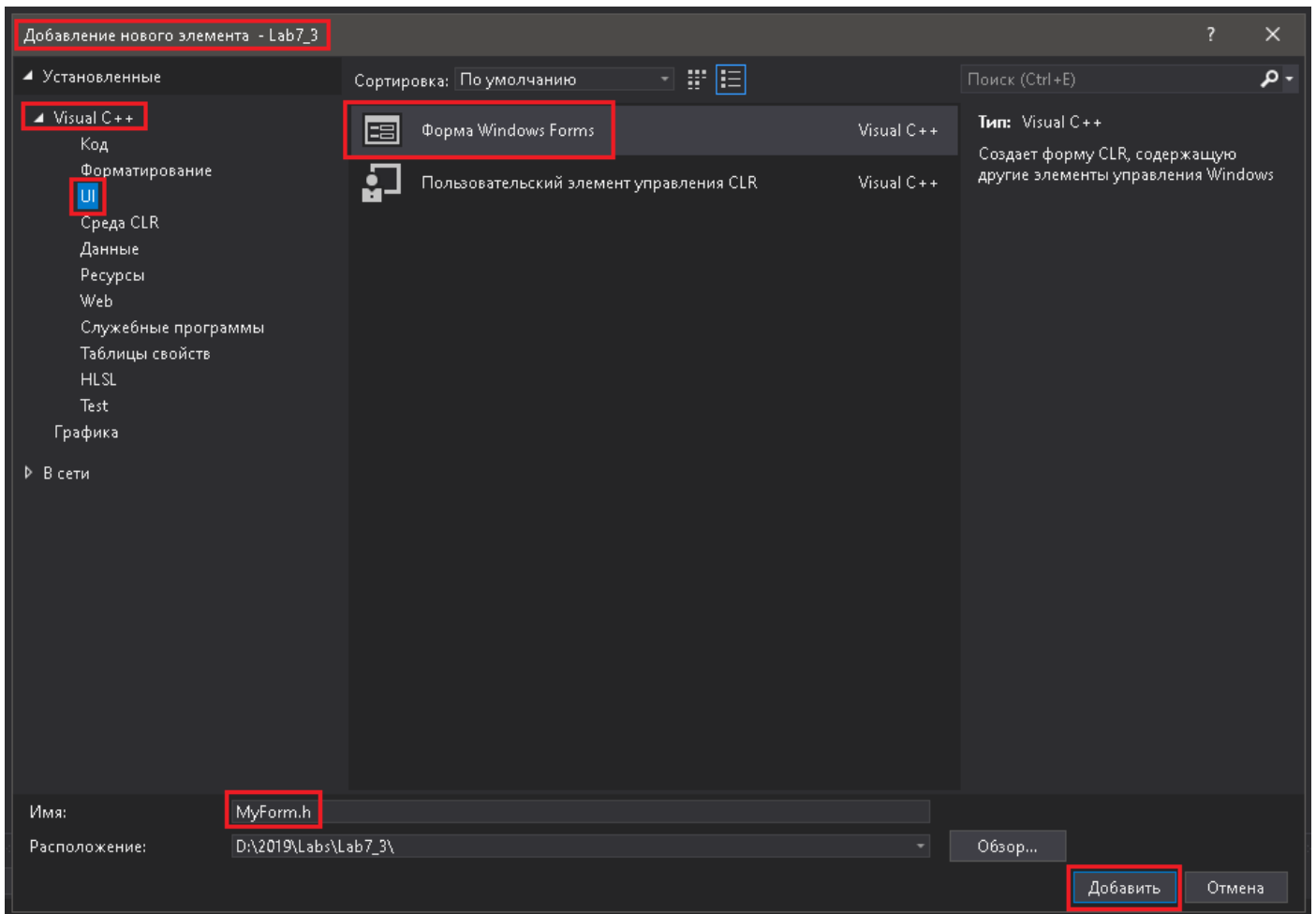
.NET Framework 4.7.2

Назад

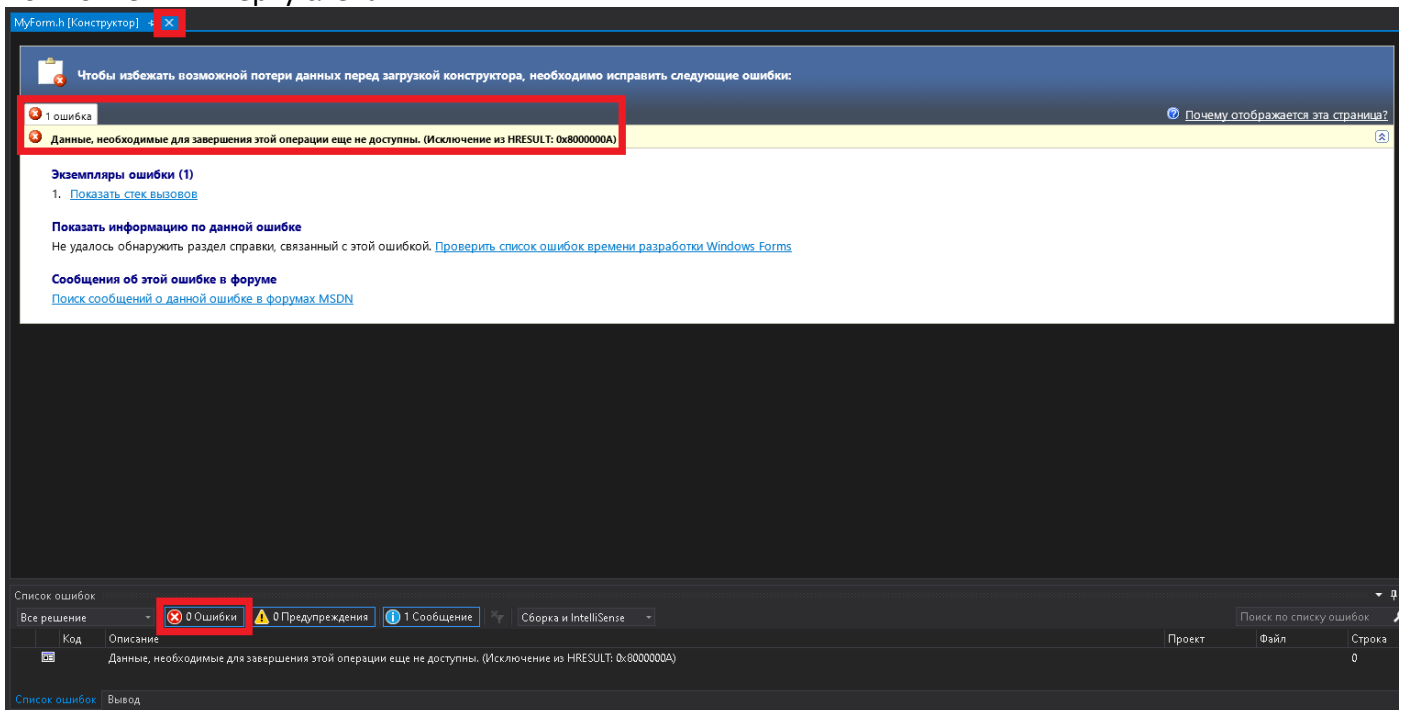
Создать

Когда проект создан, назначаем его автозагружаемым.

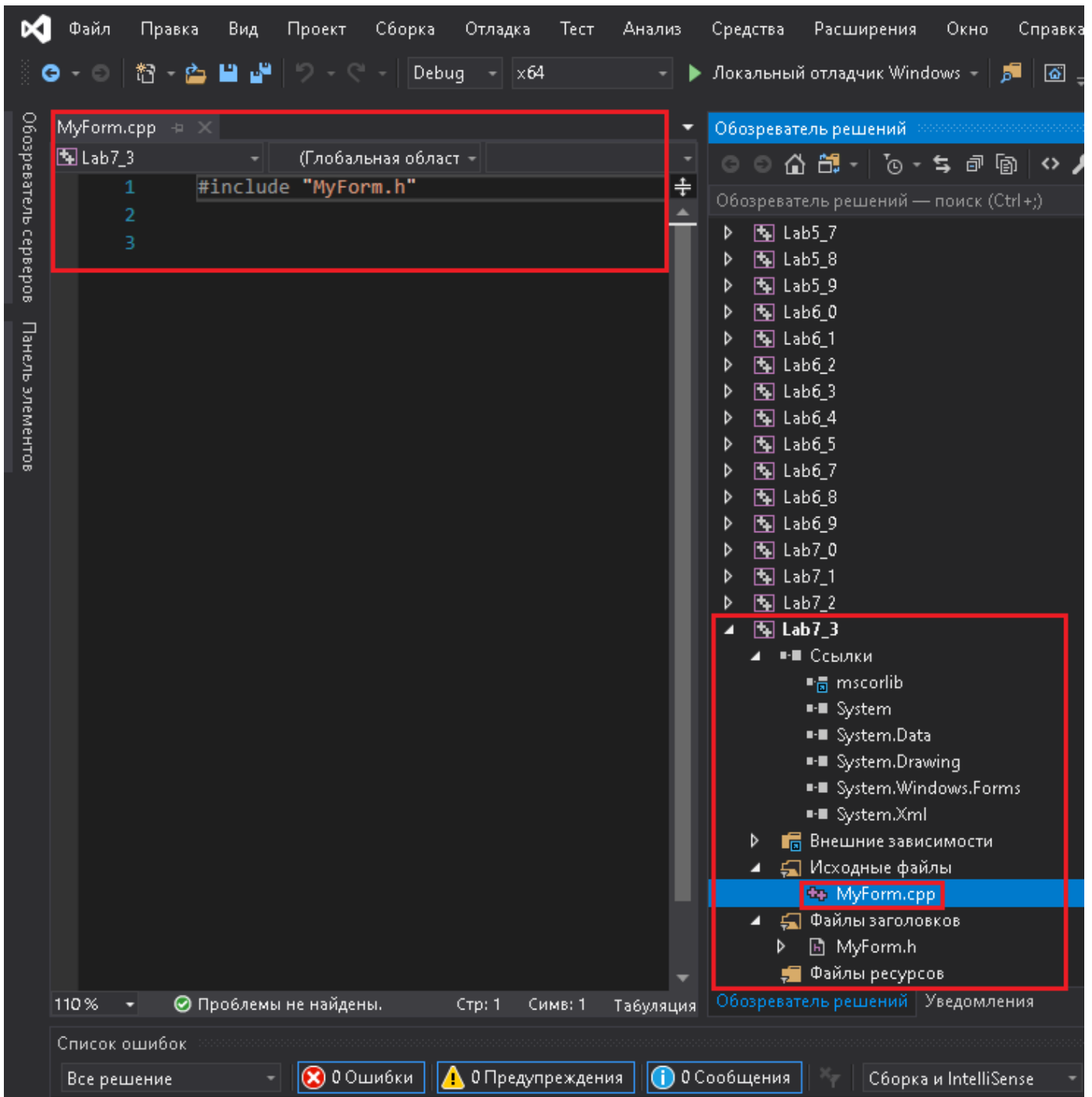
Жмем правой кнопкой мыши на название проекта и во всплывающем контекстном меню выбираем Добавить/Создать элемент/Слева в окне выбираем Visual C++/UI/справа выбираем Форма Windows Forms/Добавить. UI означает User Interface, то есть Интерфейс взаимодействия с Пользователем, который мы выберем в виде графического экранного оконного интерфейса.



Наверняка у вас появится окно с сообщением об ошибке, поскольку для нашего проекта еще недостаточно кода. Как видно по окну Список ошибок, на самом деле это Сообщение, информирующее нас о проблеме, а не настоящая ошибка в коде. Закрываем данное окно нажатием по кнопке «X» вверху слева.



В окне Обозреватель решений ищем наш проект и жмем мышью по файлу MyForm.cpp, добиваясь его открытия в режиме редактирования кода. Пока в данном файле только одна строка кода.



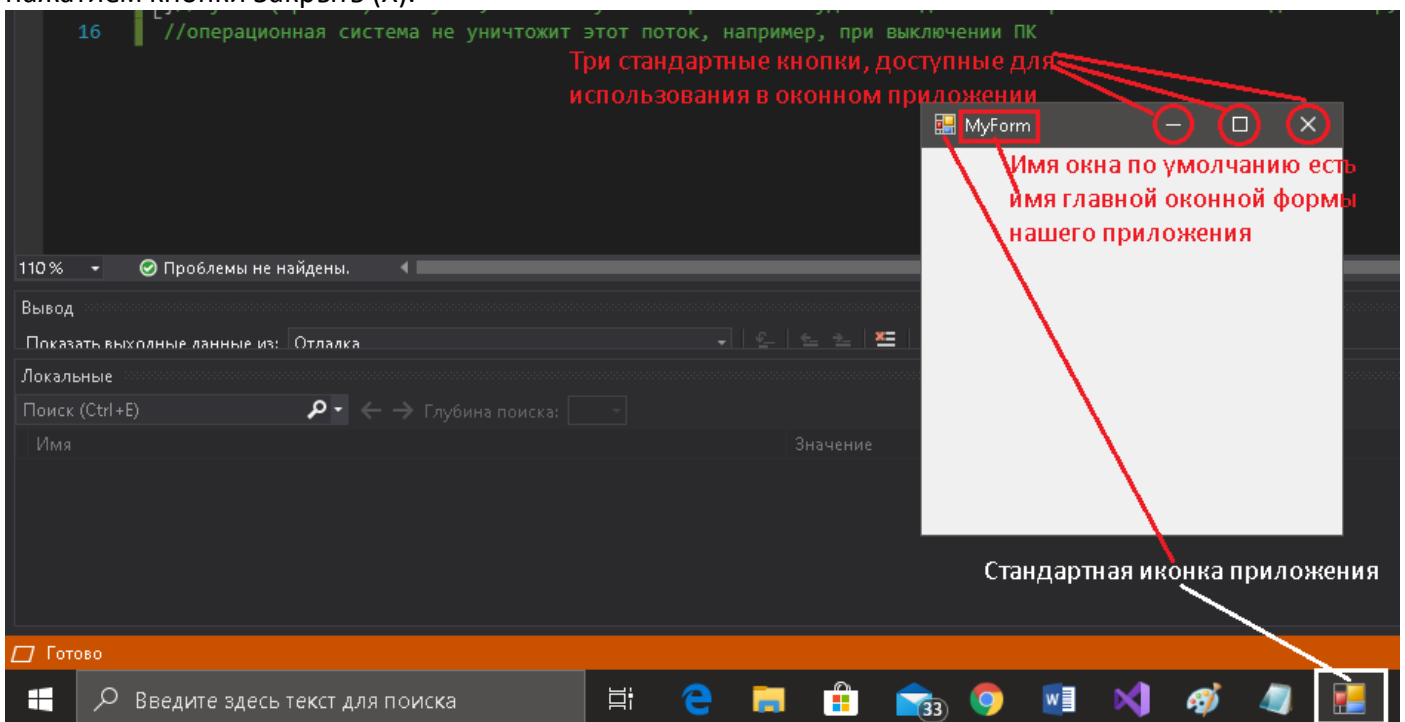
Файл `MyForm.cpp` нужно дописать нижеследующим кодом, причем именем пространства имен в каждом вашем оконном проекте будет имя конкретного вашего проекта (у меня `Lab7_3`, причем символ нижнего подчеркивания `_` набирать было нельзя и рабочее имя пространства имен было набрано как `Lab73` (проект при этом НЕ переименовывался и в окне Обзорщика решений по-прежнему называется `Lab7_3`)). Также строка кода `Application::Run(gcnew MyForm);` должна содержать имя конкретной созданной вами виндовс-формы конкретного вашего проекта. У меня виндовс-форма в диалоговом окне была названа как `MyForm.h`, поэтому и тут у меня должно быть именно это имя главной оконной формы. Остальной код стандартный и не должен содержать ошибок.

```

1 #include "MyForm.h"//эта строка кода была в файле MyForm.cpp. Означает подключение заголовочного файла MyForm.h. Все строки ниже мы пишем сами
2 #include <Windows.h>//подключение библиотеки Windows.h для работы оконных форм
3 using namespace Lab73;//используем пространство имен нашего проекта - пишем название нашего проекта тут, причем знак нижнего подчеркивания _ ПИСАТЬ НЕЛЬЗЯ - такого пространства
4 //имен MS VS2019 не видит. Набирайте код побуквенно, чтобы среда разработки успевала отображать вам доступные имена
5 [STAThreadAttribute]//указание атрибута потока однопоточного контейнера (атрибуты пишутся в квадратных скобках[]). Текущий поток должен быть задан как поток однопоточного
6 //контейнера (STA), чтобы вызовы OLE стали возможны
7 int WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)//главная функция для оконного приложения называется WinMain(), она может принимать 4 параметра и возвращает один целочисленный
8 //во всех трех строках кода ниже вызывается статический класс - методы вызываются у САМОГО КЛАССА, а НЕ у экземпляров класса, что возможно только для статических классов. В нашем
9 //приложении будет запущено только одно, поэтому реализовывать для этого "обычные" классы не было смысла и настройки для приложения реализовали через статический класс.
10 //Статический класс есть экземпляр самого себя, причем этот экземпляр возможен только один во всей конкретной программе, он создается в ее начале и уничтожается в ее конце
11 Application::EnableVisualStyles();//статический метод статического класса устанавливает доступность визуальных стилей и не позволяет отображать консоль
12 Application::SetCompatibleTextRenderingDefault(false);//статический метод статического класса устанавливает единообразное отображение шрифтов текста, если примет аргумент false
13 Application::Run(gcnew MyForm);//у класса Application вызываем метод Run, которому передаем указатель на создаваемый тут же конструктором экземпляр окна MyForm (тут пишем
14 return 0;//название НАШЕЙ оконной формы). Возвращаем из функции WinMain() значение 0, означающее успешное завершение работы нашей оконной программы. Писать перед этим
15 //system("pause") не нужно, поскольку наше приложение будет находиться в оперативной памяти ПК до тех пор, пока его не закроет пользователь нажатием на кнопку Закреть (X) или сама
16 //операционная система не уничтожит этот поток, например, при выключении ПК

```

Проект нужно сохранить и скомпилировать, чтобы создавался .exe-файл. Если ваше оконное приложение запустилось и работает, то цель создания оконного приложения как такового достигнута. Если MS VS2019 при компиляции выдает ошибки, то можно перекомпилировать проект (меню сверху: Сборка/Перестроить Lab7_3) или сохранить проект, закрыть MS VS2019 и открыть ее заново и скомпилировать проект снова. В моем случае приложение скомпилировалось сразу после сохранения, и первой компиляции. Окно приложения имеет отображающееся название MyForm и стандартную иконку, его можно двигать, сворачивать и разворачивать на весь экран, закрывать нажатием кнопки Закреть (X).



Посмотрим на созданные в оконном проекте файлы. Для этого жмем правой кнопкой мыши вверху по названию открытого файла Myform.cpp и во всплывающем контекстном меню жмем на вкладку Открыть содержащую папку. В отличие от консольных приложений, здесь больше файлов, все они так или иначе отвечают за настройки проекта, его ресурсы (иконки и т.д.), код.

Этот компьютер > Локальный диск (D:) > 2019 > Labs > Lab7_3				
Имя	Дата изменения	Тип	Размер	
x64	20.03.2020 3:06	Папка с файлами		
Lab7_3.vcxproj	20.03.2020 1:58	VC++ Project	6 КБ	
Lab7_3.vcxproj.filters	20.03.2020 1:58	VC++ Project Filte...	2 КБ	
Lab7_3.vcxproj.user	20.03.2020 1:38	Per-User Project O...	1 КБ	
MyForm.cpp	20.03.2020 3:06	C++ Source	3 КБ	
MyForm.h	20.03.2020 1:58	C/C++ Header	2 КБ	

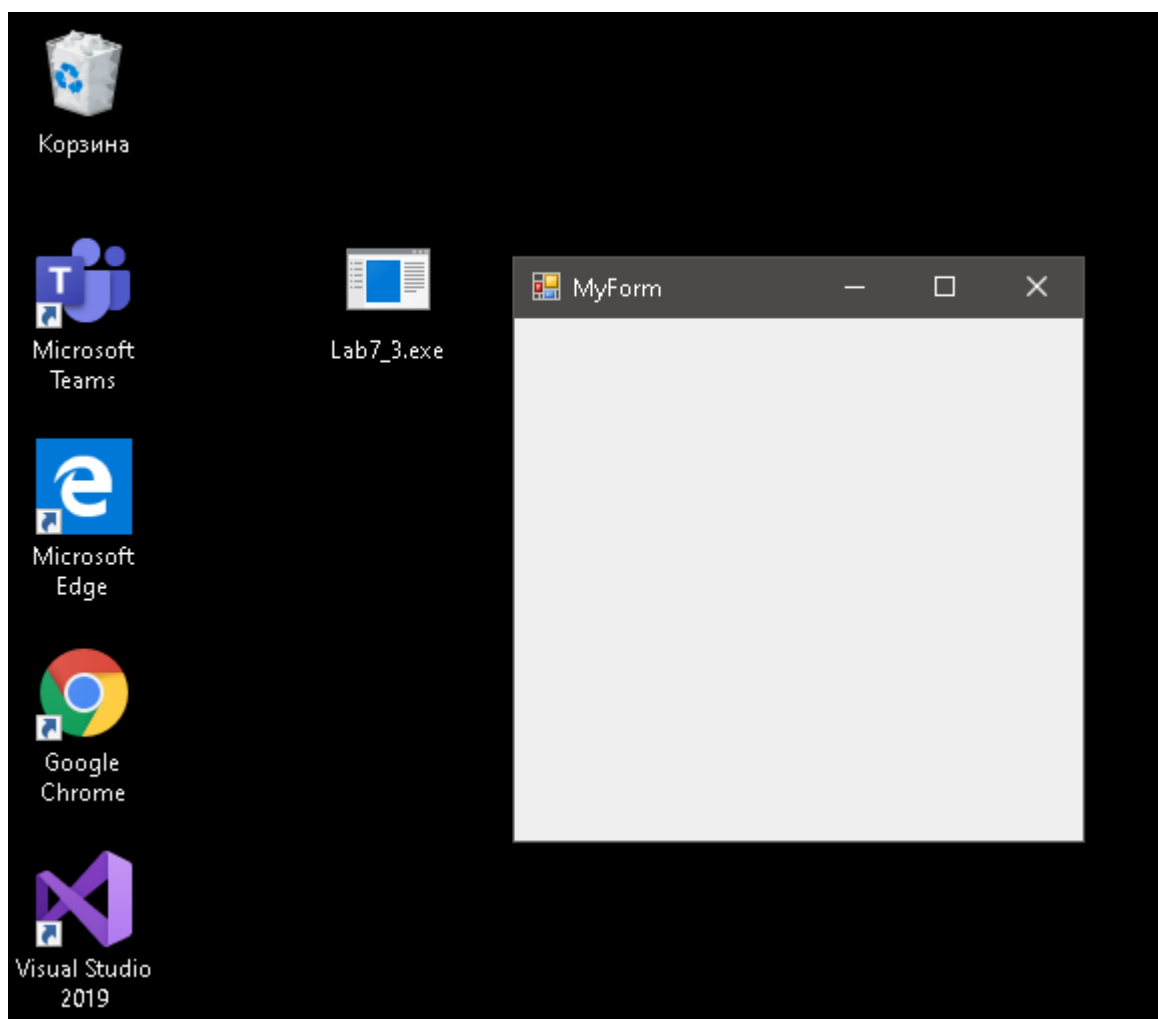
Поискем собственно само оконное приложение в виде исполнимого .exe-файла, которое будет находиться выше по дереву папок в папке с названием «x64/Debug». Приложение находится в

отладочной версии Debug, а не итоговой Release, что позволяет его не только компилировать и использовать, но и проводить его тестовую отладку, получать метрики (замеры, статистику) его работы.

Этот компьютер > Локальный диск (D:) > 2019 > Labs > x64 > Debug

Имя	Дата изменения	Тип	Размер
Lab6_9.ilc	08.03.2020 19:47	Incremental Linke...	512 КБ
Lab6_9.pdb	08.03.2020 19:47	Program Debug D...	508 КБ
Lab7_0.exe	09.03.2020 5:01	Приложение	86 КБ
Lab7_0.ilc	09.03.2020 5:01	Incremental Linke...	630 КБ
Lab7_0.pdb	09.03.2020 5:01	Program Debug D...	724 КБ
Lab7_1.exe	09.03.2020 7:14	Приложение	80 КБ
Lab7_1.ilc	09.03.2020 7:14	Incremental Linke...	697 КБ
Lab7_1.pdb	09.03.2020 7:14	Program Debug D...	724 КБ
Lab7_2.exe	13.03.2020 11:11	Приложение	73 КБ
Lab7_2.ilc	13.03.2020 11:11	Incremental Linke...	471 КБ
Lab7_2.pdb	13.03.2020 11:11	Program Debug D...	588 КБ
Lab7_3.exe	20.03.2020 3:06	Приложение	70 КБ
Lab7_3.exe.metagen	20.03.2020 3:06	Файл "METAGEN"	2 КБ
Lab7_3.pdb	20.03.2020 3:06	Program Debug D...	596 КБ
Project6_5.exe	22.01.2020 14:09	Приложение	65 КБ
Project6_5.ilc	22.01.2020 14:09	Incremental Linke...	464 КБ
Project6_5.pdb	22.01.2020 14:09	Program Debug D...	604 КБ

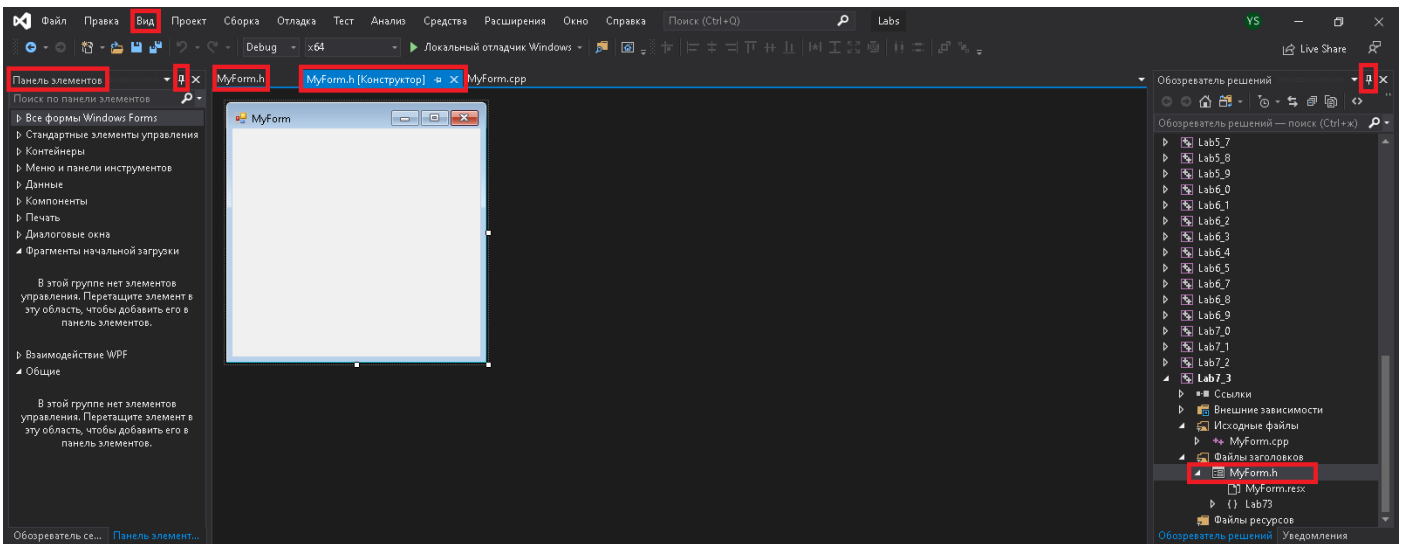
Скопируем .exe-файл на рабочий стол, закроем MS VS2019 и запустим наше приложение самостоятельно, без поддержки MS VS2019. Оно также работает. Закройте окно приложения и удалите его с рабочего стола.



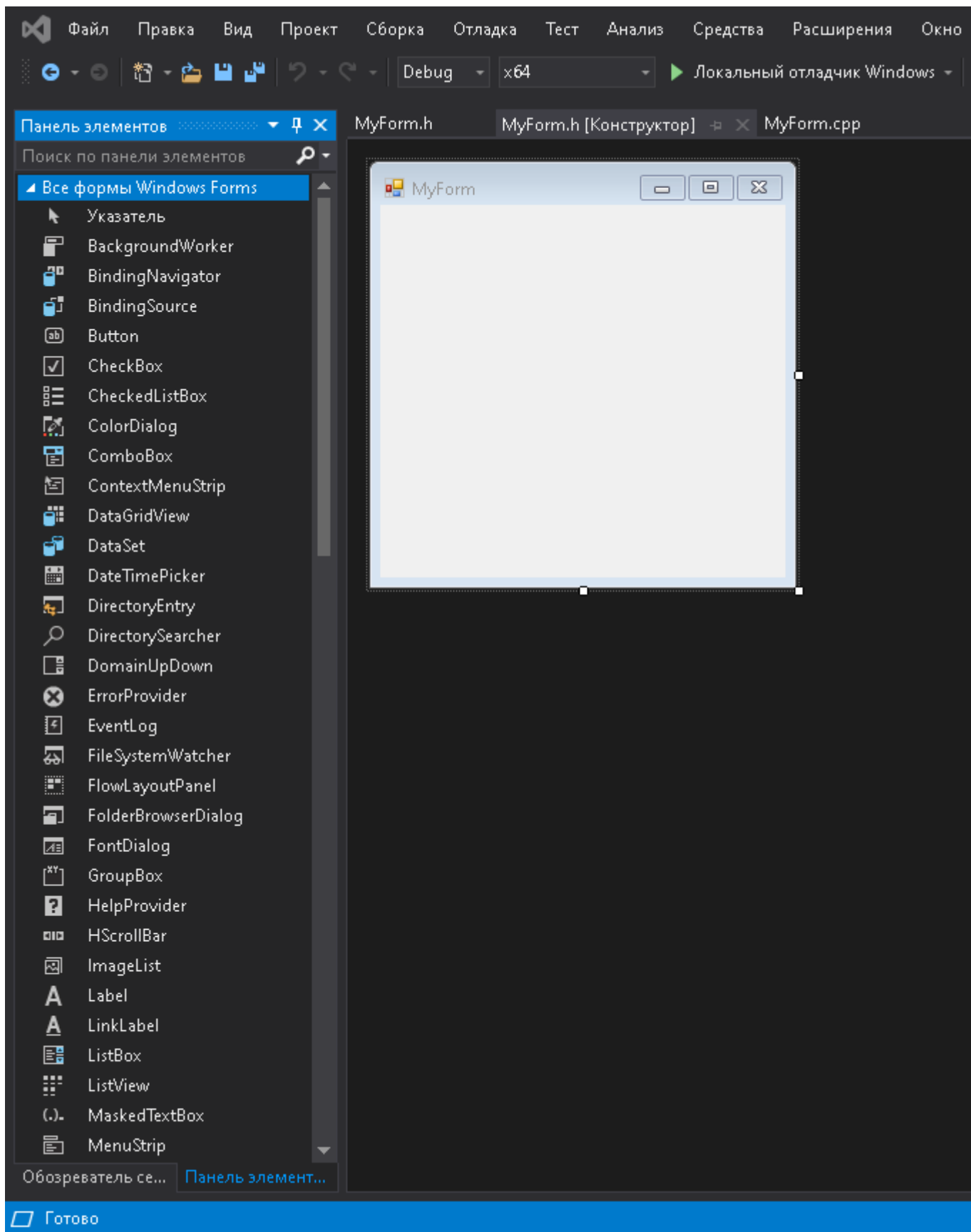
Снова запустите MS VS2019 и откройте в Обозревателе решений ваш оконный проект. Файл MyForm.cpp содержит код для запуска приложения, и этого достаточно, чтобы работало простое оконное приложение с минимальным функционалом. Но приложение пишется, чтобы удовлетворить запросы пользователя на решение его задач и потребностей. Для взаимодействия приложения и пользователя могут потребоваться различные графические элементы, основные из которых уже разработаны и доступны для использования в вашем проекте.

Сначала вам нужно открыть файл Myform.h в режиме графического Конструктора, для чего мышью дважды кликаем на имя файла MyForm.h в окне Обозревателя решений (или жмем правой кнопкой мыши по названию файла MyForm.h и во всплывающем контекстном меню выбираем пункт Открыть в конструкторе. В основной рабочей области должен открыться файл MyForm.h [Конструктор], то есть файл MyForm.h в режиме Конструктора, что мы и видим по отображающемуся макету окна нашего приложения. Этот отображающийся макет окна нашего приложения можно мышью видоизменять, например, растягивать за «квадратики» на его границах, после чего можно снова запустить приложение на исполнение и увидеть, что окно вашего приложения открывается в том размере и форме, как вы это настроили последний раз в режиме графического Конструктора.

Чтобы иметь возможность перетаскивать мышью графические элементы на макет оконной формы, нужно, чтобы было открыто окно Панель элементов. Если оно свернуто сбоку, разверните его, прикрепите на левую сторону MS VS2019, зафиксируйте в развернутом состоянии, нажав на кнопку «клепка» (находится рядом с кнопкой Закреть окно (X)). Если окна Панель элементов вы не находите, то возможно оно закрыто и его надо искать в верхнем меню Вид. Добейтесь отображения окна Панель элементов после того, как открыли файл MyForm.h в режиме графического Конструктора.



В окне Панель элементов разверните вкладку Все формы Windows Forms (смотрите иллюстрацию ниже). Вы должны увидеть список графических элементов, которые можно мышью переносить на оконную форму вашего приложения (макет окна). Учтите, что ваше оконное приложение может иметь несколько оконных форм (например, когда по нажатию кнопки на основной оконной форме отобразится новое окно с его элементами) и то, что не все элементы, перечисленные во вкладке Все формы Windows Forms являются видимыми (если их перетащить мышью на макет нашей оконной формы, они отобразятся внизу НИЖЕ макета окна нашего приложения; при работе приложения такие элементы либо отобразятся только при определенных обстоятельствах, либо не отобразятся совсем (например, база данных (массив в оперативной памяти с данными)); но визуальный элемент – сетка таблицы отображается на оконной форме, а данные для отображения в ячейках сетки таблицы могут браться из массива или базы данных). Если приложение многооконное, то главным его окном (главной оконной формой) будет являться то окно, которое запускается первым при запуске вашего приложения. Это главное окно в MyForm.cpp-файле указано в коде `Application::Run(gcnew MyForm);` // MyForm – это имя экземпляра главной оконной формы нашего приложения, запуском этого окна отобразится наше приложение на мониторе ПК.



Задание 2

Написать приложение, имеющее элемент Надпись (Label) и четыре Кнопки (Button) с функционалом:

А) надпись при запуске приложения не отображается. По нажатии на первую кнопку в поле оконной формы должна появиться надпись «Кнопка работает.»;

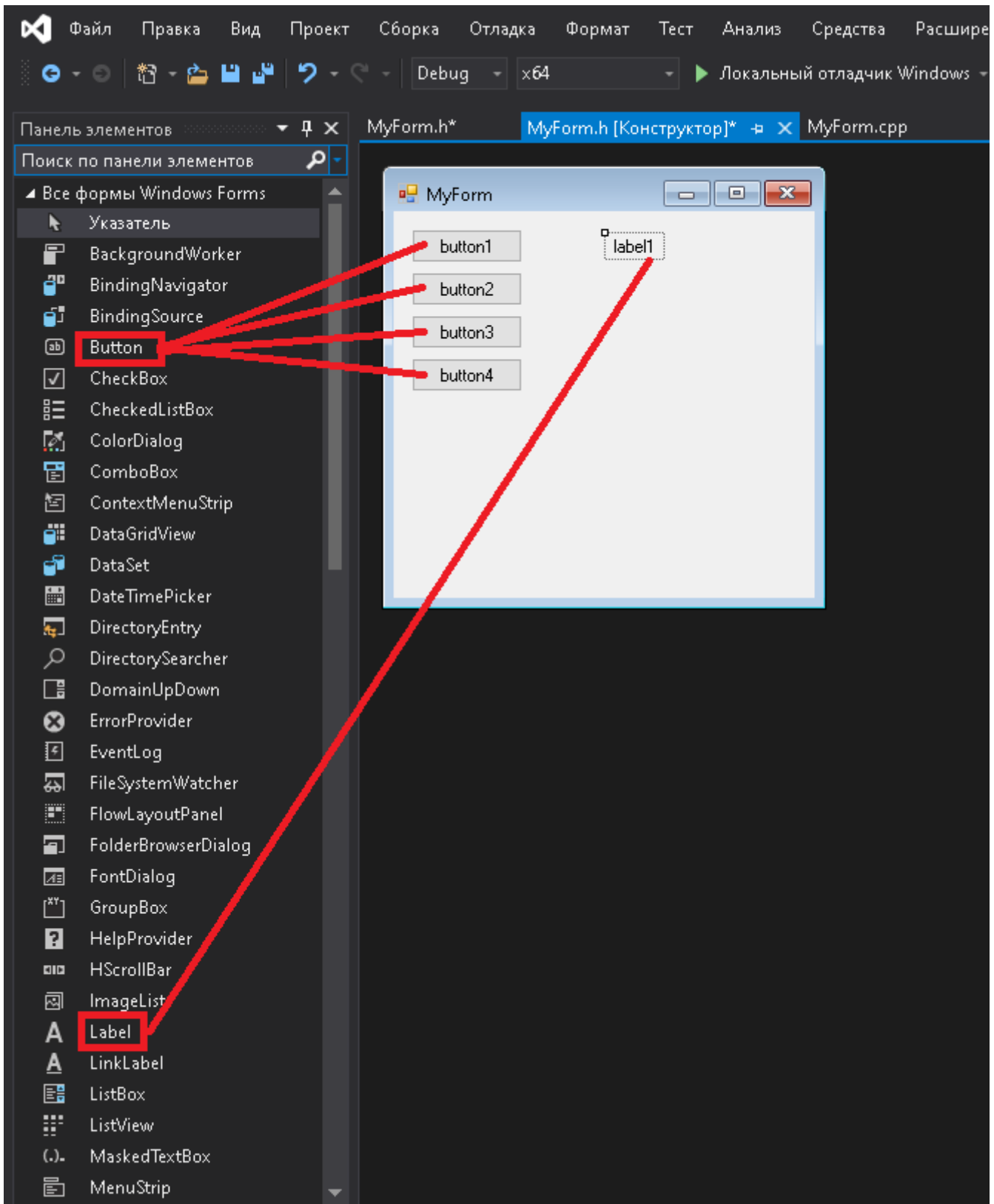
Б) при нажатии на вторую кнопку надпись должна исчезать с оконной формы. Если надпись и не отображалась, то ничего происходить не должно;

В) при нажатии на третью кнопку оконная форма меняет цвет на красный, а при повторном нажатии на эту кнопку – обратно на серый;

Г) по клику на четвертую кнопку наша оконная форма закрывается.

Выполняем. В качестве приложения возьмем наше текущее открытое приложение, поскольку оконная форма в нем фактически пустая. В окне Панель элементов / Все формы Windows Forms нажмем мышью по элементу Button (кнопка) и нажмем на макете оконной формы, куда хотим эту кнопку пометить; или мышью нажимаем на элемент Button и не отпуская левую кнопку мыши перетаскиваем элемент Button в то место на макете оконной формы, где хотим расположить данную кнопку. В дальнейшем все элементы на макете оконной формы в режиме графического Конструктора можно передвигать, изменять их размеры и размер самой оконной формы, если, например, ее уже недостаточно, чтобы расположить на ней все требуемые графические элементы. Всего перенесем на макет оконной формы 4 кнопки. Также перенесем из Панели элементов один элемент Label (Надпись; не редактируемая пользователем, посредством установки в нее курсора мыши, надпись).

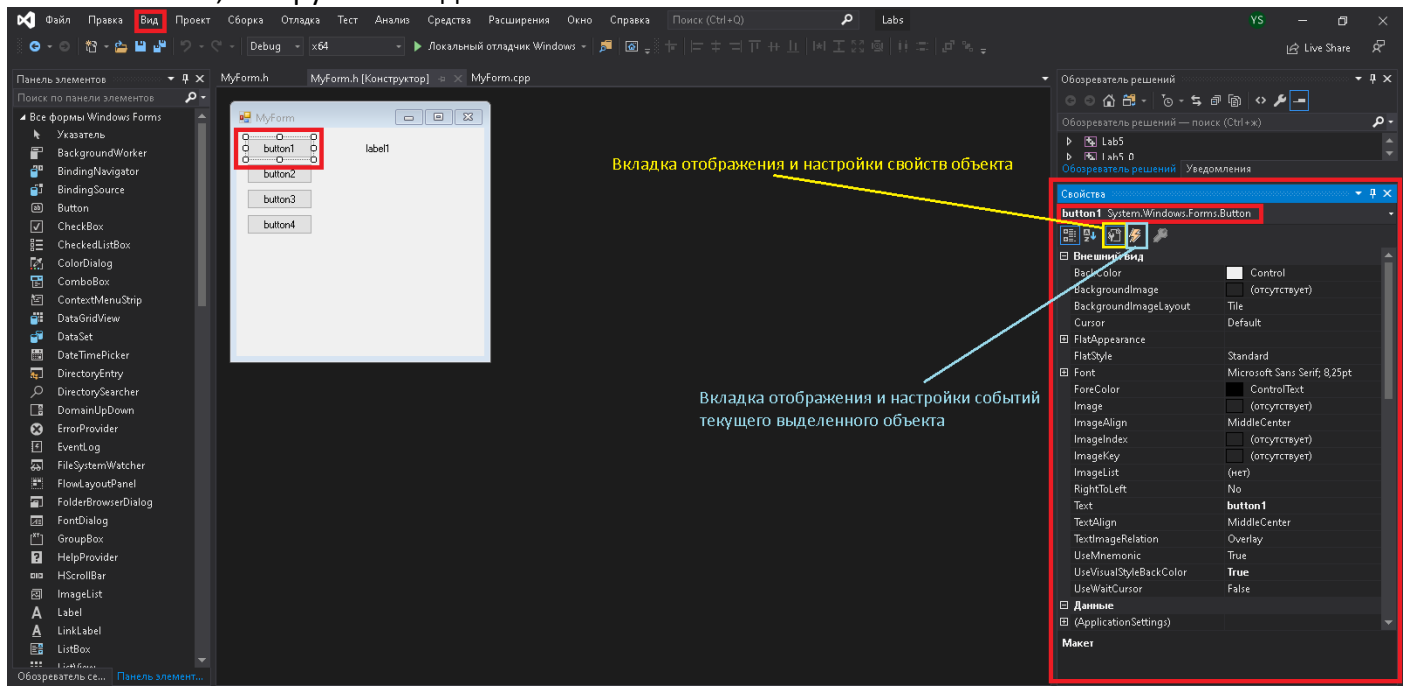
Внимание: при переносе элементов на оконную форму **НЕ** тратьте время на детальную подгонку одного элемента к другому, выстраивание их «по линии» и т.д., поскольку вы не можете знать в это время, сколько всего элементов будет расположено по итогу на вашей форме, как расположение новых элементов потребует их увязки со старыми элементами. Вероятно, вы будете менять размеры оконной формы, что неминуемо потребует изменения расположения уже находящихся на оконной форме элементов и сделает бессмысленной работу по их «точному» расположению и выравниванию ранее. Поэтому изначально располагайте элементы, не обращая внимания на точность их месторасположения (до +/- полусантиметра), далее прописывайте им обработчики событий и тестируйте работоспособность функционала этих элементов и всего приложения в целом, и только в самом конце работы, **ПОСЛЕ** тестирования основного функционала приложения, уделите внимание визуальной подгонке месторасположения элементов, многократно не трата впустую свое время на такую работу ранее.



Все доступные нам графические элементы являются экземплярами соответствующих классов (Button, Label, TextBox и т.д.). Когда мы перетаскиваем элемент на макет оконной формы, Конструктор создает экземпляр (объект) соответствующего класса. Этот экземпляр (переменная типа конкретного класса) получает автосгенерированное имя, которое можно изменить, а можно пользоваться и автосгенерированным именем, тем более, что элементы получают последовательные имена, присваиваемые им по определенным правилам. Например, все созданные 4 кнопки на оконной форме – экземпляры класса Button, получили последовательные имена button1, button2, button3 и button4. Надпись – экземпляр класса Label, получила как переменная имя label1 и т.д. Пока что это просто визуальные элементы на оконной форме. Если

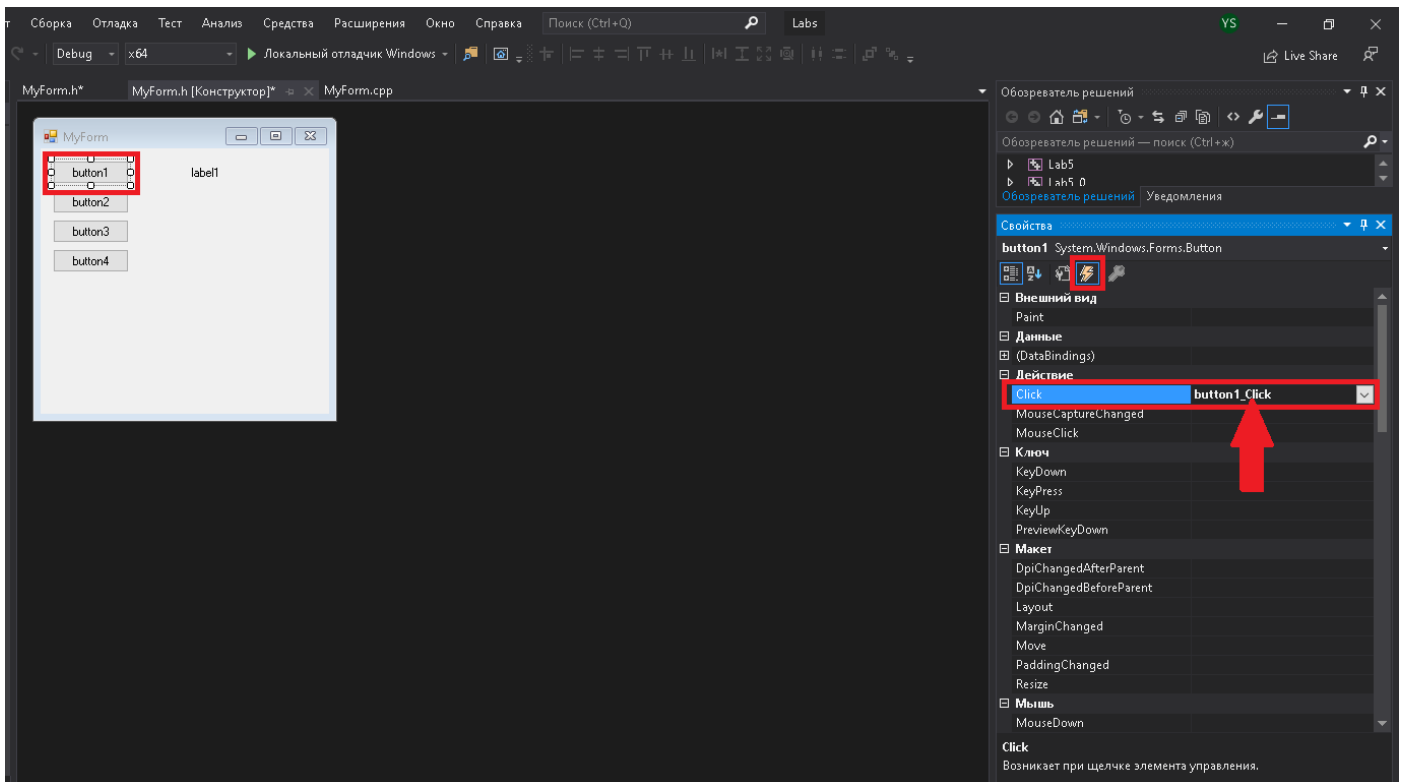
запустить приложение и нажимать на эти элементы, они будут нажиматься, будет анимация нажатия, но никакого полезного результата от нажатия на эти элементы не будет, поскольку они не имеют обработчиков соответствующих событий – однократного нажатия мышью на элемент, двукратного нажатия мышью на элемент, наведения курсора мыши на элемент, нажатия на элемент правой кнопкой мыши и т.д. Можно и нужно создать обработчики действий пользователя над соответствующими элементами на форме, поскольку именно с этой целью создаются графические элементы на оконной форме – в противном случае они не нужны на форме.

Для того, чтобы создать обработчик события – действия пользователя над графическим объектом, нужно поставить курсор на данный элемент и обратиться к окну Свойства, где будут отображаться все свойства данного объекта, которые можно у него настроить. Если вы не видите окна Свойств объекта, то надо в верхнем меню VS 2019 выбрать пункты меню Вид / Окно свойств. В окне свойств отображаются настраиваемые характеристики **того объекта**, который вы выделите мышью к этому времени: если нажать на кнопку button1, то в Окне свойств отобразятся свойства именно кнопки button1, а если нажать мышью по самому макету окна MyForm (разумеется, по свободному месту на этой оконной форме, а не объектам (кнопкам и лейблу), которые находятся на ней), то в Окне свойств отобразятся характеристики окна MyForm вашего приложения. Свойства кнопки button1, которую мы выделили мышью:



Для того, чтобы настроить обработчик события нажатия пользователем левой кнопкой мыши по кнопке button1 (то есть реакцию кнопки button1 на событие нажатия по ней) надо:

1. Выделить мышью кнопку button1;
2. В окне Свойства для объекта button1 нажать на вкладку **События** (пиктограмма Молния). Внешний вид объекта можно настроить по вкладке Свойства (пиктограмма листа бумаги и гаечного ключа);
3. Нажать мышью справа от пункта Click в разделе Действие. Название метода-обработчика нажатия (клика) мышью по кнопке button1 автоматически появится в этом поле или туда можно поставить курсор и набрать **button1_Click** и нажать кнопку Enter.



4. Если еще не открылся файл MyForm.h в режиме редактирования кода (а не графического Конструктора), то нужно либо дважды нажать мышью справа от поля Click в окне Свойства/События либо в окне Обзорщик решений правой кнопкой мыши вызвать контекстное меню у файла MyForm.h и во всплывшем окне контекстного меню выбрать пункт Перейти к коду. Вы увидите примерно такой автосгенерированный код файла. В некоторых местах его можно редактировать, дописывать.

#pragma once //стандартная директива препроцессору для заголовочного header-файла MyForm.h
 //тела методов-обработчиков событий в этом файле программисту можно и нужно, но остальной код автосгенерирован роботом и его нежелательно или запрещено изменять (робот при изменении оконной формы в графическом Конструкторе все равно регенерирует этот код и ваш код тут уничтожится или будет противоречить остальному коду - будут ошибки). Но подписки на события удалять можно

namespace Lab73 //создано пространство имен нашего проекта Lab73 (символ _ автоматически отфильтрован)

{//подключаются пространства имен Системы, Модели компонентов, Коллекций (для работы с массивами, списками и т.д.), пространство имен Forms для работы с оконными формами, Данными и Отрисовкой графических элементов. Из этих пространств имен нам доступны классы и их члены

```
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 //открытый ссылочный класс
```

Myform наследует классу Form, который находится в пространстве имен класса Forms, а он - в классе Windows, а он - в классе System. Наследование открытое. Что это значит?

```

{
public://раздел открытых общедоступных членов класса MyForm
    MyForm(void)//конструктор без параметров по умолчанию
    {
        InitializeComponent();//метод инициализации компонента, требуемый для
работы конструктора MyForm()
        //
        //TODO: добавьте код конструктора
        //
    }

protected://раздел защищенных членов класса MyForm
    /// <summary>
    /// Освободить все используемые ресурсы.
    /// </summary>
    ~MyForm()//деструктор объекта класса MyForm
    {
        if (components)//if (components != NULL)//если компоненты еще существуют
(знаимают оперативную память)
        {
            delete components;//то удалить их из оперативной памяти
        }
    }

private: System::Windows::Forms::Button^ button1;//при создании графических объектов в
конструкторе они обязательно создаются в коде файла Myform.h в виде защищенных указателей на
соответствующие объекты. Код Button^ button1 идентичен Button* button1, но при этом еще и
поддерживает очищение объекта из памяти по достижении закрывающей скобки локальной области
где он создан
protected:
private: System::Windows::Forms::Button^ button2;
private: System::Windows::Forms::Button^ button3;
private: System::Windows::Forms::Button^ button4;
private: System::Windows::Forms::Label^ label1;

private:
    /// <summary>
    /// Обязательная переменная конструктора.
    /// </summary>
    System::ComponentModel::Container ^components;//создан (объявлен) указатель на
контейнер (вместилище) для всех компонентов оконной формы

#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)//полное определение метода InitializeComponent() с
телом
    {
        //this - это указатель на текущий объект - оконную форму MyForm, экземпляр класса
MyForm

```

```

this->button1 = (gcnew System::Windows::Forms::Button()); //код this->button1
означает: указатель на кнопку button1 на ЭТОЙ оконной форме указателю присвоить адрес ДООП,
выделенной конструктором Button() под кнопку button1 и так далее по аналогии
this->button2 = (gcnew System::Windows::Forms::Button());
this->button3 = (gcnew System::Windows::Forms::Button());
this->button4 = (gcnew System::Windows::Forms::Button());
this->label1 = (gcnew System::Windows::Forms::Label());
this->SuspendLayout(); //вызов метода SuspendLayout() текущей оконной формы
//
// button1 //описание характеристик кнопки button1
//
this->button1->Location = System::Drawing::Point(12, 12); //у кнопки button1
вызываются свойства в СиШарп-нотации кода (у свойств нет круглых скобок []). Через свойства
устанавливаются значения в поля объекта, а значения эти обеспечивают размер объекта, его
месторасположение по осям X и Y, наличие на нем надписей, шрифт и размер этих надписей и
остальные характеристики. Эти характеристики нужно редактировать НЕ тут в коде, а через редактор
окна Свойства, который изменит данный код корректно
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); //на событие клика по кнопке button1 создается метод-обработчик, на
который тут делается ссылка - ссылка на метод класса
// обработчик присваивается оператором +=, а отвязывается оператором -=
// button2
//
this->button2->Location = System::Drawing::Point(12, 41); //аналогично
вышенанписанному для остальных элементов на оконной форме
this->button2->Name = L"button2";
this->button2->Size = System::Drawing::Size(75, 23);
this->button2->TabIndex = 1;
this->button2->Text = L"button2";
this->button2->UseVisualStyleBackColor = true;
//
// button3
//
this->button3->Location = System::Drawing::Point(12, 70);
this->button3->Name = L"button3";
this->button3->Size = System::Drawing::Size(75, 23);
this->button3->TabIndex = 2;
this->button3->Text = L"button3";
this->button3->UseVisualStyleBackColor = true;
//
// button4
//
this->button4->Location = System::Drawing::Point(12, 99);
this->button4->Name = L"button4";
this->button4->Size = System::Drawing::Size(75, 23);

```

```

this->button4->TabIndex = 3;
this->button4->Text = L"button4";
this->button4->UseVisualStyleBackColor = true;
//
// label1
//
this->label1->AutoSize = true;
this->label1->Location = System::Drawing::Point(145, 17);
this->label1->Name = L"label1";
this->label1->Size = System::Drawing::Size(35, 13);
this->label1->TabIndex = 4;
this->label1->Text = L"label1";
//
// 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->label1);
this->Controls->Add(this->button4);
this->Controls->Add(this->button3);
this->Controls->Add(this->button2);
this->Controls->Add(this->button1);
this->Name = L"MyForm";
this->Text = L"MyForm";
this->ResumeLayout(false);
this->PerformLayout();

}

#pragma endregion
private://раздел закрытых членов класса
    System::Void button1_Click(System::Object^ sender, System::EventArgs^ e)//сгенерировался
    метод-обработчик клика мыши пользователя по кнопке button1
    {
        //метод принимает указатель на объект-отправитель события (sender), это объект самого
        базового (родительского) класса Object, и указатель на список входных аргументов, которые объект-
        отправитель посылает объекту-получателю сообщения о свершении события
        //тело метода-обработчика пустое - его заполняет программист. Метод ничего не
        возвращает
    }
};
}

```

5. По заданию нам нужно, чтобы при нажатии пользователем по кнопке button1 на оконной форме появлялась надпись «Кнопка работает.». За эту надпись у нас отвечает элемент label1. Допишем тело обработчика события button1_Click() строкой кода label1->Text = «Кнопка работает.»,, то есть в теле этого метода вызываем объект label1, поскольку это указатель на объект, то посредством косвенной адресации -> обращаемся к свойству лейбла Text и присваиваем этому свойству-сеттеру символьный массив «Кнопка работает.». Если воспринимать свойство Text методом, то мы передали в метод-сеттер символьный массив «Кнопка работает.», а сам метод присвоил полю текст эту фразу. Причем это происходит не на этапе компиляции, и даже в момент исполнения программы это может не сработать никогда, если пользователь так и не захочет нажать

на кнопку button1. Особенность обработчиков событий в том, что они сработают (выполнятся) только после того, как произойдет событие, на которое они подписаны (в нашем случае – клик пользователя по кнопке button1). Если событие не произойдет – подписанный на него обработчик не выполнится никогда. Если событие будет происходить несколько раз, то и подписанные на них методы тоже будут срабатывать (выполняться) несколько раз сразу после свершений соответствующих событий.

`this->button1->Click += gcnew System::EventHandler(this, &MyForm::button1_Click);` //на событие клика по кнопке button1 создается метод-обработчик, на который тут делается ссылка - ссылка на метод класса

//другой код

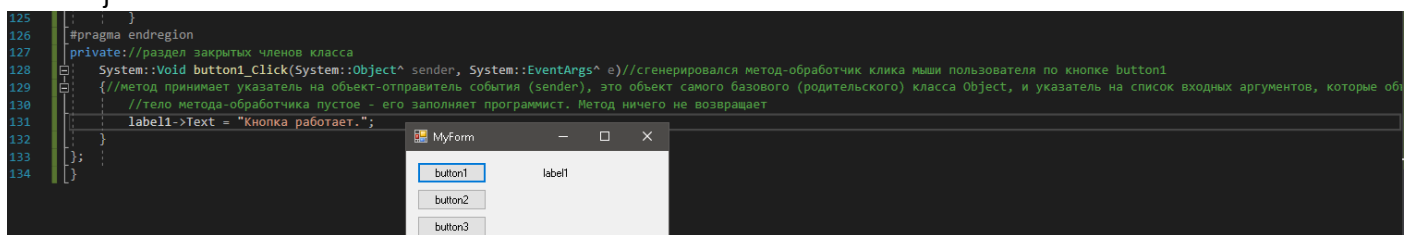
`System::Void button1_Click(System::Object^ sender, System::EventArgs^ e)` //сгенерировался метод-обработчик клика мыши пользователя по кнопке button1

//метод принимает указатель на объект-отправитель события (sender), это объект самого базового (родительского) класса Object, и указатель на список входных аргументов, которые объект-отправитель посылает объекту-получателю сообщения о свершении события

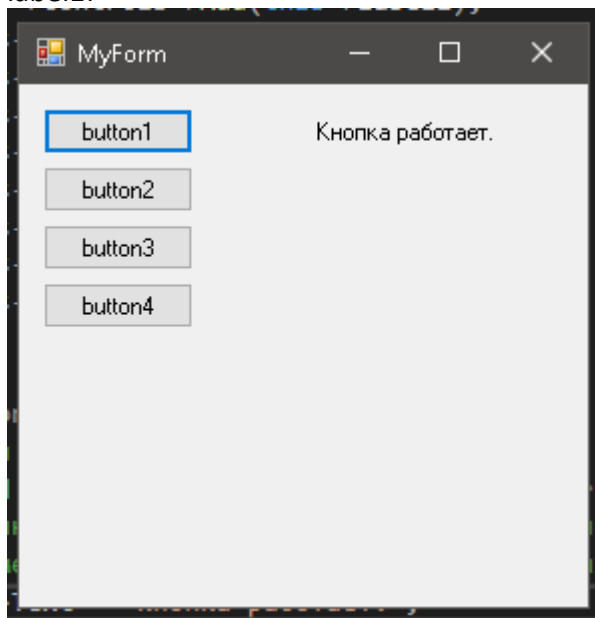
//тело метода-обработчика изначально пустое - его заполняет программист. Метод ничего не возвращает

`label1->Text = "Кнопка работает.";`

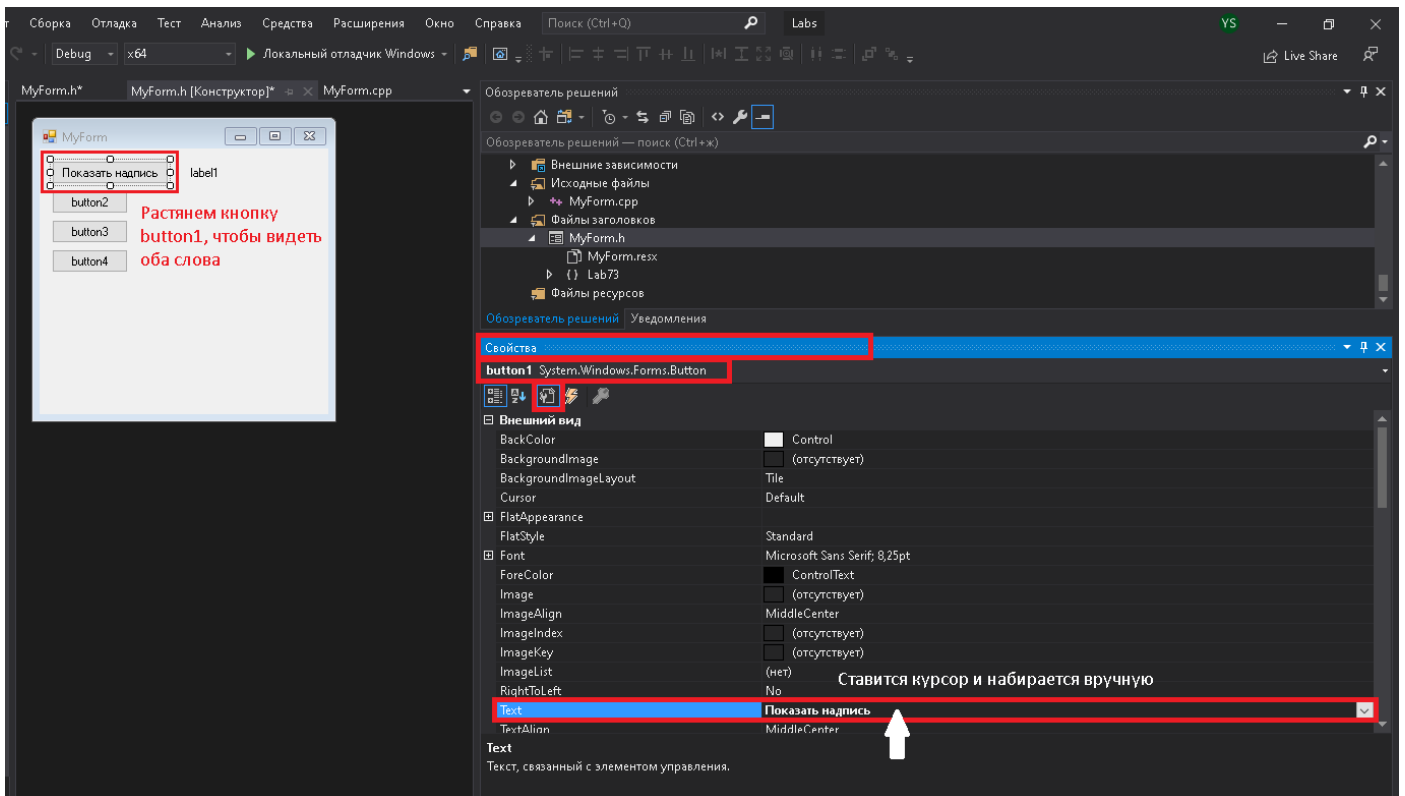
}



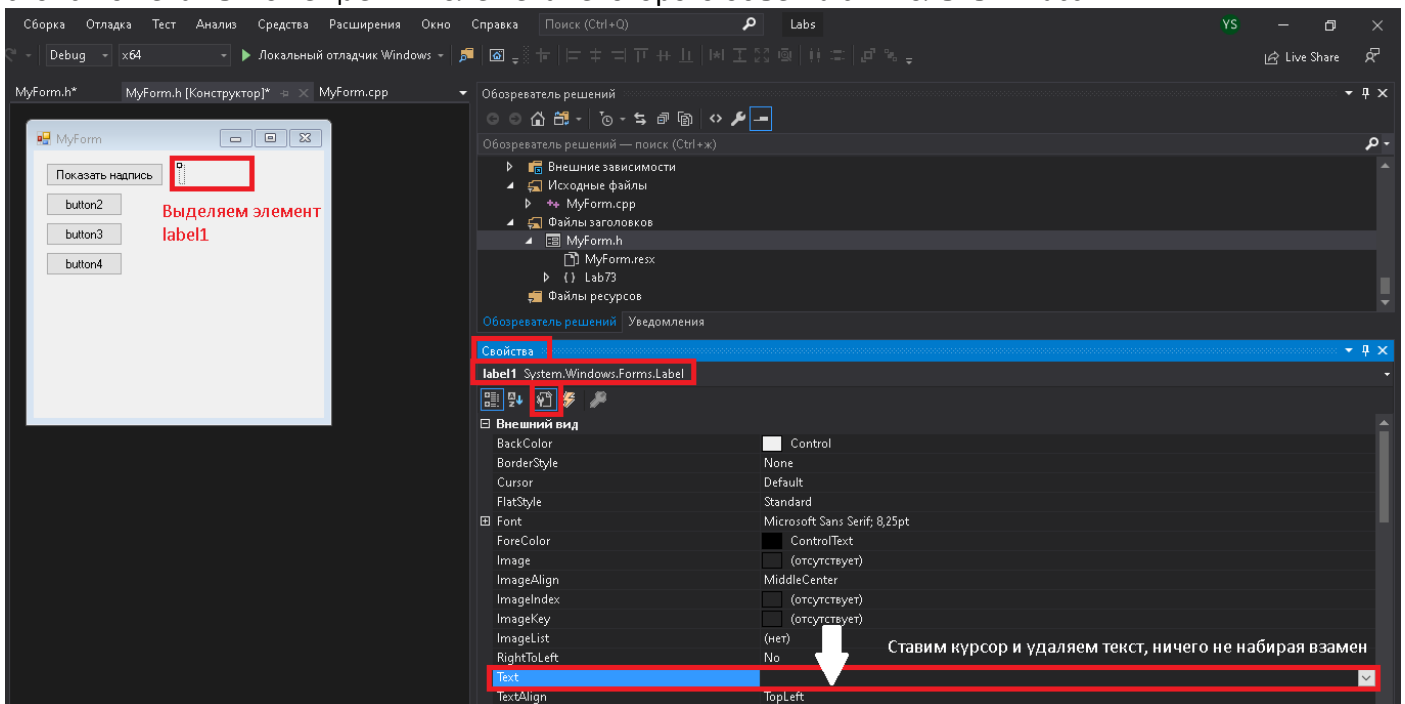
Запустим программу и нажмем на кнопку button1. Должна отобразиться фраза в элементе label1:



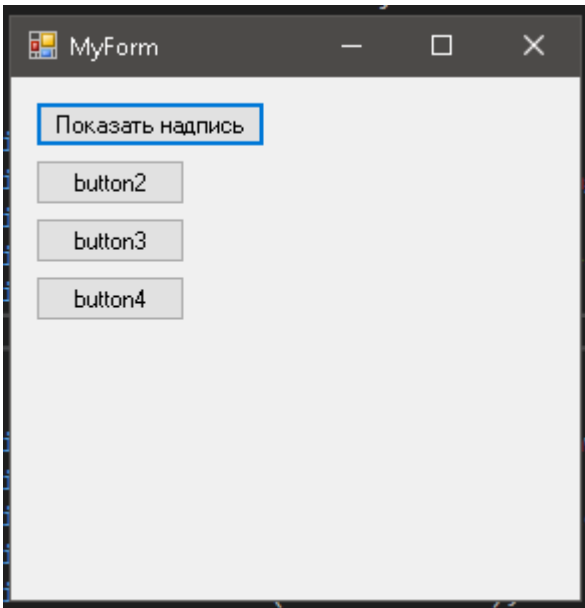
Функционал кнопки button1 работает, но при запуске приложения в качестве надписи на кнопке button1 отображается имя этого указателя на кнопку «button1», хотя была бы логична надпись на кнопке «Показать надпись». Аналогично сразу отображается надпись «label1», хотя для надписи label1 было бы логичным изначально не отображать вообще никакого текста своего названия, то есть отображать пустую строку "". Это можно настроить с помощью вкладки Свойства в окне Свойства. Выделяем нужную кнопку button1, в окне Свойства на вкладке Свойства находим поле Text и справа от него ставим курсор, удаляем старую надпись и набираем новую, жмем Enter. На форме кнопку нужно растянуть, чтобы были видны оба слова.



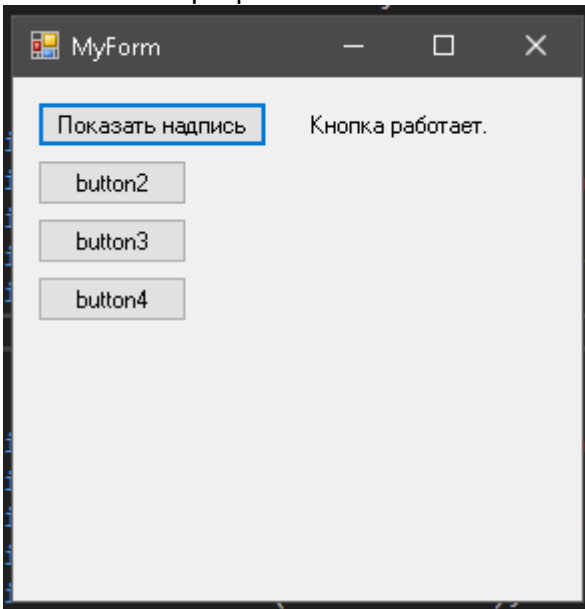
После этого выделяем элемент label1 и в его свойстве Text удаляем старую надпись и ничего не набираем взамен, чтобы при запуске приложения надпись-лейбл себя никаким текстом не выдавала. Кстати, в обоих случаях мы меняем свойства Text у объектов, но сами эти объекты (переменные, указатели) остаются в коде со своими изначальными именами, поскольку свойство Text – это просто надпись на элементе, а не действительное имя этой переменной в коде. Через свойство Text мы помещаем в поле Text некоторого объекта символьный массив.



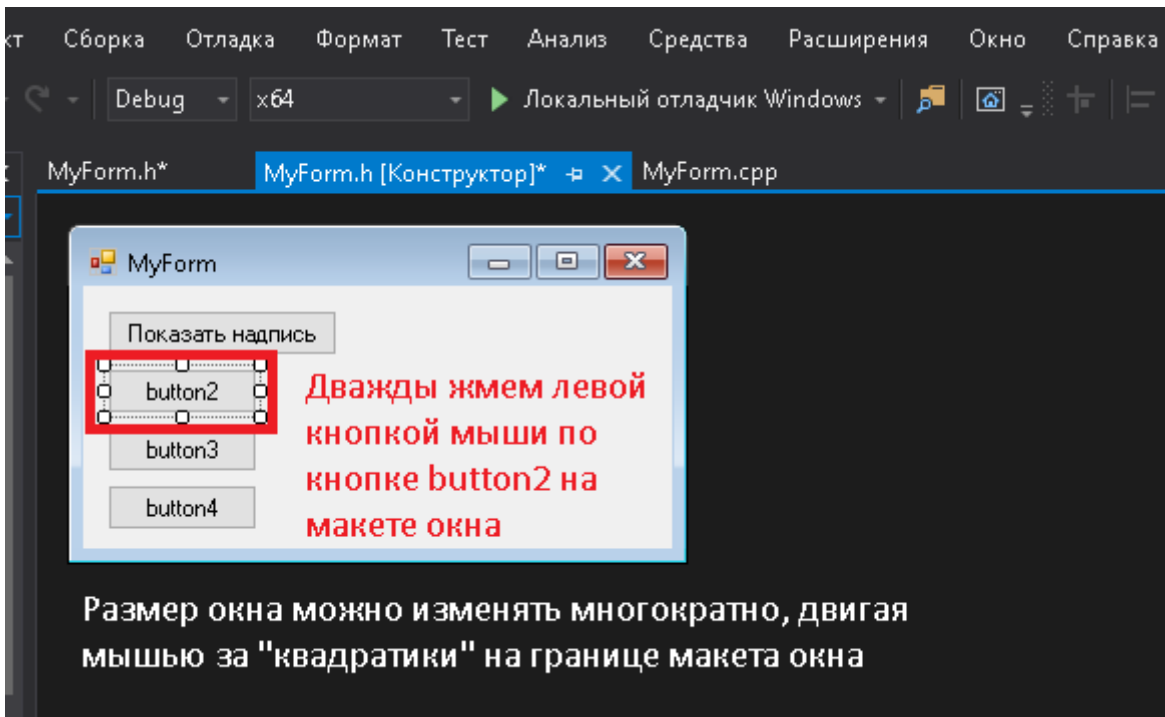
Тестируем программу. Окно программы при запуске:



Окно программы после нажатия на кнопку button1:



Выполняем задания дальше. Нужно, чтобы при клике по кнопке button2 надпись в label1 становилась невидимой, то есть там отображалась строка без единого символа. Создадим обработчик клика по кнопке button2, дважды нажав мышью по кнопке button2 на макете окна приложения в режиме графического Конструктора. Роботом автоматически должен сгенерироваться обработчик клика по кнопке button2, создаваемый метод должен подписаться на события клика по данной кнопке и в режиме отображения кода открыться файл MyForm.h с написанным обработчиком клика по этой кнопке, но, разумеется, с пустым телом, которое мы можем дописать.



В коде должно появиться примерно следующее:

```

127 private://раздел закрытых членов класса
128 System::Void button1_Click(System::Object^ sender, System::EventArgs^ e)//сгенерировался метод-обработчик клика мыши поль
129 { //метод принимает указатель на объект-отправитель события (sender), это объект самого базового (родительского) класса Ob
130     //тело метода-обработчика пустое - его заполняет программист. Метод ничего не возвращает
131     label1->Text = "Кнопка работает.";
132 }
133 private: System::Void button2_Click(System::Object^ sender, System::EventArgs^ e)
134 {
135     //пустое тело метода-обработчика события клика по кнопке button2
136 }
137 };
138 }

```

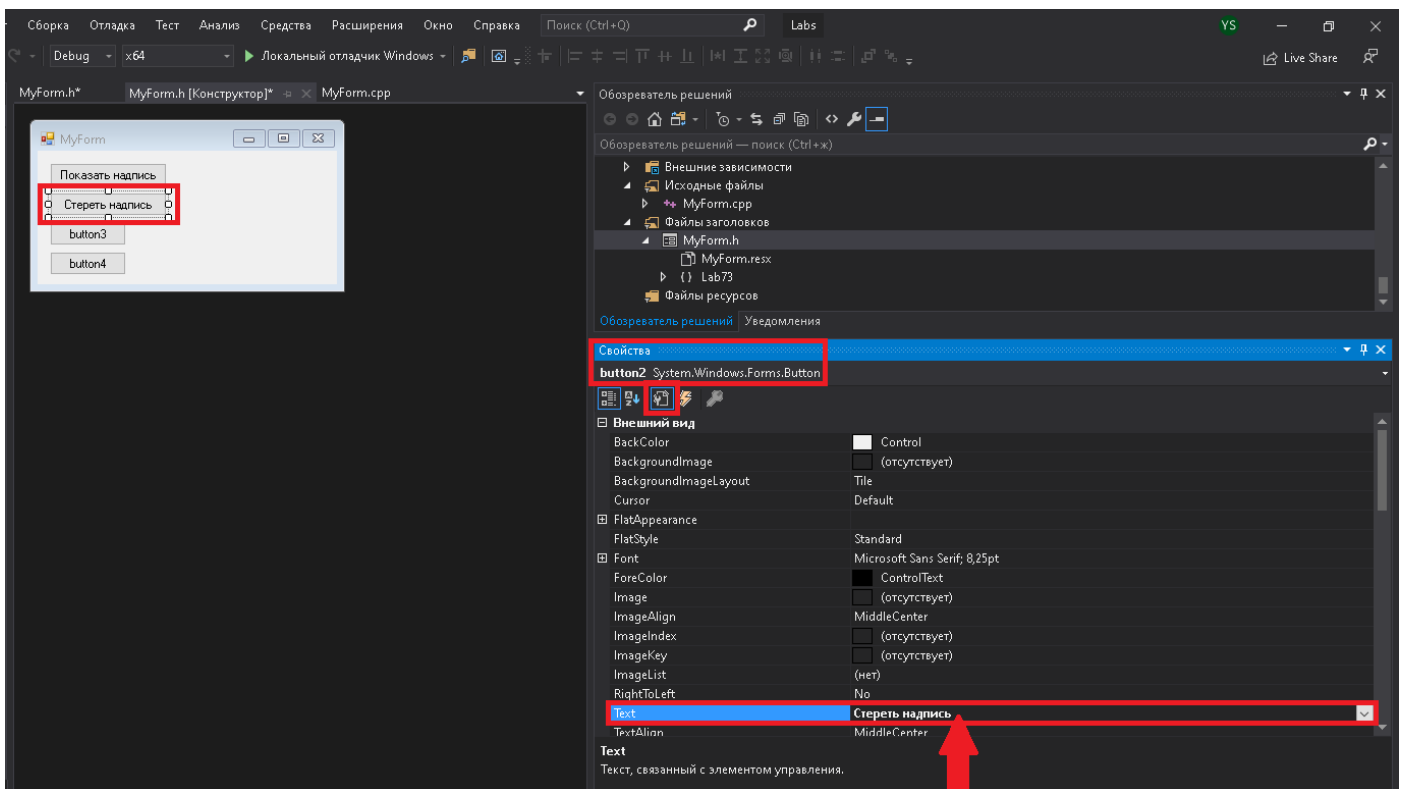
Дописываем код тела обработчика. По нажатии на данную кнопку текст надписи label1 должен становиться «пустым».

```

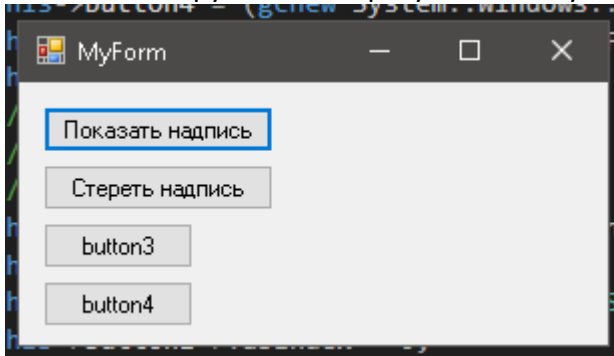
133 private: System::Void button2_Click(System::Object^ sender, System::EventArgs^ e)
134 {
135     //пустое тело метода-обработчика события клика по кнопке button2
136     label1->Text = "";
137 }
138 };
139 }

```

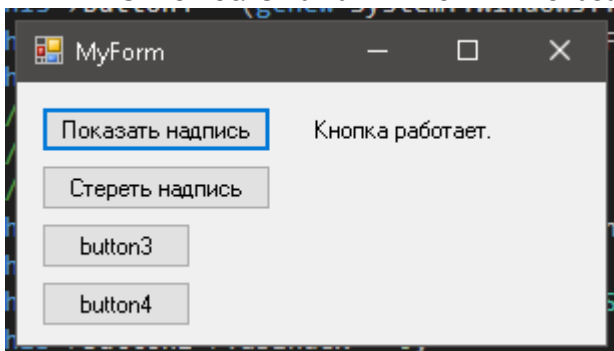
А в свойствах кнопки button2 изменим ей надпись на фразу «Скрыть надпись». Сделайте это по аналогии с изменением надписи и размера кнопки button1.



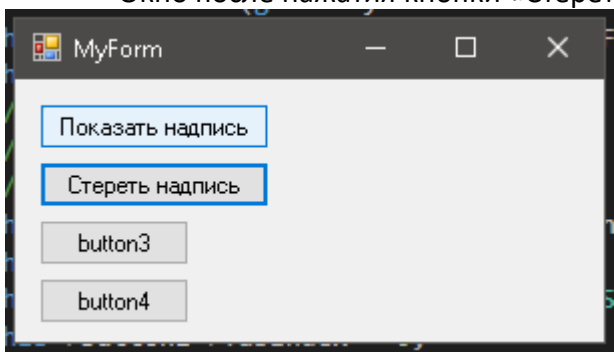
Тестируем. Окно сразу после запуска:



Окно после нажатия кнопки «Показать надпись»:

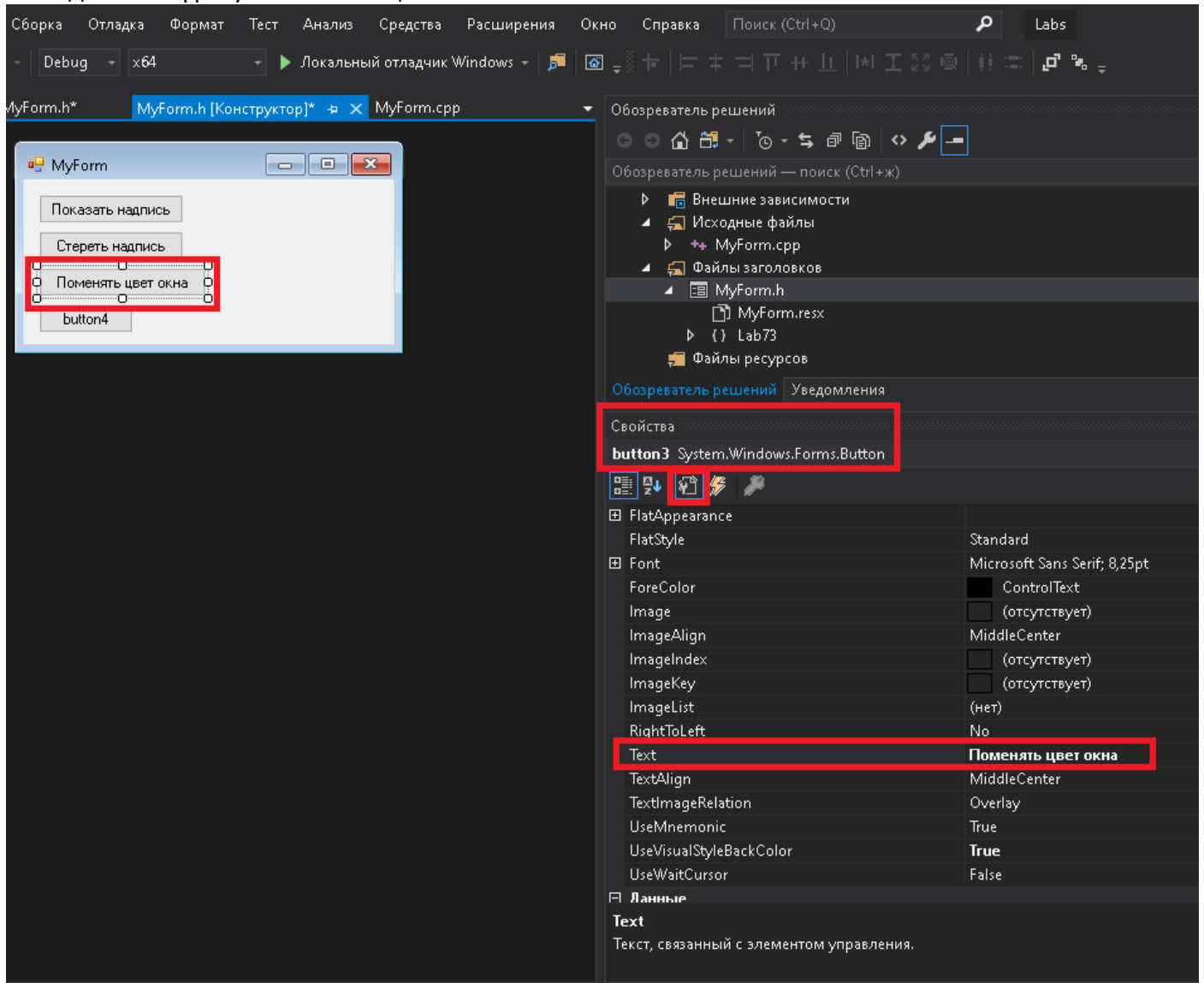


Окно после нажатия кнопки «Стереть надпись»:



И так далее. При нажатии кнопки «Стереть надпись», когда надпись еще не отображается, визуально ничего не происходит. Ошибок при этом не возникает. Программа в бесконечном цикле ожидает действия пользователя (события) до тех пор, пока он не нажмет кнопку Закреть приложение (X).

Выполним следующее задание. Форма должна менять цвета с серого на красный и обратно при нажатии на кнопку button3. Сразу сделайте кнопке button3 обработчик клика по ней и измените ей надпись на фразу «Поменять цвет окна».



Сгенерированный обработчик нажатия на кнопку button3 с пока пустым телом:

```

128 private://раздел закрытых членов класса
129 System::Void button1_Click(System::Object^ sender, System::EventArgs^ e)//сгенерировался метод-обработчик клика мыши по
130 { //метод принимает указатель на объект-отправитель события (sender), это объект самого базового (родительского) класса
131 //тело метода-обработчика пустое - его заполняет программист. Метод ничего не возвращает
132 label1->Text = "Кнопка работает.";
133 }
134 private: System::Void button2_Click(System::Object^ sender, System::EventArgs^ e)
135 {
136 //пустое тело метода-обработчика события клика по кнопке button2
137 label1->Text = "";
138 }
139 private: System::Void button3_Click(System::Object^ sender, System::EventArgs^ e)
140 {
141 }
142 }
143 };
144 }

```

Алгоритм данного обработчика следующий: если цвет окна – серый (базовый стандартный цвет окна по умолчанию), то поменять цвет окна на красный, иначе поменять цвет окна на серый. То

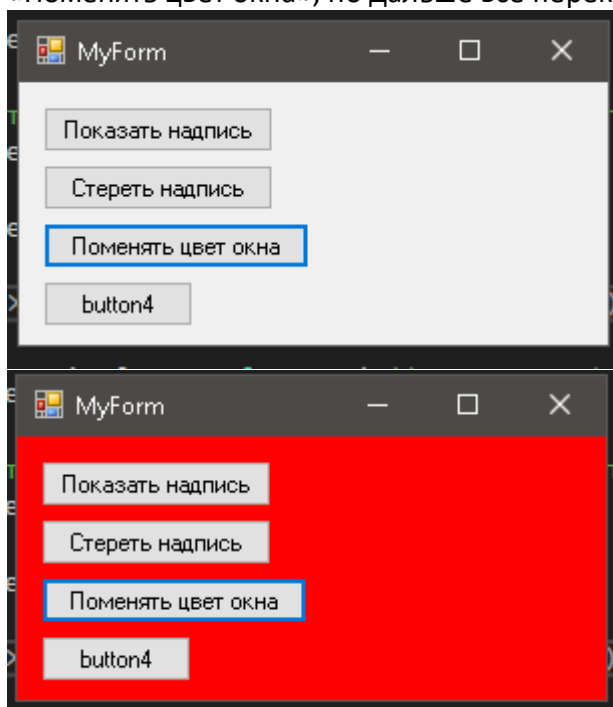
есть метод должен узнать цвет окна и поменять его на другой. Метод работает только с двумя цветами окна (фона окна). Для этого уже есть готовые классы и перечисления цветов фона окна.

```

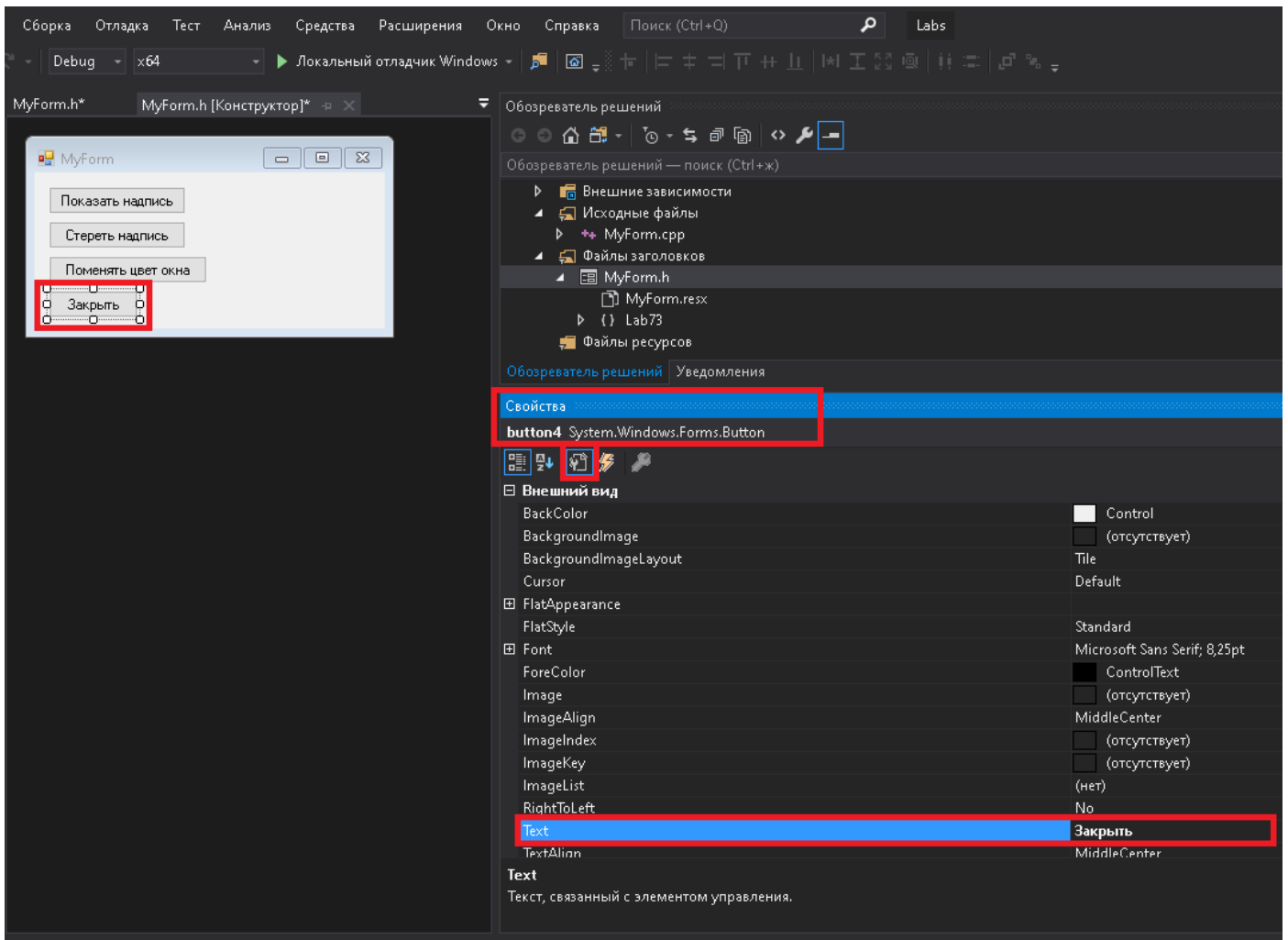
134 private: System::Void button2_Click(System::Object^ sender, System::EventArgs^ e)
135 {
136     //пустое тело метода-обработчика события клика по кнопке button2
137     label1->Text = "";
138 }
139 private: System::Void button3_Click(System::Object^ sender, System::EventArgs^ e)
140 {
141     if (this->BackColor == SystemColors::ButtonFace) //если цвет фона ЭТОГО окна "кнопочный серый" (это базовый стандартный цвет фона по умолчанию)
142     {
143         this->BackColor = Color::Red; //то сделать фоновым цветом ЭТОЙ оконной формы красный цвет, заданный в статическом классе Color
144     }
145     else //иначе
146     {
147         this->BackColor = SystemColors::ButtonFace; //назначить фоновым цветом ЭТОЙ оконной формы системный цвет "кнопочный серый"
148     }
149 }
150 }
151 }

```

Тестируем. В моем случае есть заминка программы с первым нажатием на кнопку «Поменять цвет окна», но дальше все переключения корректно чередуются.



Кнопке button4 присвоим надпись «Заккрыть» и создадим соответствующий обработчик, закрывающий данную основную оконную форму и, тем самым, все приложение.



Обработчик с телом, код которого закрывает запущенное окно приложения и, тем самым, убивает весь запущенный процесс данной программы:

```

140 private: System::Void button3_Click(System::Object^ sender, System::EventArgs^ e)
141 {
142     if (this->BackColor == SystemColors::ButtonFace)//если цвет фона ЭТОГО окна "кнопочный серый" (это базовый стандартный цвет фона по
143     {
144         this->BackColor = Color::Red;//то сделать фоновым цветом ЭТОЙ оконной формы красный цвет, заданный в статическом классе Color
145     }
146     else//иначе
147     {
148         this->BackColor = SystemColors::ButtonFace;//назначить фоновым цветом ЭТОЙ оконной формы системный цвет "кнопочный серый"
149     }
150 }
151 private: System::Void button4_Click(System::Object^ sender, System::EventArgs^ e)
152 {
153     this->Close();//метод, который закрывает ЭТУ оконную форму (текущее основное окно приложения)
154 }
155 }
156

```

Протестируйте, теперь приложение должно закрываться и по нажатии кнопки «Заккрыть», и по нажатии «крестика» в верхнем правом углу окна приложения. У этих кнопок теперь одинаковый функционал.

Ваш итоговый файл MyForm.h должен содержать примерно такой код:

```

#pragma once//стандартная директива препроцессору для заголовочного header-файла MyForm.h
//тела методов-обработчиков событий в этом файле программисту можно и нужно, но остальной
код автосгенерирован роботом и его нежелательно или запрещено изменять (робот при изменении
оконной формы в графическом Конструкторе все равно перегенерирует этот код и ваш код тут
уничтожится или будет противоречить остальному коду - будут ошибки). Но подписки на события
удалять можно
namespace Lab73//создано пространство имен нашего проекта Lab73 (символ _ автоматически
отфильтрован)

```

{//подключаются пространства имен Системы, Модели компонентов, Коллекций (для работы с массивами, списками и т.д.), пространство имен Forms для работы с оконными формами, Данными и Отрисовкой графических элементов. Из этих пространств имен нам доступны классы и их члены

```
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//открытый ссылочный класс
```

MyForm наследует классу Form, который находится в пространстве имен класса Forms, а он - в классе Windows, а он - в классе System. Наследование открытое. Что это значит?

```
{
    public://раздел открытых общедоступных членов класса MyForm
        MyForm(void)//конструктор без параметров по умолчанию
```

```
{
    InitializeComponent();//метод инициализации компонента, требуемый для
    работы конструктора MyForm()
    //
    //TODO: добавьте код конструктора
    //
}
```

```
protected://раздел защищенных членов класса MyForm
```

```
/// <summary>
```

```
/// Освободить все используемые ресурсы.
```

```
/// </summary>
```

```
~MyForm()//деструктор объекта класса MyForm
```

```
{
    if (components)//if (components != NULL)//если компоненты еще существуют
    (знаимают оперативную память)
    {
        delete components;//то удалить их из оперативной памяти
    }
}
```

private: System::Windows::Forms::Button^ button1;//при создании графических объектов в конструкторе они обязательно создаются в коде файла Myform.h в виде защищенных указателей на соответствующие объекты. Код Button^ button1 идентичен Button* button1, но при этом еще и поддерживает очищение объекта из памяти по достижении закрывающей скобки локальной области где он создан

```
protected:
```

```
private: System::Windows::Forms::Button^ button2;
```

```
private: System::Windows::Forms::Button^ button3;
```

```
private: System::Windows::Forms::Button^ button4;
```

```
private: System::Windows::Forms::Label^ label1;
```


private:

```
/// <summary>
```

```
/// Обязательная переменная конструктора.
```

```
/// </summary>
```

System::ComponentModel::Container ^components; //создан (объявлен) указатель на контейнер (вместилище) для всех компонентов оконной формы

#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) //полное определение метода InitializeComponent() с
```

телом

```
{ //this - это указатель на текущий объект - оконную форму MyForm, экземпляр класса
```

MyForm

```
this->button1 = (gcnew System::Windows::Forms::Button());
```

```
this->button2 = (gcnew System::Windows::Forms::Button());
```

```
this->button3 = (gcnew System::Windows::Forms::Button());
```

```
this->button4 = (gcnew System::Windows::Forms::Button());
```

```
this->label1 = (gcnew System::Windows::Forms::Label());
```

```
this->SuspendLayout();
```

```
//
```

```
// button1
```

```
//
```

```
this->button1->Location = System::Drawing::Point(12, 12);
```

```
this->button1->Name = L"button1";
```

```
this->button1->Size = System::Drawing::Size(115, 23);
```

```
this->button1->TabIndex = 0;
```

```
this->button1->Text = L"Показать надпись";
```

```
this->button1->UseVisualStyleBackColor = true;
```

```
this->button1->Click += gcnew System::EventHandler(this, &MyForm::button1_Click);
```

```
//
```

```
// button2
```

```
//
```

```
this->button2->Location = System::Drawing::Point(12, 41);
```

```
this->button2->Name = L"button2";
```

```
this->button2->Size = System::Drawing::Size(115, 23);
```

```
this->button2->TabIndex = 1;
```

```
this->button2->Text = L"Стереть надпись";
```

```
this->button2->UseVisualStyleBackColor = true;
```

```
this->button2->Click += gcnew System::EventHandler(this, &MyForm::button2_Click);
```

```
//
```

```
// button3
```

```
//
```

```
this->button3->Location = System::Drawing::Point(12, 70);
```

```
this->button3->Name = L"button3";
```

```
this->button3->Size = System::Drawing::Size(133, 23);
```

```
this->button3->TabIndex = 2;
```

```
this->button3->Text = L"Поменять цвет окна";
```

```

this->button3->UseVisualStyleBackColor = true;
this->button3->Click += gcnew System::EventHandler(this, &MyForm::button3_Click);
//
// button4
//
this->button4->Location = System::Drawing::Point(12, 99);
this->button4->Name = L"button4";
this->button4->Size = System::Drawing::Size(75, 23);
this->button4->TabIndex = 3;
this->button4->Text = L"Заккрыть";
this->button4->UseVisualStyleBackColor = true;
this->button4->Click += gcnew System::EventHandler(this, &MyForm::button4_Click);
//
// label1
//
this->label1->AutoSize = true;
this->label1->Location = System::Drawing::Point(145, 17);
this->label1->Name = L"label1";
this->label1->Size = System::Drawing::Size(0, 13);
this->label1->TabIndex = 4;
//
// MyForm
//
this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
this->ClientSize = System::Drawing::Size(294, 131);
this->Controls->Add(this->label1);
this->Controls->Add(this->button4);
this->Controls->Add(this->button3);
this->Controls->Add(this->button2);
this->Controls->Add(this->button1);
this->Name = L"MyForm";
this->Text = L"MyForm";
this->ResumeLayout(false);
this->PerformLayout();

```

```

}

```

```

#pragma endregion

```

```

private://раздел закрытых членов класса

```

```

    System::Void button1_Click(System::Object^ sender, System::EventArgs^ e)//сгенерировался
    метод-обработчик клика мыши пользователя по кнопке button1

```

```

    { //метод принимает указатель на объект-отправитель события (sender), это объект самого
    базового (родительского) класса Object, и указатель на список входных аргументов, которые объект-
    отправитель посылает объекту-получателю сообщения о свершении события

```

```

        //тело метода-обработчика пустое - его заполняет программист. Метод ничего не
        возвращает

```

```

        label1->Text = "Кнопка работает.";

```

```

    }

```

```

private: System::Void button2_Click(System::Object^ sender, System::EventArgs^ e)

```

```

{

```

```

//пустое тело метода-обработчика события клика по кнопке button2
label1->Text = "";
}
private: System::Void button3_Click(System::Object^ sender, System::EventArgs^ e)
{
    if (this->BackColor == SystemColors::ButtonFace)//если цвет фона ЭТОГО окна "кнопочный
серый" (это базовый стандартный цвет фона по умолчанию)
    {
        this->BackColor = Color::Red;//то сделать фоновым цветом ЭТОЙ оконной формы
красный цвет, заданный в статическом классе Color
    }
    else//иначе
    {
        this->BackColor = SystemColors::ButtonFace;//назначить фоновым цветом ЭТОЙ оконной
формы ситемный цвет "кнопочный серый"
    }
}
private: System::Void button4_Click(System::Object^ sender, System::EventArgs^ e)
{
    this->Close();//метод, который закрывает ЭТУ оконную форму (текущее основное окно
приложения)
}
};
}

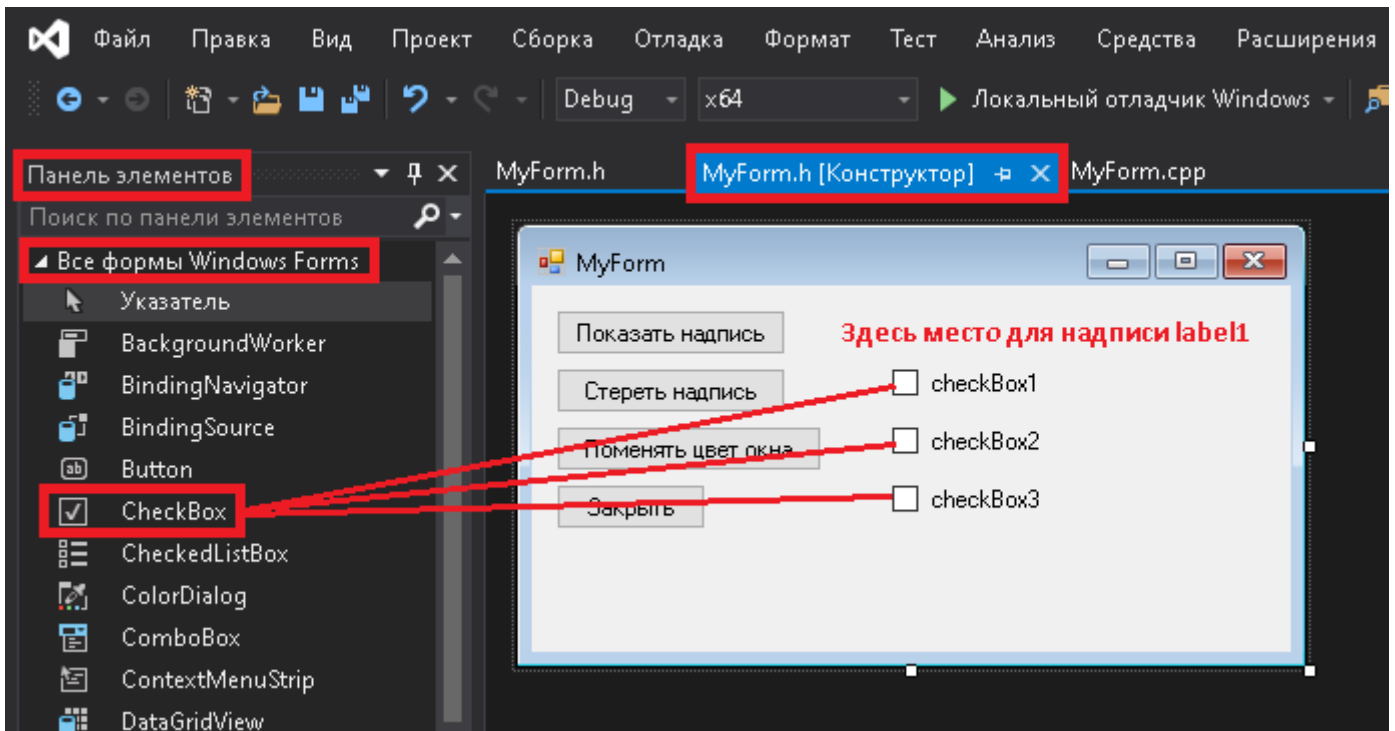
```

Задание 3

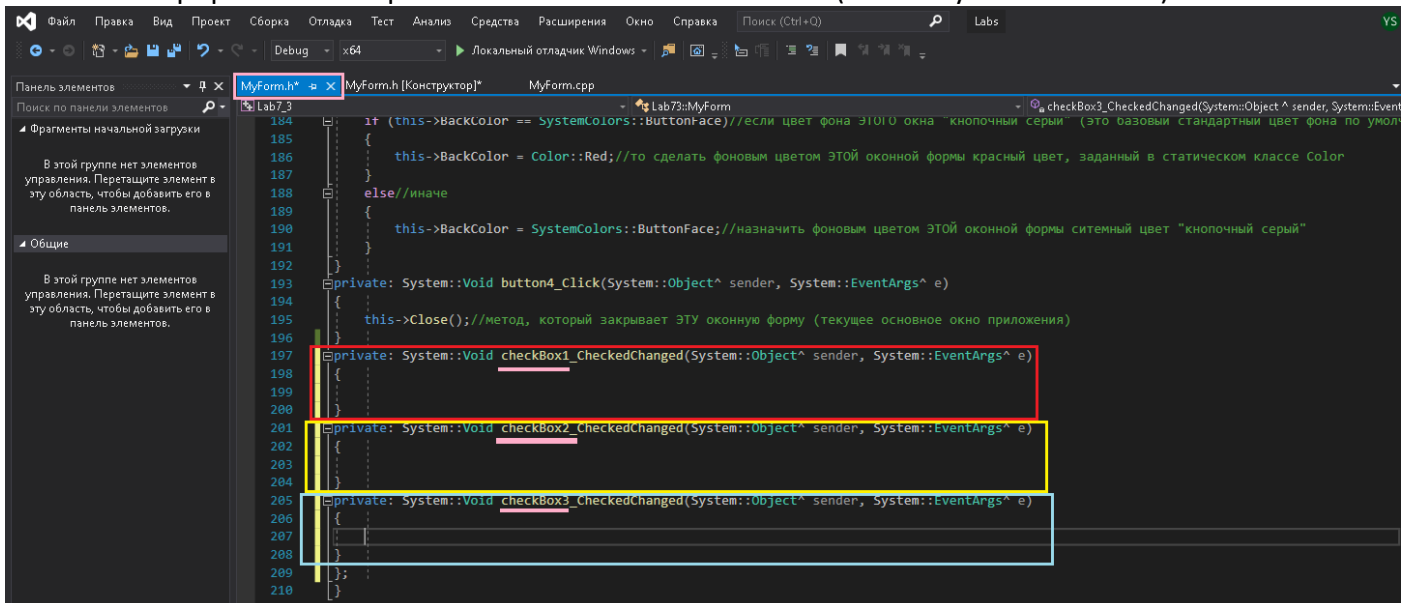
Из Панели инструментов (Toolbox) можно поместить на форму и другие элементы, причем некоторые из них не являются видимыми. Для этого нужно растянуть окошко нашего приложения в режиме Дизайнера (**работающее (запущенное на выполнение) окно тестируемого приложения должно быть закрыто**).

Поместим на форму три CheckBox`а (чекбокс – элемент, по клику в «квадратике» которого появляется флажок («галочка») и срабатывает код обработчика данного чекбокса. Чекбокс позволяет настраивать, переключать, указывать приложению какие-либо обстоятельства, которые меняются от статуса самого чекбокса (выделен – не выделен, то есть по сути к true – false). В Дизайнере дважды кликнем по первому чекбоксу и сделаем ему обработчик с телом, таким же, как и у кнопки «Поменять цвет окна», но поменяйте код в блоках if-else местами. На остальные два созданных чекбокса тоже создайте обработчики события клика мышью по ним. Пусть эти два чекбокса отвечают за присвоение оконной форме зеленого и желтого цветов соответственно.

В режиме графического Конструктора располагаем на макет оконной формы три чекбокса. Помните об элементе label1, у которого изначально не отображается надпись, но он есть на форме!



Сгенерированные обработчики нажатия на чекбоксы (пока с пустыми телами):



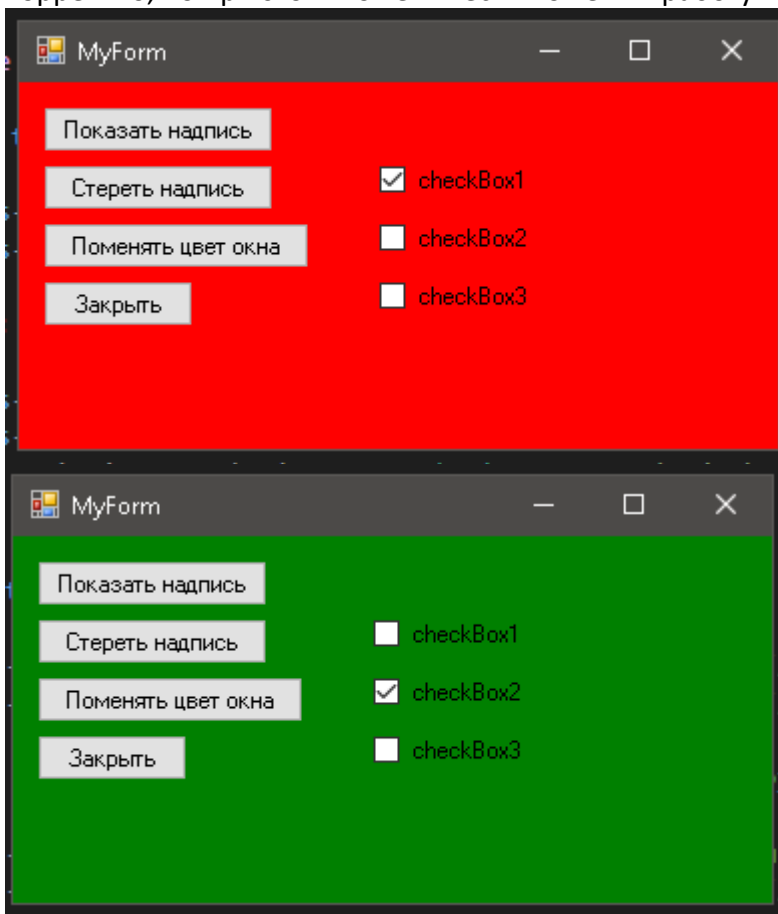
Заполняем тела обработчиков нажатия мышью по чекбоксам. Учитываем, что чекбоксы можно нажимать много раз, а по нашей логике цвет устанавливает тот чекбокс, который нажат последним, поэтому в коде тел чекбоксов предусматриваем присвоение остальным двум чекбоксам статуса «НЕ нажатый», чтобы у них не отображались «галочки». С другой стороны, если мы нажмем по одному чекбоксу четное число раз подряд, цвет формы будет определять он, но «галочки» он визуально иметь не будет. Почему? Также учитываем, что кнопка «Поменять цвет окна» может менять цвет окна наравне с чекбоксами, поэтому допишите в ней код, чтобы по нажатию на нее не только менялся цвет у формы, но еще и всем трем чекбоксам присваивался статус «НЕ нажатый». Почему это можно написать только сейчас?

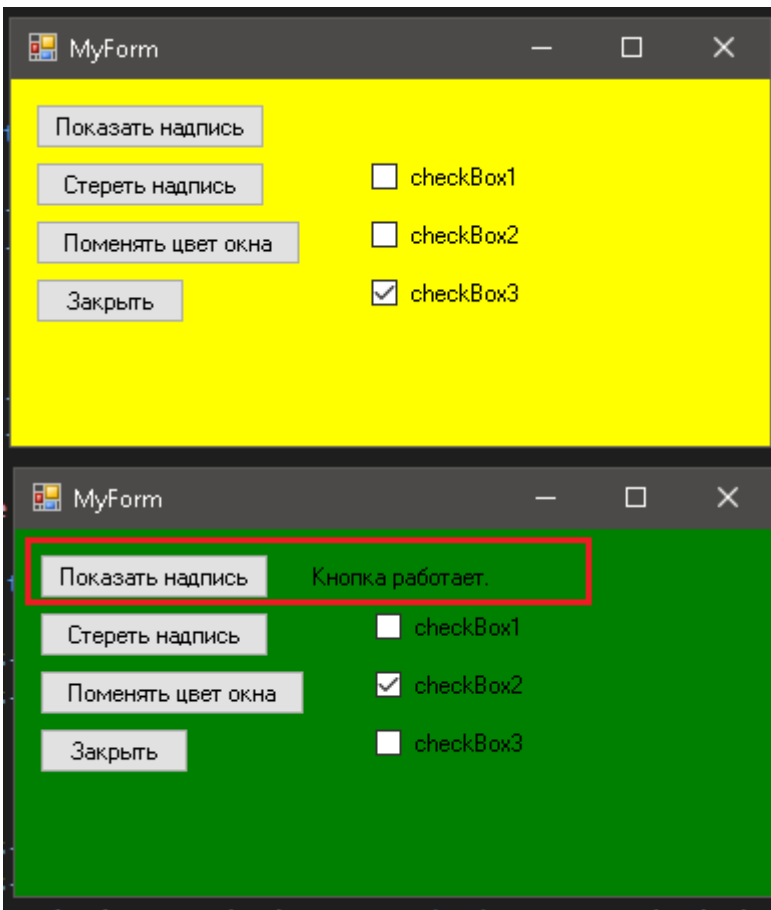
```

193 private: System::Void button4_Click(System::Object^ sender, System::EventArgs^ e)
194 {
195     this->Close(); //метод, который закрывает ЭТУ оконную форму (текущее основное окно приложения)
196 }
197 private: System::Void checkBox1_CheckedChanged(System::Object^ sender, System::EventArgs^ e) //обработчик нажатия на чекбокс1
198 { //код такой проверки сразу работает корректно
199     if (this->BackColor == Color::Red)
200     {
201         this->BackColor = SystemColors::ButtonFace;
202     }
203     else
204     {
205         this->BackColor = Color::Red;
206     }
207     this->checkBox2->CheckState = CheckState::Unchecked; //всем остальным чекбоксам присваиваем статус нажатия "НЕ нажато", чтобы предупредить
208     this->checkBox3->CheckState = CheckState::Unchecked; //случай, если другие чекбоксы были ранее нажаты и отображают неактуальные "галочки"
209 }
210 private: System::Void checkBox2_CheckedChanged(System::Object^ sender, System::EventArgs^ e) //обработчик нажатия на чекбокс2
211 {
212     this->BackColor = Color::Green; //присвоить фону формы зеленый цвет из значений класса System::Drawing::Color. Сразу пишу Color, поскольку
213     this->checkBox1->CheckState = CheckState::Unchecked; //в строке кода № 10 уже подключено пространство имен using namespace System::Drawing;
214     this->checkBox3->CheckState = CheckState::Unchecked;
215 }
216 private: System::Void checkBox3_CheckedChanged(System::Object^ sender, System::EventArgs^ e) //обработчик нажатия на чекбокс3
217 {
218     this->BackColor = Color::Yellow; //присвоить фону формы желтый цвет из значений класса System::Drawing::Color.
219     this->checkBox1->CheckState = CheckState::Unchecked; //всем остальным чекбоксам присваиваем статус нажатия "НЕ нажато", чтобы предупредить
220     this->checkBox2->CheckState = CheckState::Unchecked; //случай, если другие чекбоксы были ранее нажаты и отображают неактуальные "галочки"
221 }
222 };
223

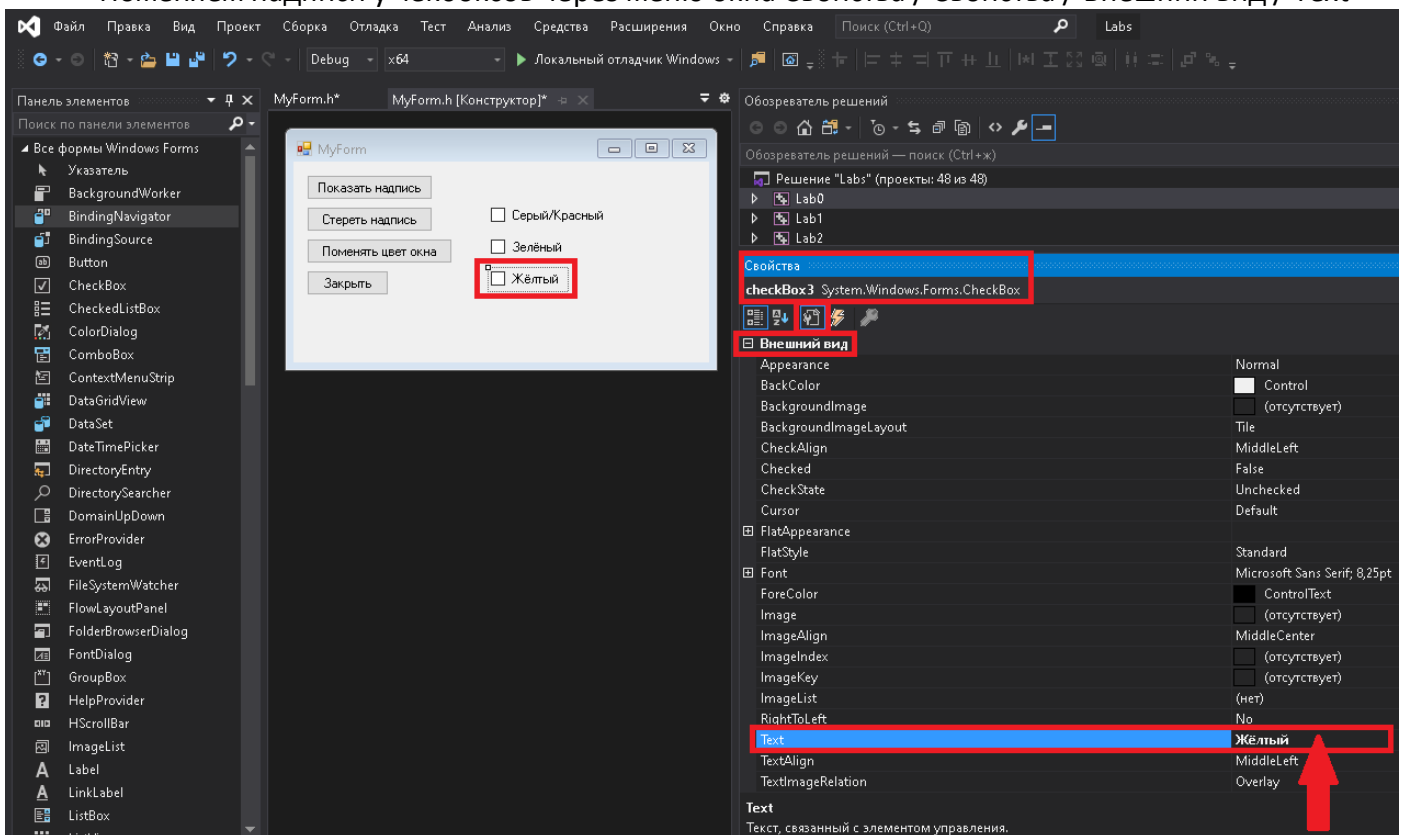
```

Тестируем работу трех чекбоксов и проверяем функционал уже ранее протестированных элементов на оконной форме (это **регрессионное тестирование** и оно лучше просто **тестирования нового функционала**, поскольку новодобавленный код в приложение сам по себе может работать корректно, но при этом может внести помехи в работу «старого» кода).

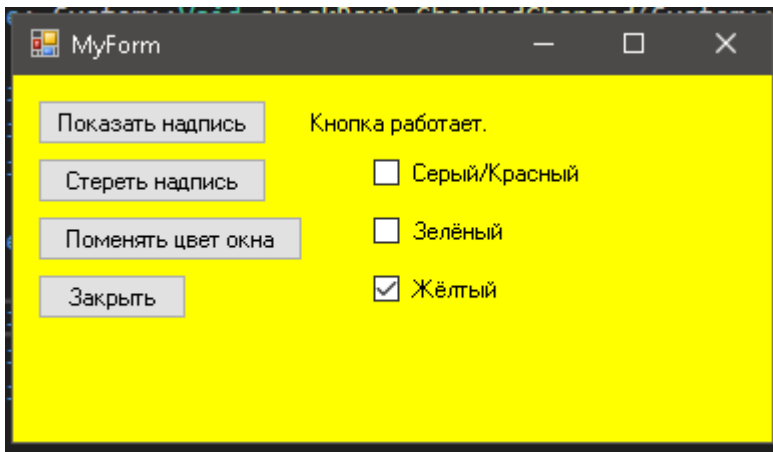




Поменяем надписи у чекбоксов через меню окна Свойства / Свойства / Внешний вид / Text



Тестируем. Программа должна стать более дружелюбной к пользователю – чекбоксы имеют осмысленные надписи, при этом сами данные переменные сохранили свои имена в коде checkBox1, checkBox2 и checkBox3. Если нужно поменять имя переменной в коде, то делаем это не в коде файлов, а в меню окна Свойства / Свойства / Разработка / (Name) ставим справа курсор и вводим с клавиатуры новое имя указателю на объект.



Все чекбоксы работают, но эффект оказывает тот из них, который нажат последним, а «галочки» с остальных мы убирали в коде тел обработчиков нажатия на чекбоксы. Но это проще сделать, используя элемент-контейнер группбокс (GroupBox – элемент-контейнер, фактически визуально это рамка, которая отделяет находящиеся в ней элементы от остальных; помещенные в группбокс элементы как бы помещаются в отдельную локальную область и уже не связаны с другими элементами, не помещенными в этот же группбокс), внутри которого мы поместим элементы радиобаттоны (RadioButton – элемент-переключатель, переключаемая кнопка, причем по умолчанию из группы радиобаттонов может быть выделен **только один** радиобаттон. При выделении другого радиобаттона остальные радиобаттоны автоматически возвращаются в невыделенное состояние. Разумеется, ОС должна знать, какие радиобаттоны находятся в одной группе, то есть они взаимосвязаны между собой. Для этого радиобаттоны нужно помещать в контейнер GroupBox, который объединяет несколько радиобаттонов и отделяет их от других элементов на форме, ведь на ней могут быть и другие группы радиобаттонов.

Поместите на форму из Панели элементов элемент GroupBox, а в него поместите два радиобаттона. Создайте для каждого радиобаттона по обработчику события клика на него мышью и поместите в них код, изменяющий заголовок формы (отображается вверху рамки оконной формы нашего приложения).

Панель элементов

- FontDialog
- GroupBox**
- HelpProvider
- HScrollBar
- ImageList
- Label
- LinkLabel
- ListBox
- ListView
- MaskedTextBox
- MenuStrip
- MessageQueue
- MonthCalendar
- NotifyIcon
- NumericUpDown
- OpenFileDialog
- PageSetupDialog
- Panel
- PerformanceCounter
- PictureBox
- PrintDialog
- PrintDocument
- PrintPreviewControl
- PrintPreviewDialog
- Process
- ProgressBar
- PropertyGrid
- RadioButton**
- RichTextBox
- SaveFileDialog

MyForm.h [Конструктор]*

MyForm

Показать надпись
Стереть надпись
Поменять цвет окна
Заккрыть

☐ Серый/Красный
☐ Зелёный
☐ Жёлтый

groupBox1

☐ radioButton1
☐ radioButton2

MyForm.h [Конструктор]

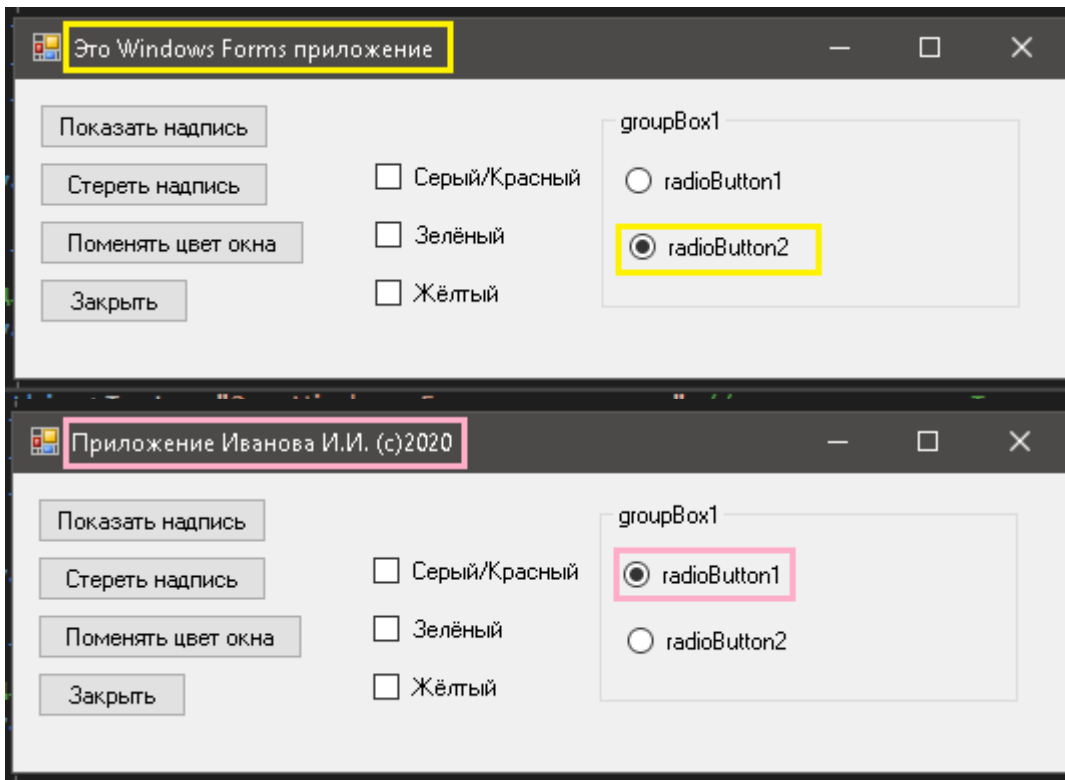
```

258 this->checkBox1->CheckedState = CheckState::Unchecked;
259 this->checkBox3->CheckedState = CheckState::Unchecked;
260
261 private: System::Void checkBox3_CheckedChanged(System::Object^ sender, System::EventArgs^ e) //обработчик нажатия на чекбокс3
262 {
263     this->BackColor = Color::Yellow; //присвоить фону формы желтый цвет из значений класса System::Drawing::Color.
264     this->checkBox1->CheckedState = CheckState::Unchecked; //всем остальным чекбоксам присваиваем статус нажатия "НЕ нажато", чтобы предупредить
265     this->checkBox2->CheckedState = CheckState::Unchecked; //случаи, если другие чекбоксы были ранее нажаты и отображают неактуальные "галочки"
266 }
267
268 private: System::Void radioButton1_CheckedChanged(System::Object^ sender, System::EventArgs^ e) //обработчик события клика мышки по radioButton1
269 {
270     this->Text = "Приложение Иванова И.И. (с)2020"; //присвоить полю Текст ЭТОЙ оконной формы строку символов "Приложение Иванова И.И. (с)2020"
271     //для текущей главной оконной формы есть только одного окна, и указатель на него - this
272 }
273 private: System::Void radioButton2_CheckedChanged(System::Object^ sender, System::EventArgs^ e) //обработчик события клика мышки по radioButton2
274 {
275     this->Text = "Это Windows Forms приложение"; //присвоить полю Текст ЭТОЙ оконной формы строку символов "Это Windows Forms приложение"
276 }
  
```

110% Проблемы Вывод Показать выходные данные

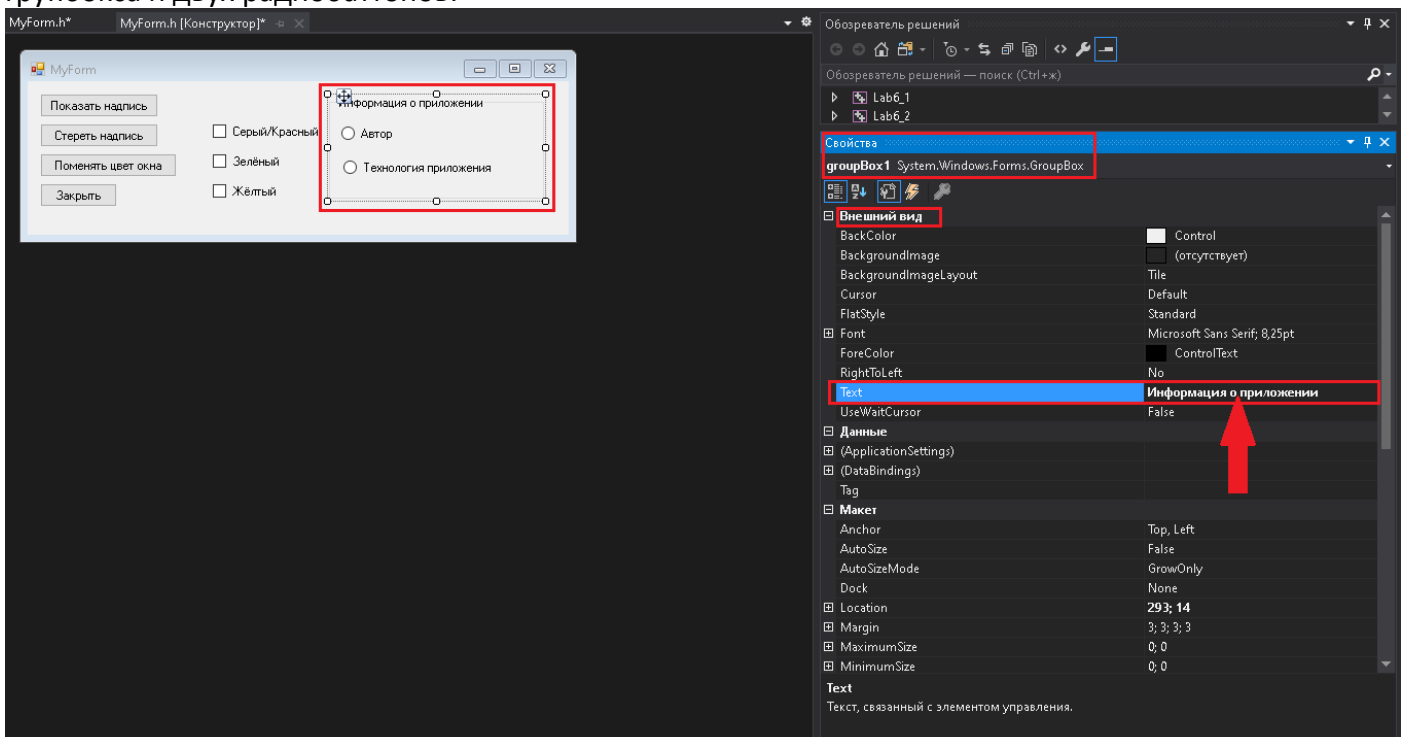
Стр: 273 Симв: 134 Столб: 137 Табуляция

Тестируем:



Работет корректно, причем из двух радиобаттнов может быть выделен только один и это не нужно нам отслеживать и прописывать в коде, а это неотъемлемое свойство радиобаттнов как элементов графического интерфейса пользователя. В группе может быть больше двух радиобаттнов, но активным будет один последний нажатый радиобаттон из группы. Если вы создавали радиобаттоны и пришли к выводу, что их надо поделить на независимые друг от друга группы (то есть сделать независимыми некоторые радиобаттоны от некоторых других, чтобы в одно время могли быть выделены несколько радиобаттонов на оконной форме, а не один), то вам нужно создать один или несколько группбоксов и перетащить ваши радиобаттоны в эти группбоксы.

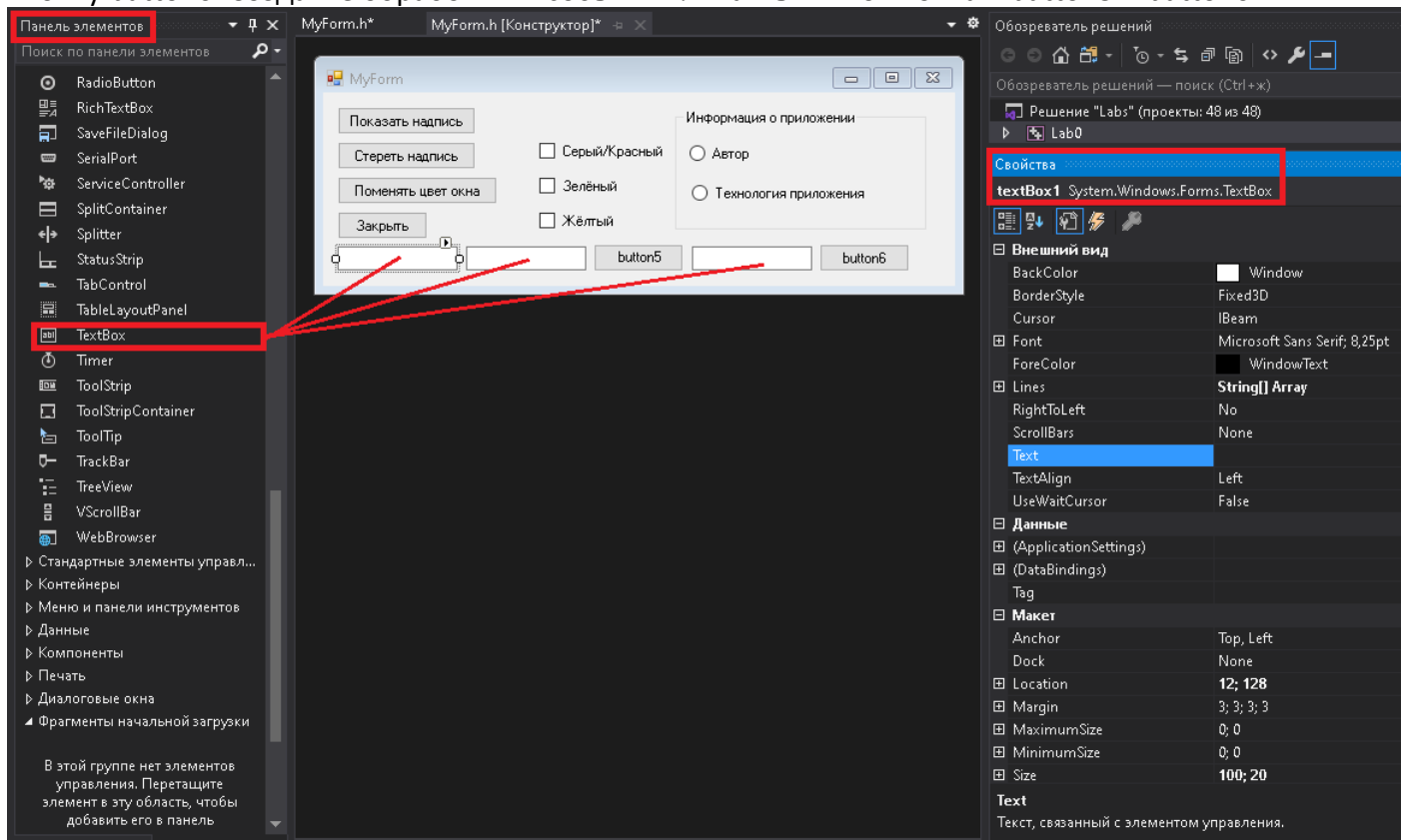
Теперь с помощью окна Свойства/Свойства/Внешний вид/Text введем названия для группбоксов и двух радиобаттонов:



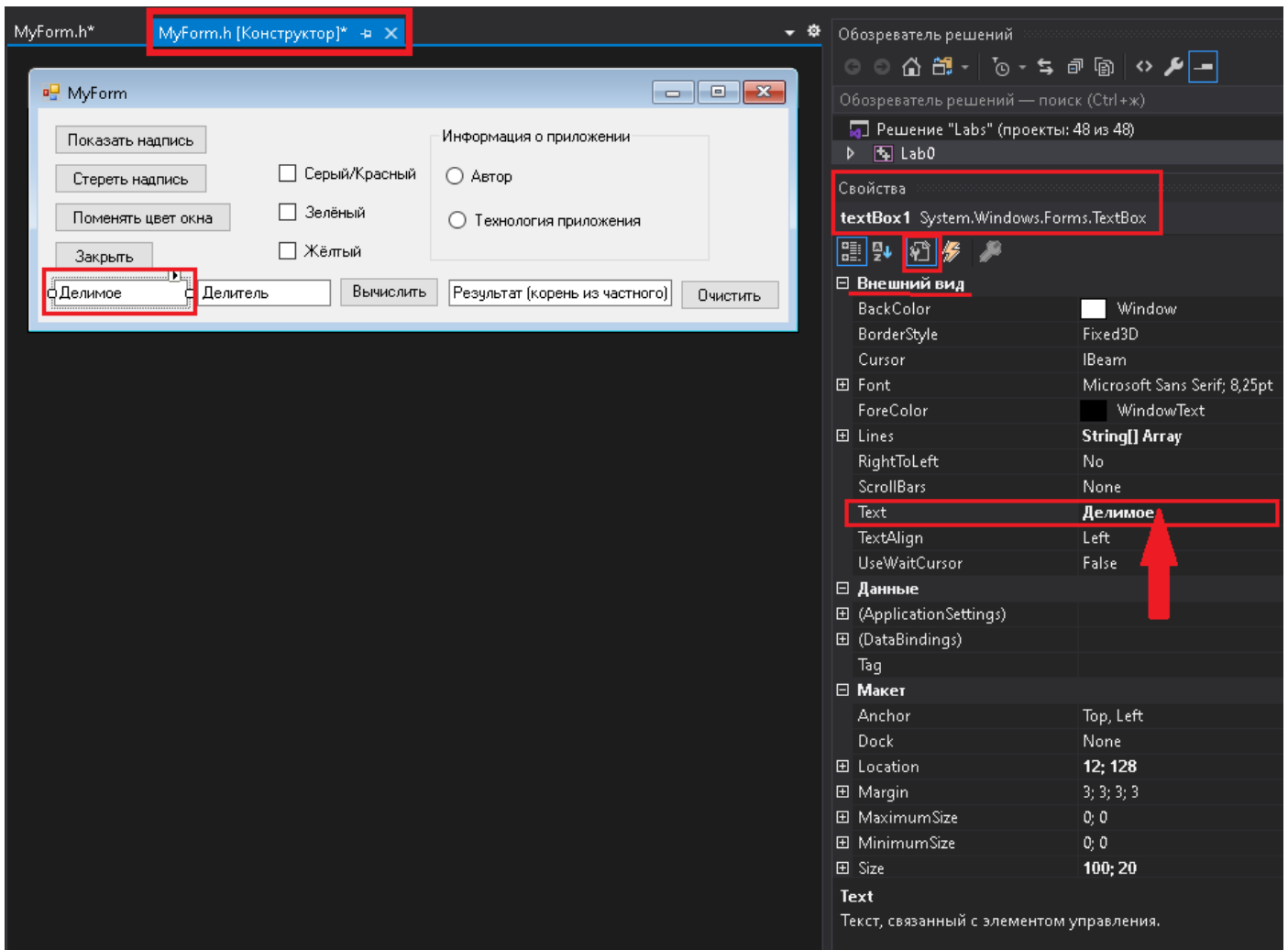
Если надо запрашивать у пользователя данные или выводить ему некоторую информацию в специальном поле, то для этого удобно использовать текстовые поля. `TextBox` – это поле для отображения

и хранения текста (символов), который туда может быть помещен кодом программы или введен с клавиатуры пользователем.

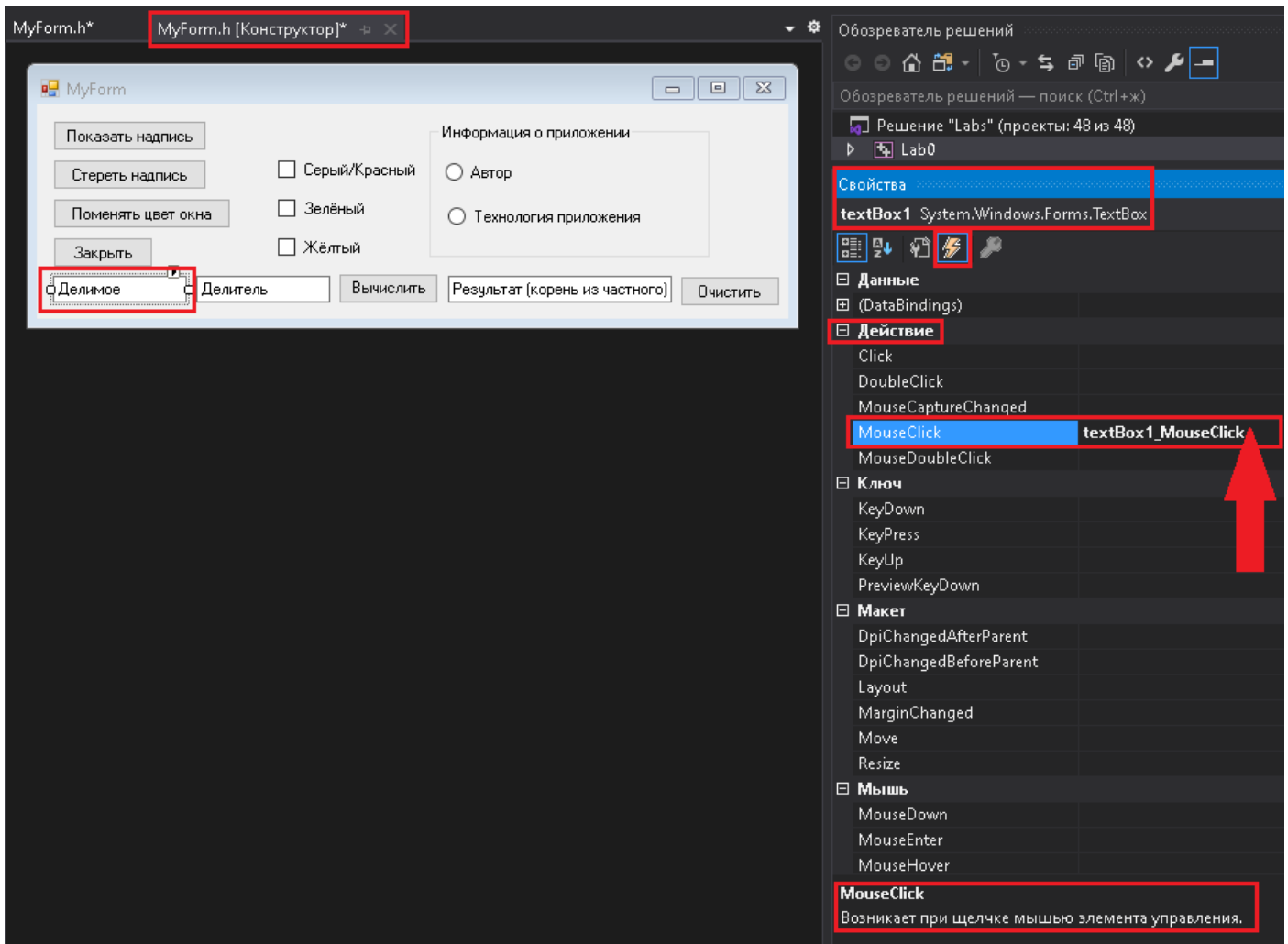
Поместим на оконную форму три текстовых поля и еще две кнопки Button. Расположите слева textBox1, правее от него – textBox2, правее – кнопку button5, правее – textBox3 и еще правее – кнопку button6. Создайте обработчики события клика мыши по кнопкам button5 и button6.



Оформим новые элементы. Перед или над текстовым полем можно расположить элемент Надпись Label и тем самым объяснять пользователю какие данные и куда ему нужно вводить. Но можно сделать иначе: мы в текстовом поле расположим фразу, которая будет объяснять пользователю, для чего этот текстовый поле. Для этого надо в меню окна Свойства/Свойства/Внешний вид/Text ввести фразу «Введите вещественное число», «Ведите делимое» или просто «Делимое». Аналогично с другим текстовыми полями и двумя новыми кнопками.



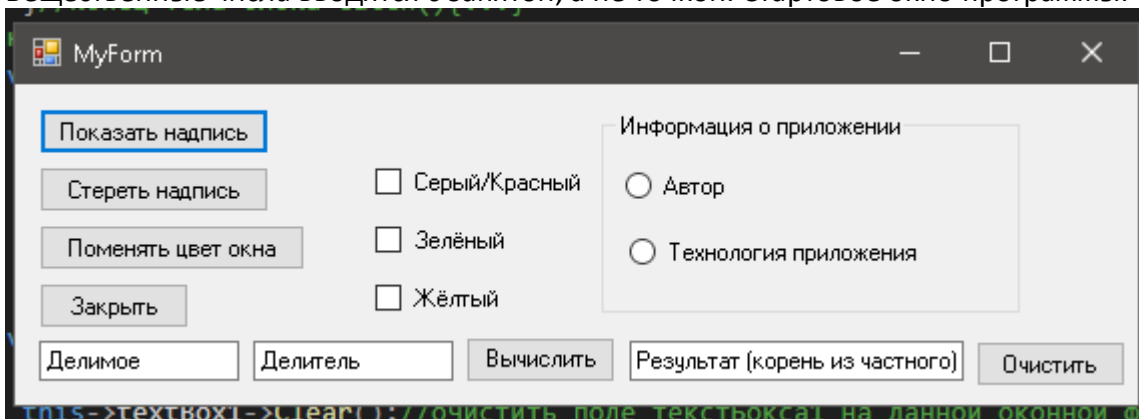
В текстовых полях пользователь может вводить данные, установив в него курсор, но с учетом наличия в текстовых полях фраз-приглашений, пользователю нужно будет сначала кнопками Backspace или Delete удалить символы наших фраз-приглашений. Сделаем лучше: пусть, когда пользователь ставит курсор, то есть происходит событие КликМышью в данном текстовом поле, мы будем отлавливать наступление данного события в нашем приложении и сразу удалять все символы из данного текстового поля, чтобы пользователь мог сразу вводить свои данные (вещественное число). Для этого в меню окна Свойства/События(кнопка Молния)/Действие/MouseClick дважды нажмем мышью справа, чтобы появился обработчик данного текстового поля, например, textBox1_MouseClick. Аналогичный обработчик на событие создайте для textBox2. Для textBox3 такой обработчик не нужен, поскольку там должен появиться результат вычислений тогда, когда пользователь нажмет на кнопку «Вычислить».



Итак, в режиме редактирования кода в файле MyForm.h мы должны увидеть четыре новых обработчика, тела которых заполним нижеследующим кодом. Обработчик события нажатия мышью по кнопке button5 «Вычислить» реализует алгоритм считывания значений из полей текстовых полей textBox1 и textBox2, конвертации этих считанных строк символов в вещественные числа (вещественное число вводится с разделителем «запятая» между целой и вещественной частями числа), проверкой возможности деления на ноль или извлечения корня из отрицательного числа (что запрещено в обычной математике), поэтому в таком случае мы вызываем мини-окно Сообщения MessageBox с сообщением об ошибке, очищаем поля от некорректных данных), и если данные прошли проверки (то есть данные корректны), то делаем вычисления и помещаем их в вещественную переменную. Далее значение из переменной-результата превращаем в строку символов и помещаем эту строку в textBox3, где должен отобразиться результат. Еще полученный результат можно в строковом виде поместить в название (заголовок) основной оконной формы. В наших примерах используется **управляемый код**, то есть код, который контролируется и обслуживается Сборщиком мусора Garbage Collector. Мы создаем динамические объекты и указатели на них, но вместо символа «звездочка» * используем здесь символ «суффикса» ^ с таким же и даже более расширенным функционалом (^ означает обязанность удаления таких объектов Сборщиком мусора по достижении окончания локальной области кода, в которой они созданы (это закрывающая скобка или конец файла)), а символ % означает & (ссылку)).

```
private: System::Void button5_Click(System::Object^ sender, System::EventArgs e)//обработчик события нажатия на кнопку "Вычислить"
{
    try//это блок try{}-catch(), отлавливающий в коде блока try(...) ошибки типов, указанные в блоке catch()
    {
        double a = Convert::ToDouble(this->textBox1->Text);//у статического класса Convert(т.е.Конвертация, перевод данных из одного типа данных в другой) вызываем метод ToDouble(),
        //то есть приведем входное значение, которое берется из текстового поля, к вещественному типу. После конвертации вещественный результат справа помещаем в вещественную переменную,
        double b = Convert::ToDouble(this->textBox2->Text);//созданную слева
        if (b == 0 || a / b < 0)//если делитель равен нулю или хотя бы результат деления меньше нуля (значит, из него НЕЛЬЗЯ извлечь квадратный корень)
        {
            //то выводим мини-окно Сообщения с заголовком "Ошибка!" и текстом "Попытка деления на ноль или извлечения корня из отрицательного числа."
            MessageBox::Show("Попытка деления на ноль или извлечения корня из отрицательного числа.", "Ошибка!");
            button6_Click(sender, System::EventArgs::Empty);//после закрытия окна Сообщения, очищаем поля от старых ошибочных данных, вызвав обработчик кнопки "Очистить"
        }
        //обработчик события должен принимать имя отправителя, создающего событие, и передаваемые данные (тут передаем ничего - Empty)
        else//если проверку на математическую корректность данные прошли успешно
        {
            //то можно разделить делимое на делитель и извлечь из этого результата квадратный корень, который берется из статического класса Math, помещая итоговый результат в
            double c = Math::Sqrt(a / b);//вещественную переменную double c
            this->textBox3->Clear();//очищаем поле textBox3 от предыдущего любого текста
            this->textBox3->Text = c.ToString();//вещественное число приводим к строковому виду (символьный массив) и помещаем в поле textBox3
            this->Text = c.ToString();//вещественное число приводим к строковому виду (символьный массив) и помещаем в качестве заголовка основного окна
        }
    }
    //конец блока try - за ним сразу должен следовать блок catch(...), отлавливающий ошибки конкретного типа
    catch (System::FormatException^ e)//отлавливать ошибки типа System::FormatException, то есть ошибки, связанные с неподходящим форматом (типом) данных
    {
        //если возникла ошибка неподходящего форматирования данных (данные не соответствуют ожидаемому от них типу данных), то
        MessageBox::Show(e->Message->ToString(), "Ошибка формата данных!");//вывести мини-окно Сообщения с заголовком "Ошибка формата данных!" и текстом сути ошибки форматирования
        button6_Click(sender, System::EventArgs::Empty);//после закрытия окна Сообщения очистить поля от старых ошибочных данных, вызвав обработчик кнопки "Очистить"
    }
    //конец тела блока catch(...)
}
//конец тела обработчика события клика по кнопке "Вычислить"
private: System::Void button6_Click(System::Object^ sender, System::EventArgs e)//обработчик события нажатия на кнопку "Очистить"
{
    this->textBox1->Clear();//очистить поле textBox1 на данной оконной форме
    this->textBox2->Clear();//очистить поле textBox2 на данной оконной форме
    this->textBox3->Clear();//очистить поле textBox3 на данной оконной форме
    this->Text = "Результат (корень из частного)";//поместить в поле уже очищенного textBox3 фразу (символы)
}
private: System::Void textBox1_MouseClick(System::Object^ sender, System::Windows::Forms::EventArgs e)//обработчик события клика мышью по textBox1
{
    this->textBox1->Clear();//очистить поле textBox1 на данной оконной форме
}
private: System::Void textBox2_MouseClick(System::Object^ sender, System::Windows::Forms::EventArgs e)//обработчик события клика мышью по textBox2
{
    this->textBox2->Clear();//очистить поле textBox2 на данной оконной форме
}
};
```

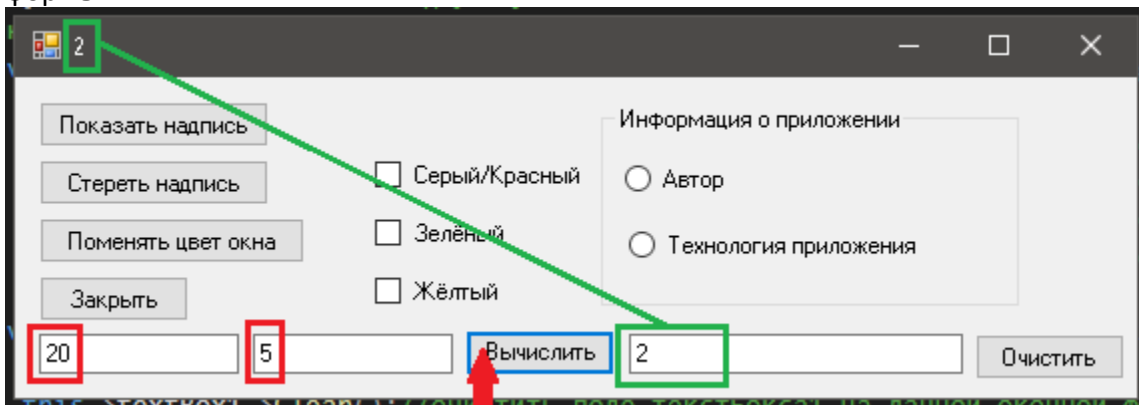
Компилируем и тестируем на корректных вводимых данных (позитивное тестирование). Вещественные числа вводятся с запятой, а не точкой. Стартовое окно программы:



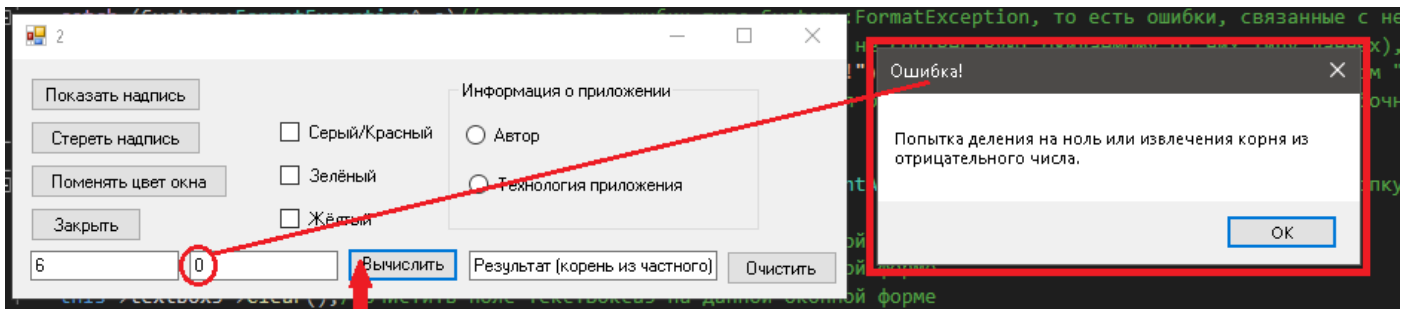
Вводим корректные целые данные (они входят во множество вещественных данных):

$20 / 5 = 4$;

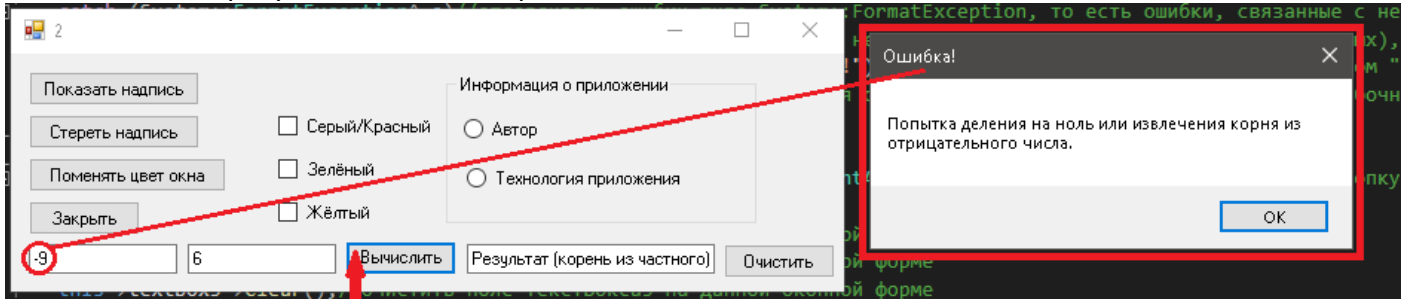
$\sqrt[2]{4} = 2$; Корректный результат отобразился и в текстовом поле 3 и в заголовке главной оконной формы



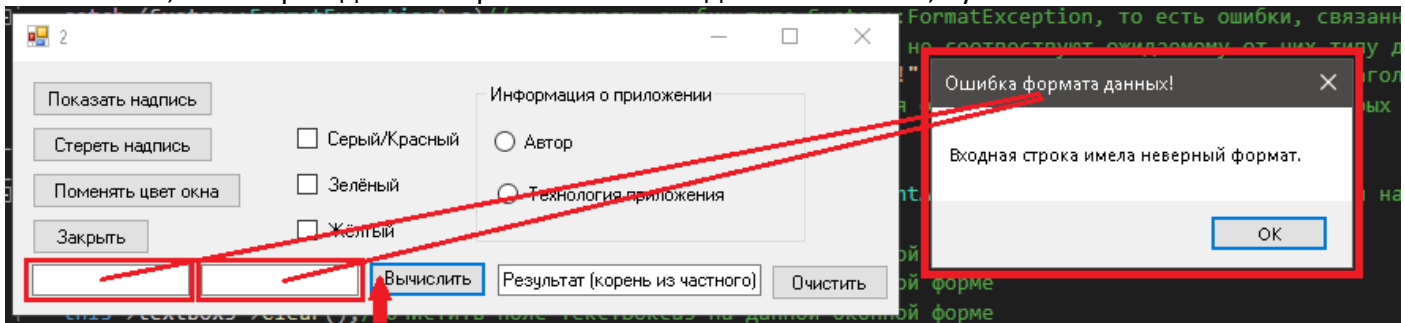
Попытка делить на ноль. Помните, что окно Сообщения является модальным (обязательным для закрытия) – то есть не закрыв его, вы не сможете работать с основным окном своего приложения.



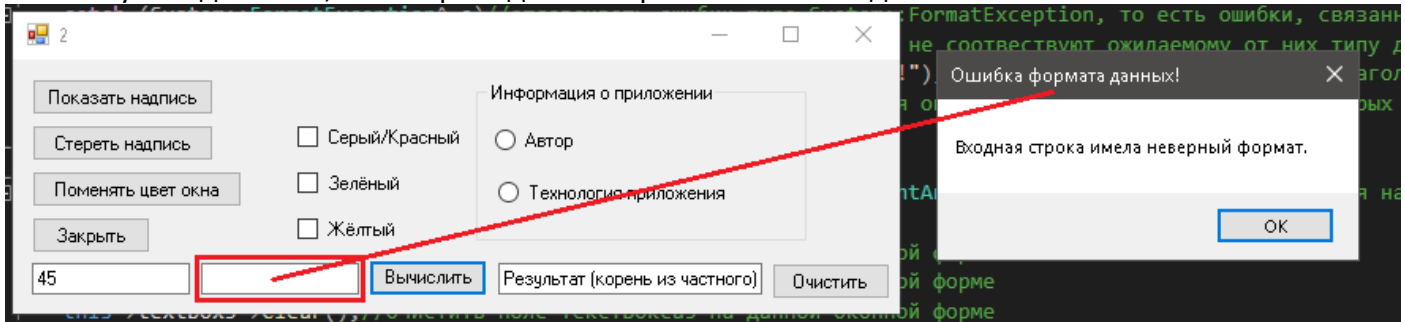
Сделаем результат деления отрицательным, введя - 9:



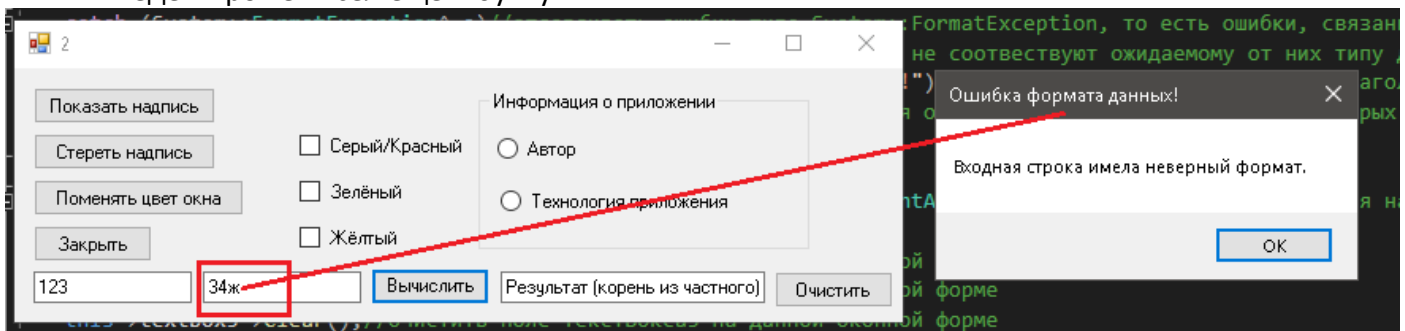
Поля, из которых должны браться значения для вычислений, пусты:



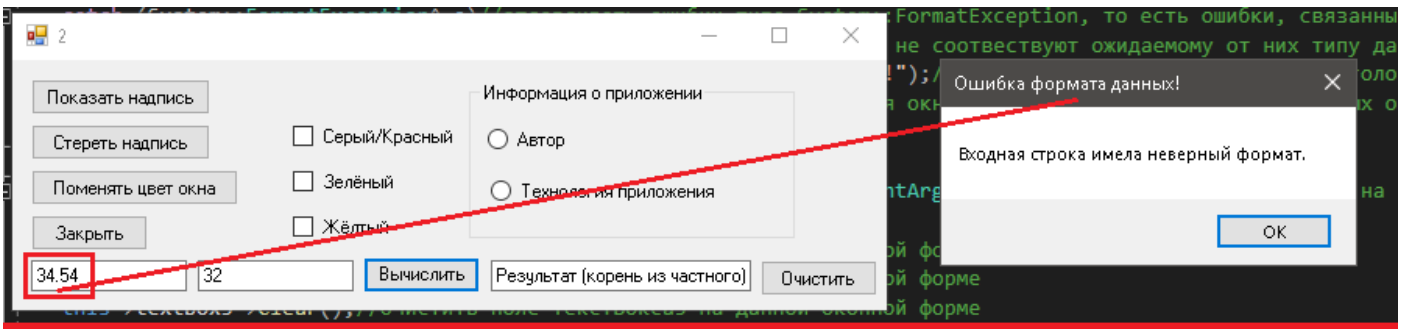
Пусто одно поле, из которого должно браться значение для вычислений:



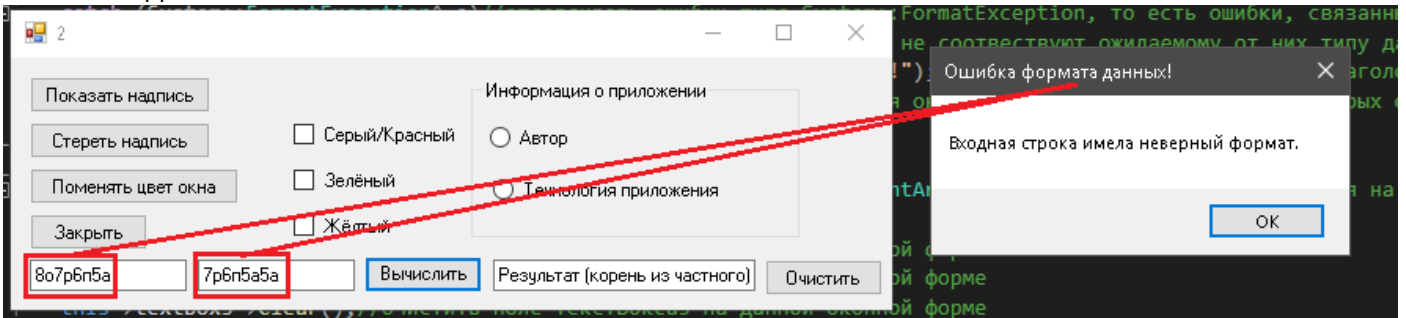
Введем кроме чисел еще и букву:



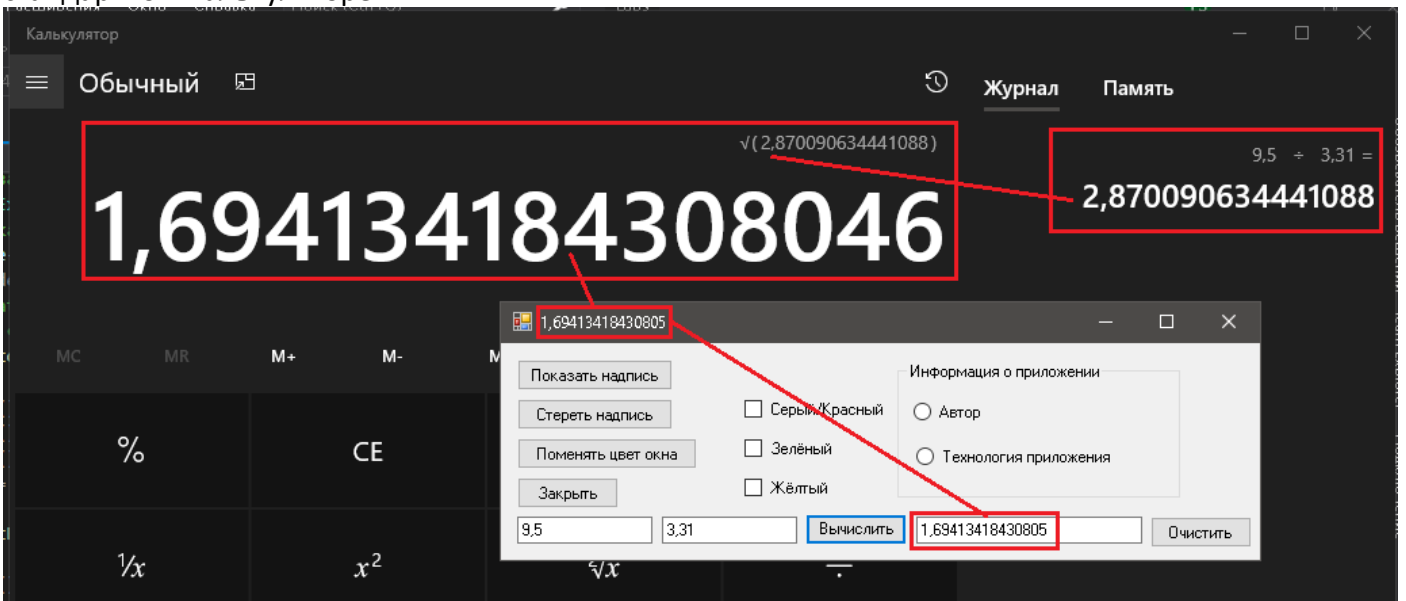
Вводим не запятую, а точку:



Введем любые символы:

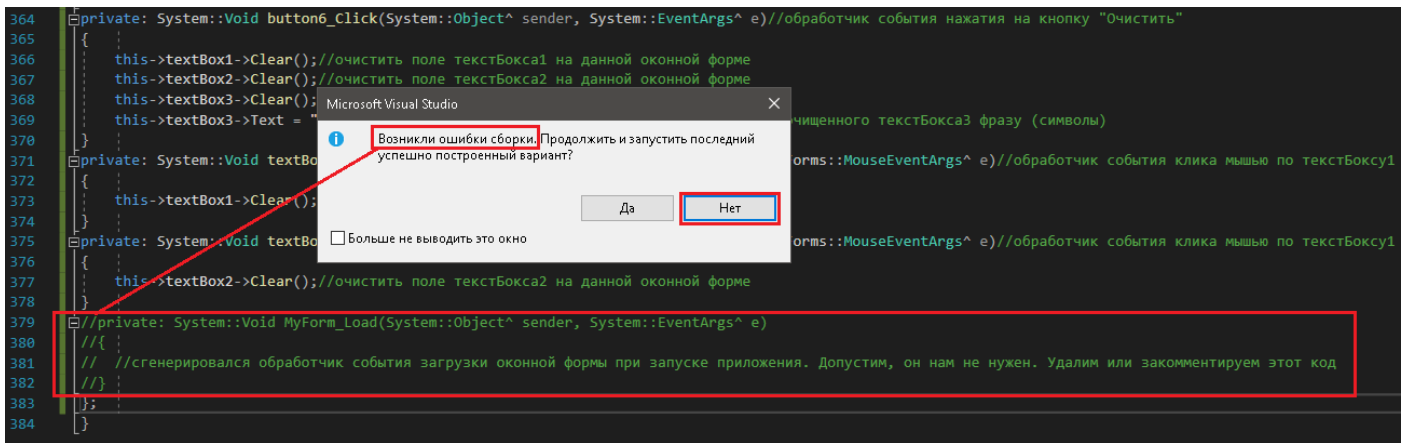


Введем вещественные числа с запятой. Результат проверим в другом приложении – стандартном Калькуляторе:

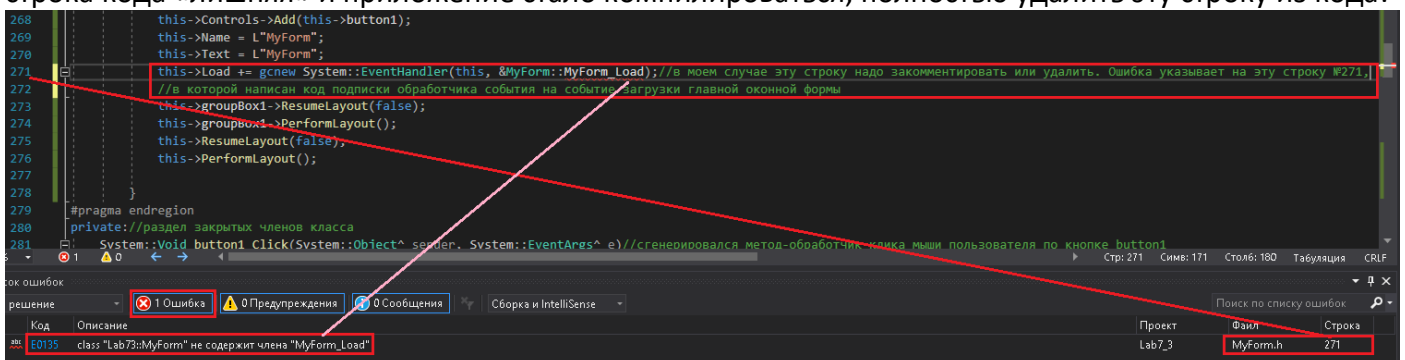


Проверьте, что уже протестированный функционал (радиобаттоны, чекбоксы, кнопки) тоже работает корректно и нет конфликта между обработчиками событий. В обработчик нажатия на кнопку «Очистить» можно дописать код, дающий заголовку окна приложения конкретную надпись, например, «Введите данные», поскольку после вывода результата вычисления при вводе новых некорректных данных у нас будет по-прежнему отображаться предыдущий результат вычислений.

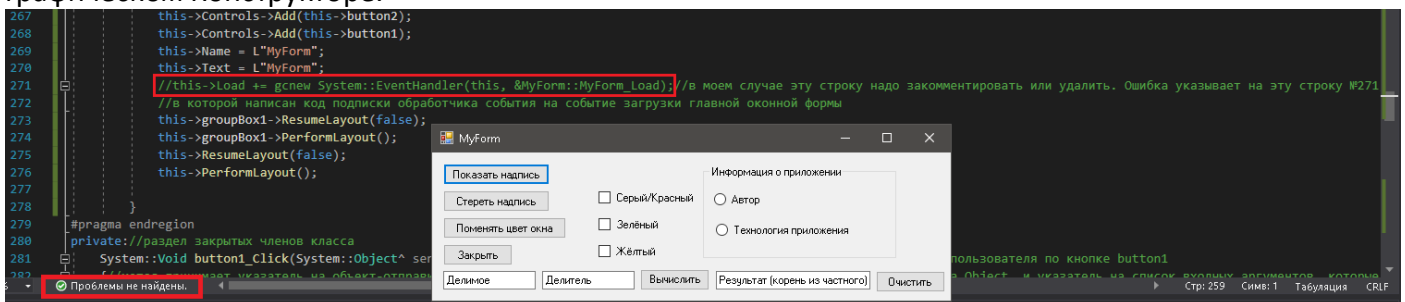
Если вы случайно кликнули по какому-либо элементу формы и автоматически сгенерировался его обработчик, то вы можете захотеть его удалить ввиду ненадобности. Но если просто удалить из файла MyForm.h код такого «лишнего» обработчика события, то появится ошибка, поскольку робот создал также в коде строку подписки обработчика события на получение сообщений о свершении данного события. Эту строку кода тоже нужно удалить и тогда снова можно скомпилировать рабочее приложение. Например, дважды кликнем по оконной форме – создается обработчик события загрузки формы (запуска оконного приложения на экране ПК), который мы выделим и удалим.



Но теперь нельзя скомпилировать приложение, поскольку в нем есть ошибка – обработчика события для компилятора уже нет, а его подписка на событие – есть. Нужно найти эту строку для данного обработчика выше в файле MyForm.h, закомментировать, скомпилировать и, если эта строка кода «лишняя» и приложение стало компилироваться, полностью удалить эту строку из кода.



Комментирование этой строки кода решило проблему. Можно удалить данные закомментированные строки кода полностью, поскольку при желании эти обработчики событий можно сгенерировать вновь. Для таких целей редактировать автосгенерированный роботом код в файле MyForm.h можно. Но создавать обработчики событий, задавать настройки проекту следует в графическом Конструкторе.



Приложение готово. Ниже приводится **листинг кода** моего файла MyForm.h (ваш может иметь отличия):

```
#pragma once//стандартная директива препроцессору для заголовочного header-файла MyForm.h
//тела методов-обработчиков событий в этом файле программисту можно и нужно, но остальной
код автосгенерирован роботом и его нежелательно или запрещено изменять (робот при изменении
оконной формы в графическом Конструкторе все равно регенерирует этот код и ваш код тут
уничтожится или будет противоречить остальному коду - будут ошибки). Но подписки на события
удалять можно
namespace Lab73//создано пространство имен нашего проекта Lab73 (символ _ автоматически
отфильтрован)
```


{//подключаются пространства имен Системы, Модели компонентов, Коллекций (для работы с массивами, списками и т.д.), пространство имен Forms для работы с оконными формами, Данными и Отрисовкой графических элементов. Из этих пространств имен нам доступны классы и их члены

```
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//открытый ссылочный класс
MyForm наследует классу Form, который находится в пространстве имен класса Forms, а он - в классе Windows, а он - в классе System. Наследование открытое. Что это значит?

```
{
    public://раздел открытых общедоступных членов класса MyForm
        MyForm(void)//конструктор без параметров по умолчанию
        {
            InitializeComponent();//метод инициализации компонента, требуемый для
работы конструктора MyForm()
            //
            //TODO: добавьте код конструктора
            //
        }
```

```
protected://раздел защищенных членов класса MyForm
    /// <summary>
    /// Освободить все используемые ресурсы.
    /// </summary>
    ~MyForm()//деструктор объекта класса MyForm
    {
        if (components)//if (components != NULL)//если компоненты еще существуют
(знаимают оперативную память)
        {
            delete components;//то удалить их из оперативной памяти
        }
    }
```

private: System::Windows::Forms::Button^ button1;//при создании графических объектов в конструкторе они обязательно создаются в коде файла Myform.h в виде защищенных указателей на соответствующие объекты. Код Button^ button1 идентичен Button* button1, но при этом еще и поддерживает очищение объекта из памяти по достижении закрывающей скобки локальной области где он создан

```
protected:
private: System::Windows::Forms::Button^ button2;
private: System::Windows::Forms::Button^ button3;
private: System::Windows::Forms::Button^ button4;
private: System::Windows::Forms::Label^ label1;
private: System::Windows::Forms::CheckBox^ checkBox1;
```

```
private: System::Windows::Forms::CheckBox^ checkBox2;
private: System::Windows::Forms::CheckBox^ checkBox3;
private: System::Windows::Forms::GroupBox^ groupBox1;
private: System::Windows::Forms::RadioButton^ radioButton2;
private: System::Windows::Forms::RadioButton^ radioButton1;
private: System::Windows::Forms::TextBox^ textBox1;
private: System::Windows::Forms::TextBox^ textBox2;
private: System::Windows::Forms::TextBox^ textBox3;
private: System::Windows::Forms::Button^ button5;
private: System::Windows::Forms::Button^ button6;
```

```
private:
```

```
    /// <summary>
```

```
    /// Обязательная переменная конструктора.
```

```
    /// </summary>
```

```
    System::ComponentModel::Container ^components;//создан (объявлен) указатель на
контейнер (вместилище) для всех компонентов оконной формы
```

```
#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)//полное определение метода InitializeComponent() с
```

телом

```
{//this - это указатель на текущий объект - оконную форму MyForm, экземпляр класса
```

MyForm

```
        this->button1 = (gcnew System::Windows::Forms::Button());
        this->button2 = (gcnew System::Windows::Forms::Button());
        this->button3 = (gcnew System::Windows::Forms::Button());
        this->button4 = (gcnew System::Windows::Forms::Button());
        this->label1 = (gcnew System::Windows::Forms::Label());
        this->checkBox1 = (gcnew System::Windows::Forms::CheckBox());
        this->checkBox2 = (gcnew System::Windows::Forms::CheckBox());
        this->checkBox3 = (gcnew System::Windows::Forms::CheckBox());
        this->groupBox1 = (gcnew System::Windows::Forms::GroupBox());
        this->radioButton2 = (gcnew System::Windows::Forms::RadioButton());
        this->radioButton1 = (gcnew System::Windows::Forms::RadioButton());
        this->textBox1 = (gcnew System::Windows::Forms::TextBox());
        this->textBox2 = (gcnew System::Windows::Forms::TextBox());
        this->textBox3 = (gcnew System::Windows::Forms::TextBox());
        this->button5 = (gcnew System::Windows::Forms::Button());
        this->button6 = (gcnew System::Windows::Forms::Button());
        this->groupBox1->SuspendLayout();
        this->SuspendLayout();
        //
        // button1 Автогенерированный конструктором код
        //
        this->button1->Location = System::Drawing::Point(12, 12);
        this->button1->Name = L"button1";
```

```

this->button1->Size = System::Drawing::Size(115, 23);
this->button1->TabIndex = 0;
this->button1->Text = L"Показать надпись";
this->button1->UseVisualStyleBackColor = true;
this->button1->Click += gcnew System::EventHandler(this, &MyForm::button1_Click);
//
// button2
//
this->button2->Location = System::Drawing::Point(12, 41);
this->button2->Name = L"button2";
this->button2->Size = System::Drawing::Size(115, 23);
this->button2->TabIndex = 1;
this->button2->Text = L"Стереть надпись";
this->button2->UseVisualStyleBackColor = true;
this->button2->Click += gcnew System::EventHandler(this, &MyForm::button2_Click);
//
// button3
//
this->button3->Location = System::Drawing::Point(12, 70);
this->button3->Name = L"button3";
this->button3->Size = System::Drawing::Size(133, 23);
this->button3->TabIndex = 2;
this->button3->Text = L"Поменять цвет окна";
this->button3->UseVisualStyleBackColor = true;
this->button3->Click += gcnew System::EventHandler(this, &MyForm::button3_Click);
//
// button4
//
this->button4->Location = System::Drawing::Point(12, 99);
this->button4->Name = L"button4";
this->button4->Size = System::Drawing::Size(75, 23);
this->button4->TabIndex = 3;
this->button4->Text = L"Закрыть";
this->button4->UseVisualStyleBackColor = true;
this->button4->Click += gcnew System::EventHandler(this, &MyForm::button4_Click);
//
// label1
//
this->label1->AutoSize = true;
this->label1->Location = System::Drawing::Point(145, 17);
this->label1->Name = L"label1";
this->label1->Size = System::Drawing::Size(0, 13);
this->label1->TabIndex = 4;
//
// checkBox1
//
this->checkBox1->AutoSize = true;
this->checkBox1->Location = System::Drawing::Point(180, 41);
this->checkBox1->Name = L"checkBox1";
this->checkBox1->Size = System::Drawing::Size(109, 17);

```

```

this->checkBox1->TabIndex = 5;
this->checkBox1->Text = L"Серый/Красный";
this->checkBox1->UseVisualStyleBackColor = true;
this->checkBox1->CheckedChanged += gcnew System::EventHandler(this,
&MyForm::checkBox1_CheckedChanged);
//
// checkBox2
//
this->checkBox2->AutoSize = true;
this->checkBox2->Location = System::Drawing::Point(180, 70);
this->checkBox2->Name = L"checkBox2";
this->checkBox2->Size = System::Drawing::Size(71, 17);
this->checkBox2->TabIndex = 6;
this->checkBox2->Text = L"Зелёный";
this->checkBox2->UseVisualStyleBackColor = true;
this->checkBox2->CheckedChanged += gcnew System::EventHandler(this,
&MyForm::checkBox2_CheckedChanged);
//
// checkBox3
//
this->checkBox3->AutoSize = true;
this->checkBox3->Location = System::Drawing::Point(180, 99);
this->checkBox3->Name = L"checkBox3";
this->checkBox3->Size = System::Drawing::Size(68, 17);
this->checkBox3->TabIndex = 7;
this->checkBox3->Text = L"Жёлтый";
this->checkBox3->UseVisualStyleBackColor = true;
this->checkBox3->CheckedChanged += gcnew System::EventHandler(this,
&MyForm::checkBox3_CheckedChanged);
//
// groupBox1
//
this->groupBox1->Controls->Add(this->radioButton2);
this->groupBox1->Controls->Add(this->radioButton1);
this->groupBox1->Location = System::Drawing::Point(293, 14);
this->groupBox1->Name = L"groupBox1";
this->groupBox1->Size = System::Drawing::Size(209, 101);
this->groupBox1->TabIndex = 8;
this->groupBox1->TabStop = false;
this->groupBox1->Text = L"Информация о приложении";
//
// radioButton2
//
this->radioButton2->AutoSize = true;
this->radioButton2->Location = System::Drawing::Point(14, 61);
this->radioButton2->Name = L"radioButton2";
this->radioButton2->Size = System::Drawing::Size(149, 17);
this->radioButton2->TabIndex = 1;
this->radioButton2->TabStop = true;
this->radioButton2->Text = L"Технология приложения";

```

```

        this->radioButton2->UseVisualStyleBackColor = true;
        this->radioButton2->CheckedChanged += gcnew System::EventHandler(this,
&MyForm::radioButton2_CheckedChanged);
        //
        // radioButton1
        //
        this->radioButton1->AutoSize = true;
        this->radioButton1->Location = System::Drawing::Point(12, 28);
        this->radioButton1->Name = L"radioButton1";
        this->radioButton1->Size = System::Drawing::Size(55, 17);
        this->radioButton1->TabIndex = 0;
        this->radioButton1->TabStop = true;
        this->radioButton1->Text = L"Автор";
        this->radioButton1->UseVisualStyleBackColor = true;
        this->radioButton1->CheckedChanged += gcnew System::EventHandler(this,
&MyForm::radioButton1_CheckedChanged);
        //
        // textBox1
        //
        this->textBox1->Location = System::Drawing::Point(12, 128);
        this->textBox1->Name = L"textBox1";
        this->textBox1->Size = System::Drawing::Size(100, 20);
        this->textBox1->TabIndex = 9;
        this->textBox1->Text = L"Делимое";
        this->textBox1->MouseClick += gcnew
System::Windows::Forms::EventHandler(this, &MyForm::textBox1_MouseClick);
        //
        // textBox2
        //
        this->textBox2->Location = System::Drawing::Point(119, 128);
        this->textBox2->Name = L"textBox2";
        this->textBox2->Size = System::Drawing::Size(100, 20);
        this->textBox2->TabIndex = 10;
        this->textBox2->Text = L"Делитель";
        this->textBox2->MouseClick += gcnew
System::Windows::Forms::EventHandler(this, &MyForm::textBox2_MouseClick);
        //
        // textBox3
        //
        this->textBox3->Location = System::Drawing::Point(307, 128);
        this->textBox3->Name = L"textBox3";
        this->textBox3->Size = System::Drawing::Size(167, 20);
        this->textBox3->TabIndex = 11;
        this->textBox3->Text = L"Результат (корень из частного)";
        //
        // button5
        //
        this->button5->Location = System::Drawing::Point(225, 126);
        this->button5->Name = L"button5";
        this->button5->Size = System::Drawing::Size(75, 23);

```

```

this->button5->TabIndex = 12;
this->button5->Text = L"Вычислить";
this->button5->UseCompatibleTextRendering = true;
this->button5->UseVisualStyleBackColor = true;
this->button5->Click += gcnew System::EventHandler(this, &MyForm::button5_Click);
//
// button6
//
this->button6->Location = System::Drawing::Point(480, 128);
this->button6->Name = L"button6";
this->button6->Size = System::Drawing::Size(75, 23);
this->button6->TabIndex = 13;
this->button6->Text = L"Очистить";
this->button6->UseVisualStyleBackColor = true;
this->button6->Click += gcnew System::EventHandler(this, &MyForm::button6_Click);
//
// MyForm
//
this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
this->ClientSize = System::Drawing::Size(561, 160);
this->Controls->Add(this->button6);
this->Controls->Add(this->button5);
this->Controls->Add(this->textBox3);
this->Controls->Add(this->textBox2);
this->Controls->Add(this->textBox1);
this->Controls->Add(this->groupBox1);
this->Controls->Add(this->checkBox3);
this->Controls->Add(this->checkBox2);
this->Controls->Add(this->checkBox1);
this->Controls->Add(this->label1);
this->Controls->Add(this->button4);
this->Controls->Add(this->button3);
this->Controls->Add(this->button2);
this->Controls->Add(this->button1);
this->Name = L"MyForm";
this->Text = L"MyForm";
this->groupBox1->ResumeLayout(false);
this->groupBox1->PerformLayout();
this->ResumeLayout(false);
this->PerformLayout();

```

```

}

```

```

#pragma endregion

```

```

private://раздел закрытых членов класса

```

```

    System::Void button1_Click(System::Object^ sender, System::EventArgs^ e)//сгенерировался
    метод-обработчик клика мыши пользователя по кнопке button1

```

```

    {//метод принимает указатель на объект-отправитель события (sender), это объект самого
    базового (родительского) класса Object, и указатель на список входных аргументов, которые объект-
    отправитель посылает объекту-получателю сообщения о свершении события
    }

```

```

        //тело метода-обработчика пустое - его заполняет программист. Метод ничего не
возвращает
        label1->Text = "Кнопка работает.";
    }
private: System::Void button2_Click(System::Object^ sender, System::EventArgs^ e)
{
    //пустое тело метода-обработчика события клика по кнопке button2
    label1->Text = "";
}
private: System::Void button3_Click(System::Object^ sender, System::EventArgs^ e)
{
    if (this->BackColor == SystemColors::ButtonFace)//если цвет фона ЭТОГО окна "кнопочный
серый" (это базовый стандартный цвет фона по умолчанию)
    {
        this->BackColor = Color::Red;//то сделать фоновым цветом ЭТОЙ оконной формы
красный цвет, заданный в статическом классе Color
    }
    else//иначе
    {
        this->BackColor = SystemColors::ButtonFace;//назначить фоновым цветом ЭТОЙ оконной
формы ситемный цвет "кнопочный серый"
    }
}
private: System::Void button4_Click(System::Object^ sender, System::EventArgs^ e)
{
    this->Close();//метод, который закрывает ЭТУ оконную форму (текущее основное окно
приложения)
}
private: System::Void checkBox1_CheckedChanged(System::Object^ sender, System::EventArgs^
e)//обработчик нажатия на чекбокс1
{//код такой проверки сразу работает корректно
    if (this->BackColor == Color::Red)
    {
        this->BackColor = SystemColors::ButtonFace;
    }
    else
    {
        this->BackColor = Color::Red;
    }
    this->checkBox2->CheckState = CheckState::Unchecked;//всем остальным чекбоксам
присваиваем статус нажатия "НЕ нажато", чтобы предупредить
    this->checkBox3->CheckState = CheckState::Unchecked;//случаи, если другие чекбоксы были
ранее нажаты и отображают неактуальные "галочки"
}
private: System::Void checkBox2_CheckedChanged(System::Object^ sender, System::EventArgs^
e)//обработчик нажатия на чекбокс2
{
    this->BackColor = Color::Green;//присвоить фону формы зеленый цвет из значений класса
System::Drawing::Color. Сразу пишу Color, поскольку

```

```

        this->checkBox1->CheckState = CheckState::Unchecked;//в строке кода № 10 уже подключено
        пространство имен using namespace System::Drawing;
        this->checkBox3->CheckState = CheckState::Unchecked;
    }
    private: System::Void checkBox3_CheckedChanged(System::Object^ sender, System::EventArgs^
    e)//обработчик нажатия на чекбокс3
    {
        this->BackColor = Color::Yellow;//присвоить фону формы желтый цвет из значений класса
        System::Drawing::Color.
        this->checkBox1->CheckState = CheckState::Unchecked;//всем остальным чекбоксам
        присваиваем статус нажатия "НЕ нажато", чтобы предупредить
        this->checkBox2->CheckState = CheckState::Unchecked;//случай, если другие чекбоксы были
        ранее нажаты и отображают неактуальные "галочки"
    }
    private: System::Void radioButton1_CheckedChanged(System::Object^ sender, System::EventArgs^
    e)//обработчик события клика мышки по radioButton1
    {
        this->Text = "Приложение Иванова И.И. (с)2020";//присвоить полю Текст ЭТОЙ оконной формы
        строку символов "Приложение Иванова И.И. (с)2020"
        //для текущей главной оконной формы есть только одного окно, и указатель на него - this
        private: System::Void radioButton2_CheckedChanged(System::Object^ sender, System::EventArgs^
        e)//обработчик события клика мышки по radioButton2
        {
            this->Text = "Это Windows Forms приложение";//присвоить полю Текст ЭТОЙ оконной формы
            строку символов "Это Windows Forms приложение"
        }
        private: System::Void button5_Click(System::Object^ sender, System::EventArgs^ e)//обработчик
        события нажатия на кнопку "Вычислить"
        {
            try//это блок try{}-catch({}), отлавливающий в коде блока try{...} ошибки типов, указанные в
            блоке catch()
            {
                double a = Convert::ToDouble(this->textBox1->Text);//у статического класса
                Convert(т.е.Конвертация, перевод данных из одного типа данных в другой) вызываем метод
                ToDouble(),
                //то есть привести входное значение, которое берется из тексБокса, к вещественному
                типу. После конвертации вещественный результат справа помещаем в вещественную переменную,
                double b = Convert::ToDouble(this->textBox2->Text);//созданную слева
                if (b == 0 || a / b < 0)//если делитель равен нулю или хотя бы результат деления
                меньше нуля (значит, из него НЕЛЬЗЯ извлечь квадратный корень)
                //то выводим мини-окно Сообщения с заголовком "Ошибка!" и текстом "Попытка
                деления на ноль или извлечения корня из отрицательного числа."
                MessageBox::Show("Попытка деления на ноль или извлечения корня из
                отрицательного числа.", "Ошибка!");
                button6_Click(sender, System::EventArgs::Empty);//после закрытия окна
                Сообщения, очищаем поля от старых ошибочных данных, вызвав обработчик кнопки "Очистить"
                //обработчик события должен принимать имя отправителя, создающего событие, и
                передаваемые данные (тут передаем ничего - Empty)
                else//если проверку на математическую корректность данные прошли успешно

```



```

        { //то можно разделить делимое на делитель и извлечь из этого результата квадратный
корень, который берется из статического класса Math, помещая итоговый результат в
        double c = Math.Sqrt(a / b); //вещественную переменную double c
        this->textBox3->Clear(); //очищаем поле текстБокса3 от предыдущего любого
текста
        this->textBox3->Text = c.ToString(); //вещественное число приводим к строковому
виду (символьный массив) и помещаем в поле текстБокса3
        this->Text = c.ToString(); //вещественное число приводим к строковому виду
(символьный массив) и помещаем в качестве заголовка основного окна
    }
    //конец блока try - за ним сразу должен следовать блок кода catch(){...}, отлавливающий
ошибки конкретного типа
    catch (System::FormatException^ e) //отлавливать ошибки типа System::FormatException, то есть
ошибки, связанные с неподходящим форматом (типом) данных
    { //если возникла ошибка неподходящего форматирования данных (данные не соответствуют
ожидаемому от них типу данных), то
        MessageBox::Show(e->Message->ToString(), "Ошибка формата данных!"); //вывести
мини-окно Сообщения с заголовком "Ошибка формата данных!" и текстом сути ошибки
форматирования
        button6_Click(sender, System::EventArgs::Empty); //после закрытия окна Сообщения
очистить поля от старых ошибочных данных, вызвав обработчик кнопки "Очистить"
    } //конец тела блока catch(){...}
} //конец тела обработчика события клика по кнопке "Вычислить"
private: System::Void button6_Click(System::Object^ sender, System::EventArgs^ e) //обработчик
события нажатия на кнопку "Очистить"
{
    this->textBox1->Clear(); //очистить поле текстБокса1 на данной оконной форме
    this->textBox2->Clear(); //очистить поле текстБокса2 на данной оконной форме
    this->textBox3->Clear(); //очистить поле текстБокса3 на данной оконной форме
    this->textBox3->Text = "Результат (корень из частного)"; //поместить в поле уже очищенного
текстБокса3 фразу (символы)
}
private: System::Void textBox1_MouseClick(System::Object^ sender,
System::Windows::Forms::MouseEventArgs^ e) //обработчик события клика мышью по текстБоксу1
{
    this->textBox1->Clear(); //очистить поле текстБокса1 на данной оконной форме
}
private: System::Void textBox2_MouseClick(System::Object^ sender,
System::Windows::Forms::MouseEventArgs^ e) //обработчик события клика мышью по текстБоксу1
{
    this->textBox2->Clear(); //очистить поле текстБокса2 на данной оконной форме
}
};
}

```

Отыщите скомпилированный .exe-файл вашей программы и запустите его на рабочем столе без MS VS2019. Попробуйте запустить его на других компьютерах с ОС MS Windows 10. **Кроме кода, имейте при себе исполнимые .exe-файлы ваших оконных программ на защите лабораторных работ.**

3.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 4

В каждом варианте предусмотреть в программе кнопку, которая выводит ваши имя, фамилию и номер группы в области рабочего окна вашего приложения. Значения для расчетов вводится пользователем, например, через элементы TextBox'ы. Если это не противоречит сути индивидуального задания, программа должна уметь обрабатывать вещественные значения. Отлавливайте некорректные данные пользователя, предусмотрите графический интерфейс, дружелюбный и понятный пользователю. Рассчитанный выводимый программой результат должен иметь указание единиц измерения.

№	Задача
1	Найти массу x литров молока, если известно, что плотность молока ρ кг/м ³ . <i>Пример: $x = 7$ л, $\rho = 1030$ кг/м³. Ответ: 7,21 кг.</i>
2	Объем цилиндра равен V , а площадь основания – S . Какова высота цилиндра H ? <i>Пример: $V = 10$ м³, $S = 5$ м². Ответ: 2 м.</i>
3	Дана длина ребра куба a . Найти объем куба V и площадь его боковой поверхности S . <i>Пример: $a = 5$ см. Ответ: $V = 125$ см³, $S = 100$ см².</i>
4	Каков объем кислорода, содержащегося в комнате размером $a * b * c$, если кислород составляет 21% объема воздуха? <i>Пример: $a = 3$ м, $b = 4$ м, $c = 5$ м. Ответ: 12,6 м³.</i>
5	Найти площадь равнобокой трапеции с основаниями a и b и углом при большем основании равным x . <i>Пример: $a = 6$ см, $b = 5$ см, $x = 45^\circ$. Ответ: 2,75 см².</i>
6	Найти угол между отрезком прямой, соединяющей начало координат с точкой $A(x, y)$, и осью Ox (точка лежит в 1-й четверти декартового пространства). <i>Пример: $x = 3$ см, $y = 4$ см. Ответ: 53,13°.</i>
7	Определить время падения камня на поверхность земли с высоты h . <i>Пример: $h = 10$ м. Ответ: 1,4278 с.</i>
8	Три сопротивления R_1, R_2, R_3 соединены параллельно. Найти сопротивление соединения. <i>Пример: $R_1 = 10$ Ом, $R_2 = 15$ Ом, $R_3 = 20$ Ом. Ответ: 4,62 Ом.</i>
9	Написать программу вычисления площади параллелограмма. Извне вводятся стороны a, b и угол между ними x . <i>Пример: $a = 10$ мм, $b = 15$ мм, $x = 30^\circ$. Ответ: 75 мм.</i>
10	Написать программу вычисления объема прямоугольного параллелепипеда. Извне вводятся длина a , ширина b и высота c . <i>Пример: $a = 10$ см, $b = 15$ см, $c = 20$ см. Ответ: 3000 см.</i>
11	Написать программу вычисления площади поверхности прямоугольного параллелепипеда. Извне вводятся длина a , ширина b и высота c . <i>Пример: $a = 10$ м, $b = 15$ м, $c = 20$ м. Ответ: 1300 м².</i>
12	Написать программу вычисления объема цилиндра. Извне вводятся радиус основания R и высота цилиндра h . <i>Пример: $R = 10$ см, $h = 15$ см. Ответ: 4712,39 см³.</i>

13	Написать программу вычисления стоимости покупки, состоящей из нескольких тетрадей и карандашей. Извне вводится цена одной тетради Ct и количество тетрадей Kt , а также цена карандаша Ck и количество карандашей Kk . <i>Пример: $Ct = 1$ рубль, $Kt = 15$ штук, $Ck = 0.2$ рубля, $Kk = 5$ штук. Ответ: 16 рублей.</i>
14	Написать программу вычисления стоимости покупки, состоящей из нескольких тетрадей и такого же количества обложек к ним. Извне вводится цена одной тетради Ct , одной обложки Cb и количество тетрадей Kt . <i>Пример: $Ct = 1.2$ рублей, $Kt = 15$ штук, $Cb = 0,2$ рубля. Ответ: 21 рубль.</i>
15	Написать программу вычисления стоимости некоторого количества (по весу) яблок. Извне вводится цена одного килограмма яблок C и вес яблок V . <i>Пример: $C = 25$ рублей, $V = 1,5$ кг. Ответ: 37,5 рублей.</i>
16	Написать программу вычисления сопротивления электрической цепи, состоящей из двух параллельно соединенных сопротивлений. Извне вводится величина первого и второго сопротивления. <i>Пример: $R_1 = 10$ Ом, $R_2 = 15$ Ом. Ответ: 6 Ом.</i>
17	Написать программу вычисления сопротивления электрической цепи, состоящей из двух последовательно соединенных сопротивлений. Извне вводится величина первого и второго сопротивления. <i>Пример: $R_1 = 10$ Ом, $R_2 = 15$ Ом. Ответ: 25 Ом.</i>
18	Написать программу вычисления силы тока в электрической цепи. Извне вводятся напряжение U и сопротивление R . <i>Пример: $U = 10$ Вт, $R = 15$ Ом. Ответ: 0,6667 А.</i>
19	Составить программу, которая поменяет местами значения введенных вещественных переменных x и y не используя дополнительной переменной. <i>Пример: $x = 5$, $y = -2$. Ответ: $x = -2$, $y = 5$.</i>
20	Составить программу, которая поменяет местами значения введенных целочисленных переменных x , y , z так, чтобы в переменной x оказалось значение переменной y , в y – значение переменной z , а в z – прежнее значение переменной x не используя дополнительной переменной. <i>Пример: $x = 5$, $y = -2$, $z = 9$. Ответ: $x = -2$, $y = 9$, $z = 5$.</i>
21	Составить программу перевода километров в мили (1 миля = 1,609 км). <i>Пример: 8,045 км. Ответ: 5 миль.</i>
22	Составить программу перевода метров в версты (1 верста = 1066,8 м). <i>Пример: 3200,4 м. Ответ: 3 версты.</i>
23	Составить программу перевода сантиметров в вершки (1 вершок = 4,445 см). <i>Пример: 26,67 см. Ответ: 6 вершков.</i>
24	Составить программу перевода грамм в золотники (1 золотник = 4,26575 грамм). <i>Пример: 17,063 грамма. Ответ: 4 золотника.</i>
25	Составить программу перевода литров в бочки (1 бочка = 491,96 литров). <i>Пример: 4919,6 литров. Ответ: 10 бочек.</i>
26	Составить программу, которая поменяет местами значения введенных вещественных переменных x и y без использования дополнительной переменной. <i>Пример: $x = 3.8$, $y = -9.01$. Ответ: $x = -9.01$, $y = 3.8$.</i>
27	Составить программу, которая поменяет местами значения введенных целочисленных переменных x , y , z так, чтобы в переменной x оказалось значение переменной y , в y – значение переменной z , а в z – прежнее значение переменной x без использования дополнительной переменной. <i>Пример: $x = 3$, $y = -2$, $z = 1$. Ответ: $x = -2$, $y = 1$, $z = 3$.</i>

28	Найти угол между отрезком прямой, соединяющей начало координат с точкой $A(x, y)$, и осью OX (точка A лежит в 1-й (верхней правой) четверти декартового пространства координат). <i>Пример: $x = 3$ см, $y = 4$ см. Ответ: $53,13^\circ$.</i>
----	--

Задание 5

Задания по варианту (каждому учащемуся сделать новый проект (одну программу), реализующий все три пункта задания по вашему варианту по журналу группы):

1	1	Написать оконную программу, принимающую два вещественных числа, извлекающую из каждого из них квадратный корень, складывающую полученный промежуточный результат и делящую его на первое число. Полученный результат выводить в отдельном поле. Обработать ввод пользователем некорректных данных.
	2	Сделать в программе два радиобаттона, устанавливающие окошку вашего приложения Yellow или Tomato цвет.
	3	Сделать в вашей программе два чекбокса: 1) по нажатию на первый в поле заголовка окна отображается фраза с вашей фамилией, именем и отчеством, например «Иванова Алеся Андреевна»; 2) при нажатии второго чекбокса в окне приложения отображается фраза с номером вашей группы и вашим номером по журналу группы, например «Т-834 № 9».
2	1	Написать оконную программу, принимающую вещественные числа x и c , вычисляющую с ними выражение $\sqrt[3]{x} / (\sqrt[3]{c} - x)$, и выводящую полученный результат в отдельном поле. Обработать ввод пользователем некорректных данных, деления на ноль, извлечения корня из отрицательного числа.
	2	Сделать в программе два радиобаттона, устанавливающие окошку вашего приложения White или OrangeRed цвет.
	3	Сделать в вашей программе два чекбокса: 1) по нажатию на первый в поле заголовка окна отображается фраза с вашей фамилией, именем и отчеством, например «Иванова Алеся Андреевна»; 2) при нажатии второго чекбокса в окне приложения отображается фраза с номером вашей группы и вашим номером по журналу группы, например «Т-834 № 9».
3	1	Написать оконную программу, принимающую два вещественных числа, перемножающую их, извлекающую квадратный корень из промежуточного результата, делящую его на куб второго числа и выводящую полученный результат в отдельном поле. Обработать ввод пользователем некорректных данных.
	2	Сделать в программе два радиобаттона, устанавливающие окошку вашего приложения Violet или Black цвет.
	3	Сделать в вашей программе два чекбокса: 1) по нажатию на первый в поле заголовка окна отображается фраза с вашей фамилией, именем и отчеством, например «Иванова Алеся Андреевна»; 2) при нажатии второго чекбокса в окне приложения отображается фраза с номером вашей группы и вашим номером по журналу группы, например «Т-834 № 9».
4	1	Написать оконную программу, принимающую два вещественных числа, делящую первое число на второе, возводящую промежуточный результат в куб, отнимающую от него квадратный корень из первого числа и выводящую полученный результат в отдельном поле. Обработать ввод пользователем некорректных данных и ошибку деления на ноль.
	2	Сделать в программе два радиобаттона, устанавливающие окошку вашего приложения SteelBlue или Brown цвет.
	3	Сделать в вашей программе два чекбокса: 1) по нажатию на первый в поле заголовка окна отображается фраза с вашей фамилией, именем и отчеством, например «Иванова Алеся Андреевна»; 2) при нажатии второго чекбокса в окне приложения отображается фраза с номером вашей группы и вашим номером по журналу группы, например «Т-834 № 9».
5	1	Написать оконную программу, принимающую три вещественных числа, перемножающую их, извлекающую из этого произведения квадратный корень, делящую его на кубический корень из второго числа и выводящую полученный результат в отдельном поле и в заголовке рамки

		окна приложения. Обработать ввод пользователем некорректных данных.
	2	Сделать в программе два радиобаттона, устанавливающие окошку вашего приложения Snow или Orange цвет.
	3	Сделать в вашей программе два чекбокса: 1) по нажатию на первый в поле заголовка окна отображается фраза с вашей фамилией, именем и отчеством, например «Иванова Алеся Андреевна»; 2) при нажатии второго чекбокса в окне приложения отображается фраза с номером вашей группы и вашим номером по журналу группы, например «Т-834 № 9».
6	1	Написать оконную программу, принимающую два вещественных числа, извлекающую из каждого из них квадратный корень, сумму этих корней делящую на произведение этих чисел и выводящую полученный результат в отдельном поле и в заголовке рамки окна приложения. Обработать ввод пользователем некорректных данных и извлечения корня из отрицательного числа.
	2	Сделать в программе два радиобаттона, устанавливающие окошку вашего приложения Silver или Chocolate цвет.
	3	Сделать в вашей программе два чекбокса: 1) по нажатию на первый в поле заголовка окна отображается фраза с вашей фамилией, именем и отчеством, например «Иванова Алеся Андреевна»; 2) при нажатии второго чекбокса в окне приложения отображается фраза с номером вашей группы и вашим номером по журналу группы, например «Т-834 № 9».
7	1	Написать оконную программу, принимающую три вещественных числа, извлекающую из их произведения кубический корень, делящую его на квадратный корень первого числа и выводящую полученный результат в отдельном поле и в заголовке рамки окна приложения. Обработать ввод пользователем некорректных данных.
	2	Сделать в программе два радиобаттона, устанавливающие окошку вашего приложения Sienna или Coral цвет.
	3	Сделать в вашей программе два чекбокса: 1) по нажатию на первый в поле заголовка окна отображается фраза с вашей фамилией, именем и отчеством, например «Иванова Алеся Андреевна»; 2) при нажатии второго чекбокса в окне приложения отображается фраза с номером вашей группы и вашим номером по журналу группы, например «Т-834 № 9».
8	1	Написать оконную программу, принимающую вещественные числа x и c , вычисляющую с ними выражение $\frac{\sqrt{x^2 + \sqrt{x}}}{c}$, и выводящую полученный результат в отдельном поле. Обработать ввод пользователем некорректных данных.
	2	Сделать в программе два радиобаттона, устанавливающие окошку вашего приложения SeaGreen или Pink цвет.
	3	Сделать в вашей программе два чекбокса: 1) по нажатию на первый в поле заголовка окна отображается фраза с вашей фамилией, именем и отчеством, например «Иванова Алеся Андреевна»; 2) при нажатии второго чекбокса в окне приложения отображается фраза с номером вашей группы и вашим номером по журналу группы, например «Т-834 № 9».
9	1	Написать оконную программу, принимающую вещественные числа x и c , вычисляющую с ними выражение $\sqrt{\frac{c^2}{x^3 - c}}$, и выводящую полученный результат в отдельном поле. Обработать ввод пользователем некорректных данных.
	2	Сделать в программе два радиобаттона, устанавливающие окошку вашего приложения SandyBrown или Olive цвет.
	3	Сделать в вашей программе два чекбокса: 1) по нажатию на первый в поле заголовка окна отображается фраза с вашей фамилией, именем и отчеством, например «Иванова Алеся Андреевна»; 2) при нажатии второго чекбокса в окне приложения отображается фраза с номером вашей группы и вашим номером по журналу группы, например «Т-834 № 9».
10	1	Написать оконную программу, принимающую вещественные числа x и c , вычисляющую с

		ними выражение $\frac{x^2}{\sqrt{c-x}}$, и выводящую полученный результат в отдельном поле. Обработать ввод пользователем некорректных данных.
	2	Сделать в программе два радиобаттона, устанавливающие окошку вашего приложения Salmon или RosyBrown цвет.
	3	Сделать в вашей программе два чекбокса: 1) по нажатию на первый в поле заголовка окна отображается фраза с вашей фамилией, именем и отчеством, например «Иванова Алеся Андреевна»; 2) при нажатии второго чекбокса в окне приложения отображается фраза с номером вашей группы и вашим номером по журналу группы, например «Т-834 № 9».
11	1	Написать оконную программу, принимающую вещественные числа x и c , вычисляющую с ними выражение $(x^3 - 1,5 * c^2) / \text{Корень}(c)$, и выводящую полученный результат в отдельном поле. Обработать ввод пользователем некорректных данных и деления на ноль.
	2	Сделать в программе два радиобаттона, устанавливающие окошку вашего приложения Red или Aquamarine цвет.
	3	Сделать в вашей программе два чекбокса: 1) по нажатию на первый в поле заголовка окна отображается фраза с вашей фамилией, именем и отчеством, например «Иванова Алеся Андреевна»; 2) при нажатии второго чекбокса в окне приложения отображается фраза с номером вашей группы и вашим номером по журналу группы, например «Т-834 № 9».
12	1	Написать оконную программу, принимающую вещественные числа x и c , вычисляющую с ними выражение $(\text{Корень}(x) - c) / (3 * x + c)$, и выводящую полученный результат в отдельном поле. Обработать ввод пользователем некорректных данных, деления на ноль и извлечения квадратного корня из отрицательного числа.
	2	Сделать в программе два радиобаттона, устанавливающие окошку вашего приложения Purple или Beige цвет.
	3	Сделать в вашей программе два чекбокса: 1) по нажатию на первый в поле заголовка окна отображается фраза с вашей фамилией, именем и отчеством, например «Иванова Алеся Андреевна»; 2) при нажатии второго чекбокса в окне приложения отображается фраза с номером вашей группы и вашим номером по журналу группы, например «Т-834 № 9».
13	1	Написать оконную программу, принимающую вещественные числа x и c , вычисляющую с ними выражение $(2 * x + 1,9 * c^2) / \text{Корень_кубический}(x)$, и выводящую полученный результат в отдельном поле. Обработать ввод пользователем некорректных данных, деления на ноль.
	2	Сделать в программе два радиобаттона, устанавливающие окошку вашего приложения Plum или Bisque цвет.
	3	Сделать в вашей программе два чекбокса: 1) по нажатию на первый в поле заголовка окна отображается фраза с вашей фамилией, именем и отчеством, например «Иванова Алеся Андреевна»; 2) при нажатии второго чекбокса в окне приложения отображается фраза с номером вашей группы и вашим номером по журналу группы, например «Т-834 № 9».
14	1	Написать оконную программу, принимающую вещественные числа x и c , вычисляющую с ними выражение $(x^3 - 3,2 * c) / (\text{корень}(x) + \text{корень}(c))$, и выводящую полученный результат в отдельном поле. Обработать ввод пользователем некорректных данных, деления на ноль, извлечения корня из отрицательного числа.
	2	Сделать в программе два радиобаттона, устанавливающие окошку вашего приложения Peru или Blue цвет.
	3	Сделать в вашей программе два чекбокса: 1) по нажатию на первый в поле заголовка окна отображается фраза с вашей фамилией, именем и отчеством, например «Иванова Алеся Андреевна»; 2) при нажатии второго чекбокса в окне приложения отображается фраза с номером вашей группы и вашим номером по журналу группы, например «Т-834 № 9».
15	1	Написать оконную программу, принимающую два вещественных числа, вычитающую из первого числа второе, извлекающую из этого корень квадратный, делящую его на куб

		второго числа и выводящую полученный результат в отдельном поле. Обработать ввод пользователем некорректных данных.
	2	Сделать в программе два радиобаттона, устанавливающие окошку вашего приложения ParayaWhip или Azure цвет.
	3	Сделать в вашей программе два чекбокса: 1) по нажатию на первый в поле заголовка окна отображается фраза с вашей фамилией, именем и отчеством, например «Иванова Алеся Андреевна»; 2) при нажатии второго чекбокса в окне приложения отображается фраза с номером вашей группы и вашим номером по журналу группы, например «Т-834 № 9».
16	1	Написать оконную программу, принимающую два вещественных числа, складывающую их квадратные корни, делящую полученную сумму на разность этих двух чисел, и выводящую полученный результат в отдельном поле. Обработать ввод пользователем некорректных данных.
	2	Сделать в программе два радиобаттона, устанавливающие окошку вашего приложения PaleGreen или Orchid цвет.
	3	Сделать в вашей программе два чекбокса: 1) по нажатию на первый в поле заголовка окна отображается фраза с вашей фамилией, именем и отчеством, например «Иванова Алеся Андреевна»; 2) при нажатии второго чекбокса в окне приложения отображается фраза с номером вашей группы и вашим номером по журналу группы, например «Т-834 № 9».
17	1	Написать оконную программу, принимающую вещественные числа x и c , вычисляющую с ними выражение $\sqrt[3]{x - c^2 + 3} / (\sqrt[3]{c} - x)$, и выводящую полученный результат в отдельном поле. Обработать ввод пользователем некорректных данных, деления на ноль, извлечения корня из отрицательного числа.
	2	Сделать в программе два радиобаттона, устанавливающие окошку вашего приложения Aqua или Green цвет.
	3	Сделать в вашей программе два чекбокса: 1) по нажатию на первый в поле заголовка окна отображается фраза с вашей фамилией, именем и отчеством, например «Иванова Алеся Андреевна»; 2) при нажатии второго чекбокса в окне приложения отображается фраза с номером вашей группы и вашим номером по журналу группы, например «Т-834 № 9».
18	1	Написать оконную программу, принимающую два вещественных числа, перемножающую их, делящую это произведение на корень суммы их кубов и выводящую полученный результат в отдельном поле. Обработать ввод пользователем некорректных данных.
	2	Сделать в программе два радиобаттона, устанавливающие окошку вашего приложения Yellow или Tomato цвет.
	3	Сделать в вашей программе два чекбокса: 1) по нажатию на первый в поле заголовка окна отображается фраза с вашей фамилией, именем и отчеством, например «Иванова Алеся Андреевна»; 2) при нажатии второго чекбокса в окне приложения отображается фраза с номером вашей группы и вашим номером по журналу группы, например «Т-834 № 9».
19	1	Написать оконную программу, принимающую два вещественных числа, делящую первое число на корень квадратный из второго числа, увеличивающую этот промежуточный результат на куб первого числа и выводящую полученный результат в отдельном поле. Обработать ввод пользователем некорректных данных и ошибку деления на ноль.
	2	Сделать в программе два радиобаттона, устанавливающие окошку вашего приложения White или OrangeRed цвет.
	3	Сделать в вашей программе два чекбокса: 1) по нажатию на первый в поле заголовка окна отображается фраза с вашей фамилией, именем и отчеством, например «Иванова Алеся Андреевна»; 2) при нажатии второго чекбокса в окне приложения отображается фраза с номером вашей группы и вашим номером по журналу группы, например «Т-834 № 9».
20	1	Написать оконную программу, принимающую вещественное число, возводящую его в куб, складывающую его с корнем из дроби 13 -и, деленной на введенное число, и выводящую полученный результат в отдельном поле и в заголовке рамки окна приложения. Обработать

		ввод пользователем некорректных данных.
	2	Сделать в программе два радиобаттона, устанавливающие окошку вашего приложения Violet или Black цвет.
	3	Сделать в вашей программе два чекбокса: 1) по нажатию на первый в поле заголовка окна отображается фраза с вашей фамилией, именем и отчеством, например «Иванова Алеся Андреевна»; 2) при нажатии второго чекбокса в окне приложения отображается фраза с номером вашей группы и вашим номером по журналу группы, например «Т-834 № 9».
21	1	Написать оконную программу, принимающую вещественное число, извлекающую из него квадратный корень, складывающую его с кубическим корнем из частного 7-и и этого числа, и выводящую полученный результат в отдельном поле и в заголовке рамки окна приложения. Обработать ввод пользователем некорректных данных и извлечения корня из отрицательного числа.
	2	Сделать в программе два радиобаттона, устанавливающие окошку вашего приложения SteelBlue или Brown цвет.
	3	Сделать в вашей программе два чекбокса: 1) по нажатию на первый в поле заголовка окна отображается фраза с вашей фамилией, именем и отчеством, например «Иванова Алеся Андреевна»; 2) при нажатии второго чекбокса в окне приложения отображается фраза с номером вашей группы и вашим номером по журналу группы, например «Т-834 № 9».
22	1	Написать оконную программу, принимающую вещественное число, извлекающую из него кубический корень, делящую его на сумму квадрата и куба данного числа, и выводящую полученный результат в отдельном поле и в заголовке рамки окна приложения. Обработать ввод пользователем некорректных данных.
	2	Сделать в программе два радиобаттона, устанавливающие окошку вашего приложения Snow или Orange цвет.
	3	Сделать в вашей программе два чекбокса: 1) по нажатию на первый в поле заголовка окна отображается фраза с вашей фамилией, именем и отчеством, например «Иванова Алеся Андреевна»; 2) при нажатии второго чекбокса в окне приложения отображается фраза с номером вашей группы и вашим номером по журналу группы, например «Т-834 № 9».
23	1	Написать оконную программу, принимающую вещественные числа x и c , вычисляющую с ними выражение $\text{Корень}(x^2 + 2*x + c) / (c - x)$, и выводящую полученный результат в отдельном поле. Обработать ввод пользователем некорректных данных.
	2	Сделать в программе два радиобаттона, устанавливающие окошку вашего приложения Silver или Chocolate цвет.
	3	Сделать в вашей программе два чекбокса: 1) по нажатию на первый в поле заголовка окна отображается фраза с вашей фамилией, именем и отчеством, например «Иванова Алеся Андреевна»; 2) при нажатии второго чекбокса в окне приложения отображается фраза с номером вашей группы и вашим номером по журналу группы, например «Т-834 № 9».
24	1	Написать оконную программу, принимающую вещественные числа x и c , вычисляющую с ними выражение $(1,5*x^2 - 2*x*c - 9) / \text{Корень}(x - c^2)$, и выводящую полученный результат в отдельном поле. Обработать ввод пользователем некорректных данных.
	2	Сделать в программе два радиобаттона, устанавливающие окошку вашего приложения Sienna или Coral цвет.
	3	Сделать в вашей программе два чекбокса: 1) по нажатию на первый в поле заголовка окна отображается фраза с вашей фамилией, именем и отчеством, например «Иванова Алеся Андреевна»; 2) при нажатии второго чекбокса в окне приложения отображается фраза с номером вашей группы и вашим номером по журналу группы, например «Т-834 № 9».
25	1	Написать оконную программу, принимающую вещественные числа x и c , вычисляющую с ними выражение $\text{Корень}(x^3*c - 7,5*x + c/\text{Корень}(x))$, и выводящую полученный результат в отдельном поле. Обработать ввод пользователем некорректных данных.
	2	Сделать в программе два радиобаттона, устанавливающие окошку вашего приложения

		SeaGreen или Pink цвет.
	3	Сделать в вашей программе два чекбокса: 1) по нажатию на первый в поле заголовка окна отображается фраза с вашей фамилией, именем и отчеством, например «Иванова Алеся Андреевна»; 2) при нажатии второго чекбокса в окне приложения отображается фраза с номером вашей группы и вашим номером по журналу группы, например «Т-834 № 9».
26	1	Написать оконную программу, принимающую вещественные числа x и s , вычисляющую с ними выражение $(x^3 - 1,5*s^2) / (s + \text{Корень}(x))$, и выводящую полученный результат в отдельном поле. Обработать ввод пользователем некорректных данных и деления на ноль.
	2	Сделать в программе два радиобаттона, устанавливающие окошку вашего приложения SandyBrown или Olive цвет.
	3	Сделать в вашей программе два чекбокса: 1) по нажатию на первый в поле заголовка окна отображается фраза с вашей фамилией, именем и отчеством, например «Иванова Алеся Андреевна»; 2) при нажатии второго чекбокса в окне приложения отображается фраза с номером вашей группы и вашим номером по журналу группы, например «Т-834 № 9».
27	1	Написать оконную программу, принимающую вещественные числа x и s , вычисляющую с ними выражение $(\text{Корень}(x) - s) / 3*x$, и выводящую полученный результат в отдельном поле. Обработать ввод пользователем некорректных данных, деления на ноль и извлечения квадратного корня из отрицательного числа.
	2	Сделать в программе два радиобаттона, устанавливающие окошку вашего приложения Salmon или RosyBrown цвет.
	3	Сделать в вашей программе два чекбокса: 1) по нажатию на первый в поле заголовка окна отображается фраза с вашей фамилией, именем и отчеством, например «Иванова Алеся Андреевна»; 2) при нажатии второго чекбокса в окне приложения отображается фраза с номером вашей группы и вашим номером по журналу группы, например «Т-834 № 9».
28	1	Написать оконную программу, принимающую вещественные числа x и s , вычисляющую с ними выражение $(\text{КореньКвадратный}(x-2*s)) + 2*x + 1,9*s^2) / \text{Корень_кубический}(x)$, и выводящую полученный результат в отдельном поле. Обработать ввод пользователем некорректных данных, деления на ноль.
	2	Сделать в программе два радиобаттона, устанавливающие окошку вашего приложения Red или Aquamarine цвет.
	3	Сделать в вашей программе два чекбокса: 1) по нажатию на первый в поле заголовка окна отображается фраза с вашей фамилией, именем и отчеством, например «Иванова Алеся Андреевна»; 2) при нажатии второго чекбокса в окне приложения отображается фраза с номером вашей группы и вашим номером по журналу группы, например «Т-834 № 9».
29	1	Написать оконную программу, принимающую вещественные числа x и s , вычисляющую с ними выражение $(x^3 - 3,2*s) / (\text{корень}(x) - \text{корень}(s))$, и выводящую полученный результат в отдельном поле. Обработать ввод пользователем некорректных данных, деления на ноль, извлечения корня из отрицательного числа.
	2	Сделать в программе два радиобаттона, устанавливающие окошку вашего приложения Purple или Beige цвет.
	3	Сделать в вашей программе два чекбокса: 1) по нажатию на первый в поле заголовка окна отображается фраза с вашей фамилией, именем и отчеством, например «Иванова Алеся Андреевна»; 2) при нажатии второго чекбокса в окне приложения отображается фраза с номером вашей группы и вашим номером по журналу группы, например «Т-834 № 9».
30	1	Написать оконную программу, принимающую два вещественных числа, вычитающую из корня кубического первого числа корень квадратный из второго числа, прибавляющую к ним частное этих чисел, и выводящую полученный результат в отдельном поле. Обработать ввод пользователем некорректных данных.
	2	Сделать в программе два радиобаттона, устанавливающие окошку вашего приложения Plum или Bisque цвет.

	3 Сделать в вашей программе два чекбокса: 1) по нажатию на первый в поле заголовка окна отображается фраза с вашей фамилией, именем и отчеством, например «Иванова Алеся Андреевна»; 2) при нажатии второго чекбокса в окне приложения отображается фраза с номером вашей группы и вашим номером по журналу группы, например «Т-834 № 9».
--	---

6. Контрольные вопросы

1. Какие классы используются для создания пользовательского интерфейса в Windows-приложениях, выполняемых в среде CLR?
2. Для чего предназначен класс Form?
3. Какой метод уничтожает форму?
4. Зачем нужен MessageBox?
5. Чем текстБоксы отличаются от лэйблов?
6. Чем чекБоксы отличаются от радиоБаттонов?
7. Как создать оконное приложение в Release-версии?
8. Объясните основные участки кода в MyForm.h файле.
9. Где полезен группБокс?
10. Почему нежелательно ручное редактирование кода в MyForm.h файле?

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

Председатель ЦК _____ С.В. Банцевич