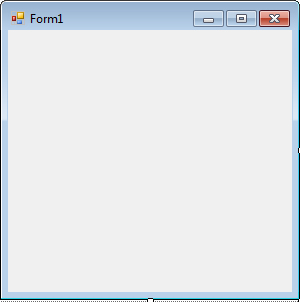
**Работа с Windows Form**

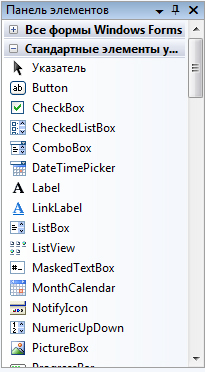
**1. Создание Windows Form (для MVS 2010 Express Edition)**

**Начало: Файл→Создать→Проект**, выбираем пункт **CLR**, выбираем **Приложение Windows Forms**, даем имя проекта, к примеру **factr** и нажимаем **Ok**.

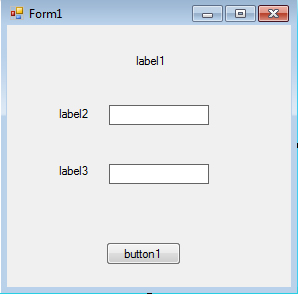
После того как мы создали проект, должна появиться пустая форма:



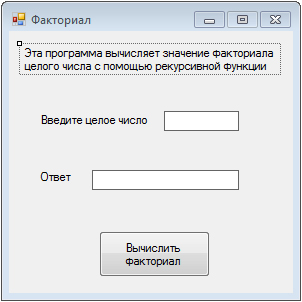
Справа должна появиться панель элементов, если ее нет, то можно включить ее в меню **Вид→Панель Элементов** или нажатием горячих клавиш — **Ctrl + Alt + X**.



Нам понадобятся элементы: **Button** (Кнопка), **Label** (Надпись) и **TextBox** (Текстовое поле). Перетащите на форму три элемента **Label**, Два элемента **TextBox**, и одну кнопку (**Button**):



Теперь нужно изменить текст надписей для элементов **Label** и **Button** и Задать подходящие имена классов для элементов **TextBox**. Для этого Выделите элемент **label1**, перейдите в **Панель свойств и задайте необходимые имена элементов:**



Теперь задайте для текстовых полей (**TextBox**) Атрибуты **Name** в **Панели Свойств** – для первого поля это будет **num1**, а для второго, соответственно **num2**. Тем самым мы изменили имена классов для полей **TextBox**.

Перейдем к написанию обработчика событий для кнопки **Button**, т.е. что будет происходить при нажатии на эту кнопку.

Для этого добавляем заголовочный файл **fact.h** в проект, для этого щелкаем правой кнопкой мыши в **Обозревателе решений** на папке **Заголовочные файлы**, далее меню **Добавить** -> **Создать Элемент**, вписываем название файла – **fact.h** и нажимаем **Добавить**. В данном файле будет содержаться прототип функции для вычисления факториала. Добавляем туда следующий исходный код:

long double fact(int N);

Далее подключаем этот файл к проекту с помощью директивы **#include** – Открываем файл **fact.cpp** и добавляем после строчки #include "stdafx.h" следующий код:

#include "fact.h"

Здесь же функция вычисления факториала:

#pragma once

#include "stdafx.h"

long double fact(int N)

{

if(N < 0)

return 0;

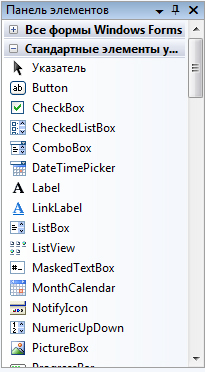
if (N == 0)

return 1;

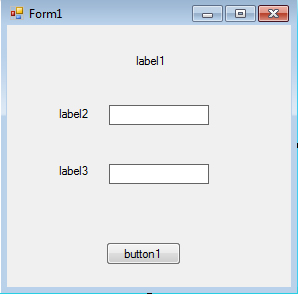
else

return N \* fact(N - 1);

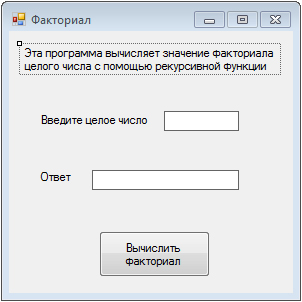
}



Нам понадобятся элементы: **Button** (Кнопка), **Label** (Надпись) и **TextBox** (Текстовое поле). Перетащите на форму три элемента **Label**, Два элемента **TextBox**, и одну кнопку (**Button**):



Теперь нужно изменить текст надписей для элементов **Label** и **Button** и Задать подходящие имена классов для элементов **TextBox**. Для этого Выделите элемент **label1**, перейдите в **Панель свойств и задайте необходимые имена элементов:**



Теперь задайте для текстовых полей (**TextBox**) Атрибуты **Name** в **Панели Свойств** – для первого поля это будет **num1**, а для второго, соответственно **num2**. Тем самым мы изменили имена классов для полей **TextBox**.

Перейдем к написанию обработчика событий для кнопки **Button**, т.е. что будет происходить при нажатии на эту кнопку.

Для этого добавляем заголовочный файл **fact.h** в проект, для этого щелкаем правой кнопкой мыши в **Обозревателе решений** на папке **Заголовочные файлы**, далее меню **Добавить** -> **Создать Элемент**, вписываем название файла – **fact.h** и нажимаем **Добавить**. В данном файле будет содержаться прототип функции для вычисления факториала. Добавляем туда следующий исходный код:

long double fact(int N);

Далее подключаем этот файл к проекту с помощью директивы **#include** – Открываем файл **fact.cpp** и добавляем после строчки #include "stdafx.h" следующий код:

#include "fact.h"

Здесь же функция вычисления факториала:

#pragma once

#include "stdafx.h"

long double fact(int N)

{

if(N < 0)

return 0;

if (N == 0)

return 1;

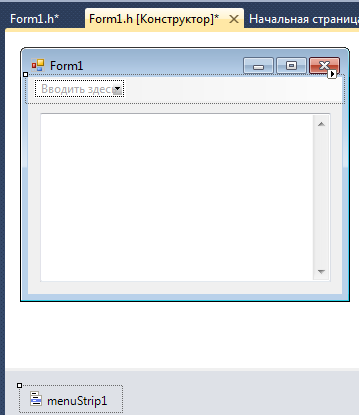
else

return N \* fact(N - 1);

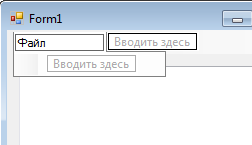
}

**2. Элемент MenuStrip и свойство Anchor в MVS C++**

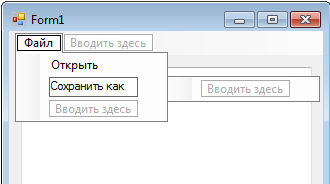
**"MenuStrip"** – это, по сути, выпадающее меню, в котором есть определённые пункты. Свойство **"Anchor"** – это свойство, при котором определяется к каким сторонам формы будет привязан элемент, если к правой, то при увеличении или уменьшении размера формы – правая сторона **"textBox"**, так же будет увеличиваться или уменьшаться. Если поставлена кнопка"button", то она будет перемещаться за стороной формы, к которой привязана. Перенесите на форму элемент **"MenuStrip"** и элемент **"textBox"**. У элемента “textBox” включите **"Multeline"** и свойство **"ScrollBars"->"Vertical"**:



Далее нажмите на текстовое поле **"Вводить здесь"** и напишите – **"Файл**:

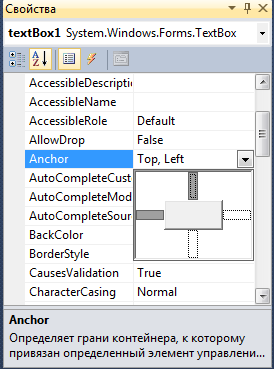


После этого в нижнем, новом появившемся текстовом поле, напишите **"Открыть"**, после чего появиться ещё одно, в котором напишите **"Сохранить как"**:

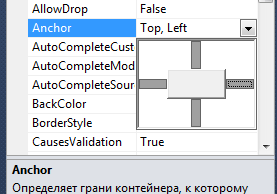


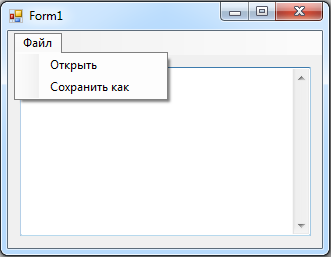
И в последнем текстовом поле напишите – **"Выход"**:

Перейдем к свойству **"Achor"** – для этого нажмите на **"textBox"**. После чего на панели свойств, расположенной слева, выберите свойство **"Achor"**:



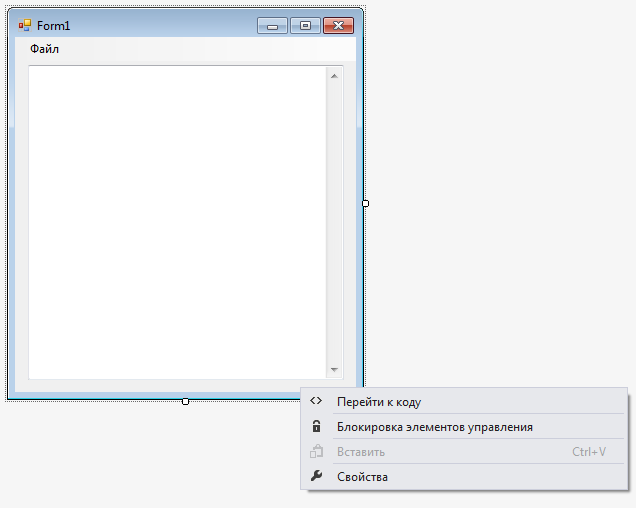
Как в можете видеть - “textBox” привязан к левой и верхней части формы. Теперь выделите так же правую и нижнюю часть, после чего размер элемента “textBox” будет изменяться пропорционально размеру формы:

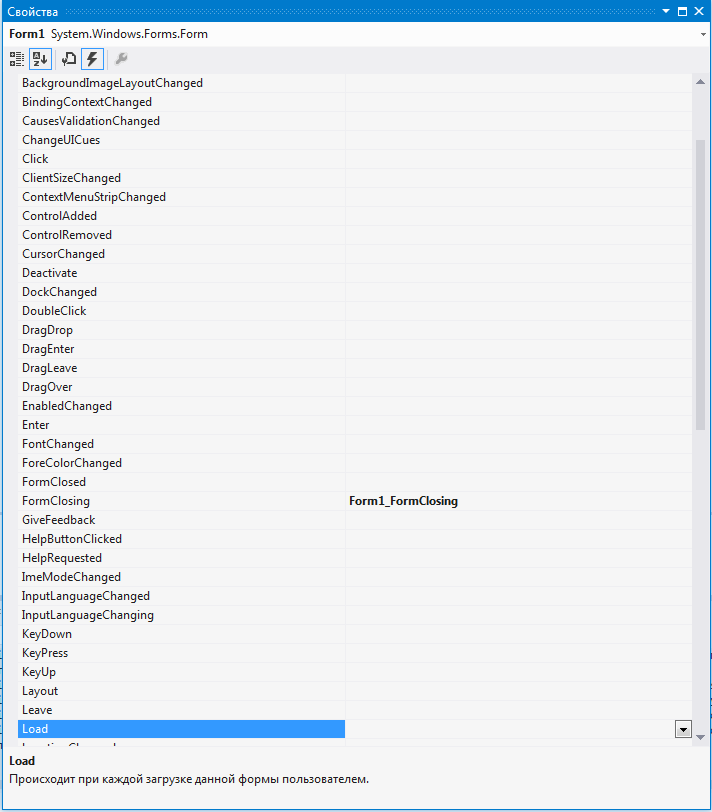


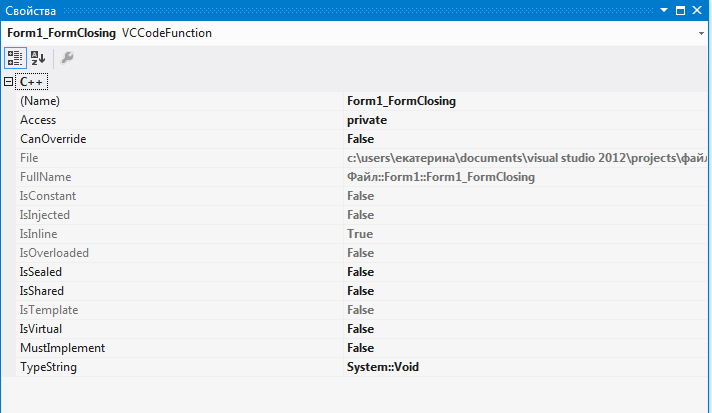
Запустите программу и проверьте изменённые свойства:  


## 2.1. Пошаговый разбор – обработка события Windows Forms FormClosing

Следующие шаги разбирают реализацию метода обработки события FormClosing:  
1.    С использованием Visual Studio создайте новый проект Windows Forms и добавьте новую форму с именем Form1.  
2.    Открыв Form1 в режиме конструктора, щелкните правой кнопкой мыши по открытому участку формы (не по элементу управления), и выберите "Свойства" из всплывающего меню. Появится диалоговое окно "Свойства", отображающее свойства формы или события. Если это еще не выбрано, нажмите кнопку "События" (она имеет иконку светящейся молнии) на панели инструментов вверху диалогового окна свойства.

  
3.    В диалоговом окне события найдите событие, на которое ваше приложение должно реагировать. В нашем случае это событие FormClosing. Дважды щелкните где-нибудь в строке, в которой указано FormClosing.





Код для обработчика события FormClosing:

private: System::Void Form1\_FormClosing(System::Object^ sender, System::Windows::Forms::FormClosingEventArgs^ e) {

if (textBox1->Modified == false) return;

auto MeBox = MessageBox::Show("Текст был изменён. \nСохранить изменения?",

"Закрыть", MessageBoxButtons::YesNoCancel, MessageBoxIcon::Exclamation);

if (MeBox == Windows::Forms::DialogResult::No) return;

if (MeBox == Windows::Forms::DialogResult::Cancel) e->Cancel = true;

if (MeBox == Windows::Forms::DialogResult::Yes)

{

if (saveFileDialog1->ShowDialog() == Windows::Forms::DialogResult::OK)

{

Save(); return;

}

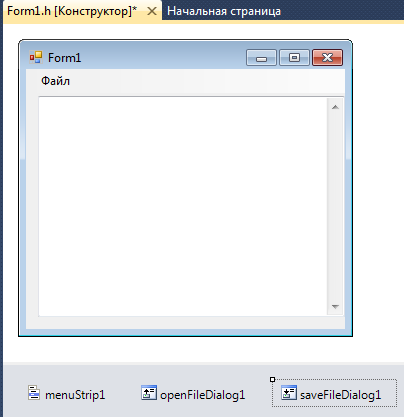
else e->Cancel = true;

}

}

**2.2. Открытие и запись текстового файла в MVS C++**

В данной программе мы будем открывать какой-то уже созданный текстовый файл и редактировать его или же сами писать текст и сохранять его, как новый текстовый файл в нужную вам папку. Ещё один явный признак текстового редактора – это если вы что-то написали и нажимайте крестик чтобы выйти – при этом программа спрашивает: “Сохранить изменения”. Для этого необходимы следующие элементы: **"MenuStrip"**, **"textBox"**, **"openFiledialog"**, **"saveFiledialog"**. Перетащите все эти элементы на форму, назовите заголовок "MenuStrip" **"Файл"** создайте в нём три пункта: **"Открыть"**, **"Сохранить как"**, **"Выход"**, привяжите **"textBox"** ко всем сторонам формы, если не знаете включите **"Multeline"** и **"ScrollBars->Vertical"** (все указанное описано выше). Оформление программы:



В коде программы будут созданы **"MyReader"** и **"MyWriter"** , с помощью которых программа будет читать и записывать текст в файл. Помимо этого в коде создаётся кодировка, благодаря которой программа будет понимать русский текст. У формы нужно вызвать событие **"FormClousing"**, о котором было рассказано ранее.

**2.3. Код программы**

**C++ (файл Form1.h)**

#pragma once

namespace Файл {

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 Form1

/// </summary>

public ref class Form1 : public System::Windows::Forms::Form

{

public:

Form1(void)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~Form1()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::MenuStrip^ menuStrip1;

protected:

private: System::Windows::Forms::ToolStripMenuItem^ файлToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ открытьToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ сохранитьКакToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ выходToolStripMenuItem;

private: System::Windows::Forms::TextBox^ textBox1;

private: System::Windows::Forms::OpenFileDialog^ openFileDialog1;

private: System::Windows::Forms::SaveFileDialog^ saveFileDialog1;

private:

/// <summary>

/// Требуется переменная конструктора.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Обязательный метод для поддержки конструктора - не изменяйте

/// содержимое данного метода при помощи редактора кода.

/// </summary>

void InitializeComponent(void)

{

this->menuStrip1 = (gcnew System::Windows::Forms::MenuStrip());

this->файлToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->открытьToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->сохранитьКакToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->выходToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->textBox1 = (gcnew System::Windows::Forms::TextBox());

this->openFileDialog1 = (gcnew System::Windows::Forms::OpenFileDialog());

this->saveFileDialog1 = (gcnew System::Windows::Forms::SaveFileDialog());

this->menuStrip1->SuspendLayout();

this->SuspendLayout();

//

// menuStrip1

//

this->menuStrip1->Items->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(1) {this->файлToolStripMenuItem});

this->menuStrip1->Location = System::Drawing::Point(0, 0);

this->menuStrip1->Name = L"menuStrip1";

this->menuStrip1->Size = System::Drawing::Size(341, 24);

this->menuStrip1->TabIndex = 0;

this->menuStrip1->Text = L"Файл";

//

// файлToolStripMenuItem

//

this->файлToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(3) {this->открытьToolStripMenuItem,

this->сохранитьКакToolStripMenuItem, this->выходToolStripMenuItem});

this->файлToolStripMenuItem->Name = L"файлToolStripMenuItem";

this->файлToolStripMenuItem->Size = System::Drawing::Size(48, 20);

this->файлToolStripMenuItem->Text = L"Файл";

this->файлToolStripMenuItem->Click += gcnew System::EventHandler(this, &Form1::файлToolStripMenuItem\_Click);

//

// открытьToolStripMenuItem

//

this->открытьToolStripMenuItem->Name = L"открытьToolStripMenuItem";

this->открытьToolStripMenuItem->Size = System::Drawing::Size(153, 22);

this->открытьToolStripMenuItem->Text = L"Открыть";

this->открытьToolStripMenuItem->Click += gcnew System::EventHandler(this, &Form1::открытьToolStripMenuItem\_Click);

//

// сохранитьКакToolStripMenuItem

//

this->сохранитьКакToolStripMenuItem->Name = L"сохранитьКакToolStripMenuItem";

this->сохранитьКакToolStripMenuItem->Size = System::Drawing::Size(153, 22);

this->сохранитьКакToolStripMenuItem->Text = L"Сохранить как";

this->сохранитьКакToolStripMenuItem->Click += gcnew System::EventHandler(this, &Form1::сохранитьКакToolStripMenuItem\_Click);

//

// выходToolStripMenuItem

//

this->выходToolStripMenuItem->Name = L"выходToolStripMenuItem";

this->выходToolStripMenuItem->Size = System::Drawing::Size(153, 22);

this->выходToolStripMenuItem->Text = L"Выход";

this->выходToolStripMenuItem->Click += gcnew System::EventHandler(this, &Form1::выходToolStripMenuItem\_Click);

//

// textBox1

//

this->textBox1->Anchor = static\_cast<System::Windows::Forms::AnchorStyles>((((System::Windows::Forms::AnchorStyles::Top | System::Windows::Forms::AnchorStyles::Bottom)

| System::Windows::Forms::AnchorStyles::Left)

| System::Windows::Forms::AnchorStyles::Right));

this->textBox1->Location = System::Drawing::Point(13, 28);

this->textBox1->Multiline = true;

this->textBox1->Name = L"textBox1";

this->textBox1->ScrollBars = System::Windows::Forms::ScrollBars::Vertical;

this->textBox1->Size = System::Drawing::Size(316, 315);

this->textBox1->TabIndex = 1;

//

// openFileDialog1

//

this->openFileDialog1->FileName = L"openFileDialog1";

//

// Form1

//

this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->ClientSize = System::Drawing::Size(341, 355);

this->Controls->Add(this->textBox1);

this->Controls->Add(this->menuStrip1);

this->MainMenuStrip = this->menuStrip1;

this->Name = L"Form1";

this->Text = L"Form1";

this->FormClosing += gcnew System::Windows::Forms::FormClosingEventHandler(this, &Form1::Form1\_FormClosing);

this->menuStrip1->ResumeLayout(false);

this->menuStrip1->PerformLayout();

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

private: System::Void файлToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

this->Text = "Текстовый редактор";

openFileDialog1->FileName = "D:\\РАСПИСАНИЕ\\Text.txt";

openFileDialog1->Filter = "Текстовые файлы (\*.txt)|\*.txt|All files (\*.\*)|\*.\*";

saveFileDialog1->Filter = "Текстовые файлы (\*.txt)|\*.txt|All files (\*.\*)|\*.\*";

}

private: System::Void открытьToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

openFileDialog1->ShowDialog();

if (openFileDialog1->FileName == nullptr) return;

try

{

auto MyReader = gcnew IO::StreamReader(openFileDialog1->FileName, System::Text::Encoding::GetEncoding(1251));

textBox1->Text= MyReader->ReadToEnd();

MyReader->Close();

}

catch (IO::FileNotFoundException^ Ситуация)

{

MessageBox::Show(Ситуация->Message + "\nФайл не найден", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Exclamation);

}

catch (Exception^ Ситуация)

{

MessageBox::Show(Ситуация->Message, "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Exclamation);

}

}

private: System::Void сохранитьКакToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

saveFileDialog1->FileName = openFileDialog1->FileName;

if (saveFileDialog1->ShowDialog() == Windows::Forms::DialogResult::OK) Save();

}

void Save()

{

try

{

// Создание экземпляра StreamWriter для записи в файл:

auto MyWriter = gcnew IO::StreamWriter(saveFileDialog1->FileName, false, System::Text::Encoding::GetEncoding(1251));

MyWriter->Write(textBox1->Text);

MyWriter->Close(); textBox1->Modified = false;

}

catch (Exception^ Ситуация)

{

MessageBox::Show(Ситуация->Message, "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Exclamation);

}

}

private: System::Void выходToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

this->Close();

}

private: System::Void Form1\_FormClosing(System::Object^ sender, System::Windows::Forms::FormClosingEventArgs^ e) {

if (textBox1->Modified == false) return;

auto MeBox = MessageBox::Show("Текст был изменён. \nСохранить изменения?",

"Закрыть", MessageBoxButtons::YesNoCancel, MessageBoxIcon::Exclamation);

if (MeBox == Windows::Forms::DialogResult::No) return;

if (MeBox == Windows::Forms::DialogResult::Cancel) e->Cancel = true;

if (MeBox == Windows::Forms::DialogResult::Yes)

{

if (saveFileDialog1->ShowDialog() == Windows::Forms::DialogResult::OK)

{

Save(); return;

}

else e->Cancel = true;

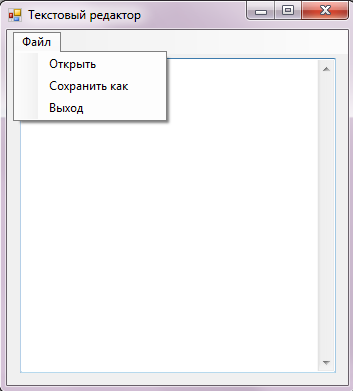
}

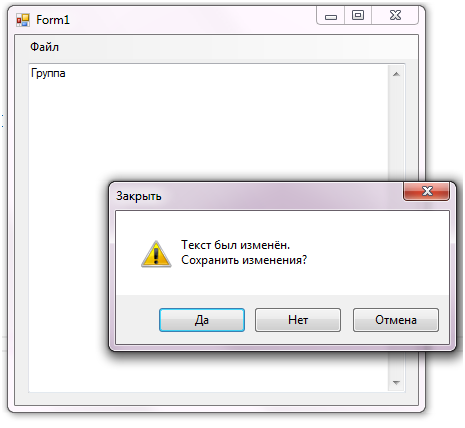
}

};

}

Пример работы:





# 2.4. Краткая справка (режим доступа - http://msdn.microsoft.com/ru-ru/library/System.Windows.Forms.Form%28v=vs.110%29.aspx)

# А) System.Windows - пространства имен

Пространства имен System.Windows содержат типы, используемые в приложениях Windows Presentation Foundation (WPF), включая клиенты анимации, элементы управления пользовательского интерфейса, привязку данных и преобразование типов. Пространство имен System.Windows.Forms и его дочерние пространства имен служат для разработки приложений Windows Forms.

# Б) System.Windows.Forms - пространство имен

Пространство имен **System.Windows.Forms** содержит классы для создания приложений Windows, которые позволяют наиболее эффективно использовать расширенные возможности пользовательского интерфейса, доступные в операционной системе Microsoft Windows.

# В) Form - класс

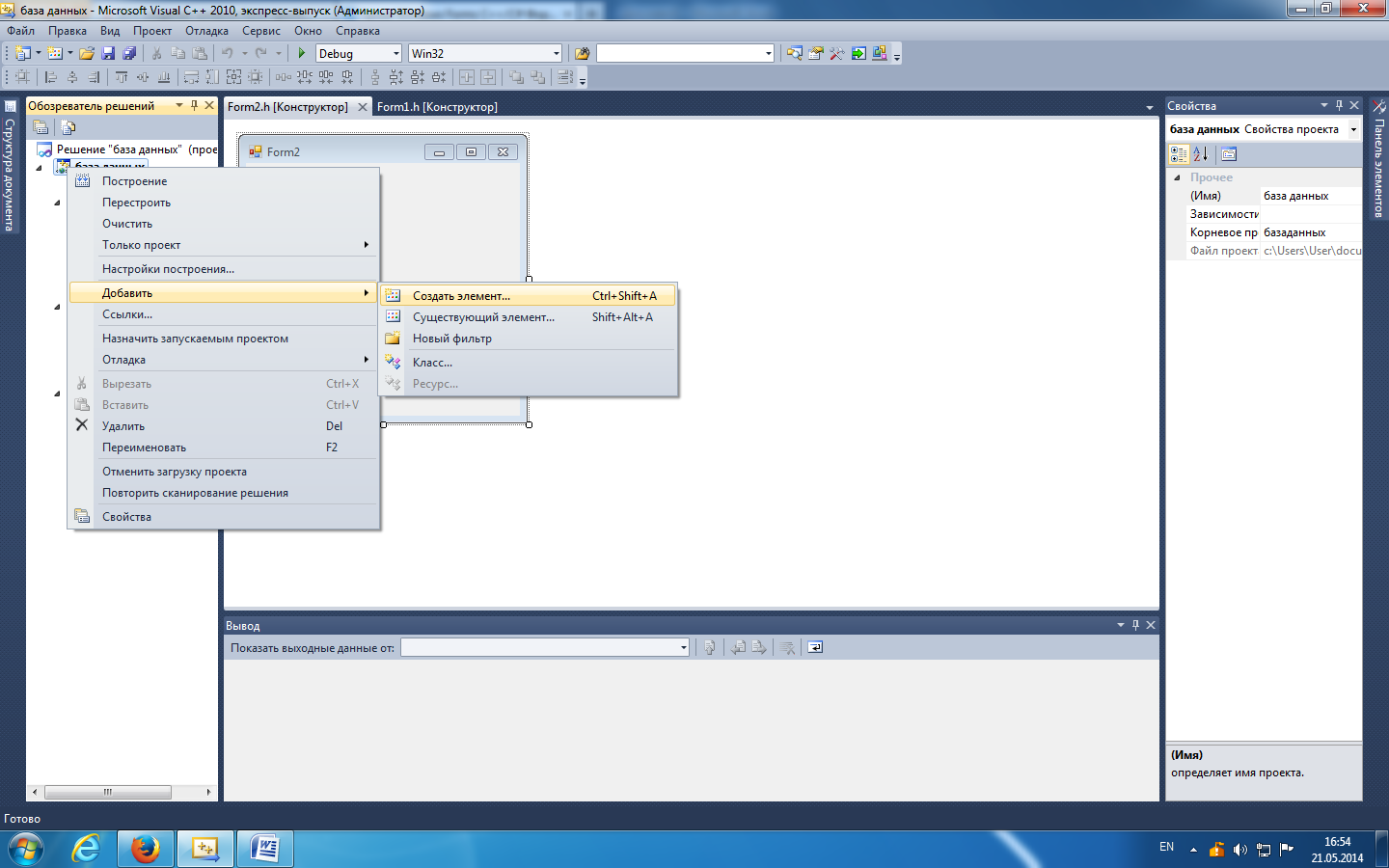
Представляет окно или диалоговое окно, которое составляет пользовательский интерфейс приложения.

[Иерархия наследования](javascript:void(0))

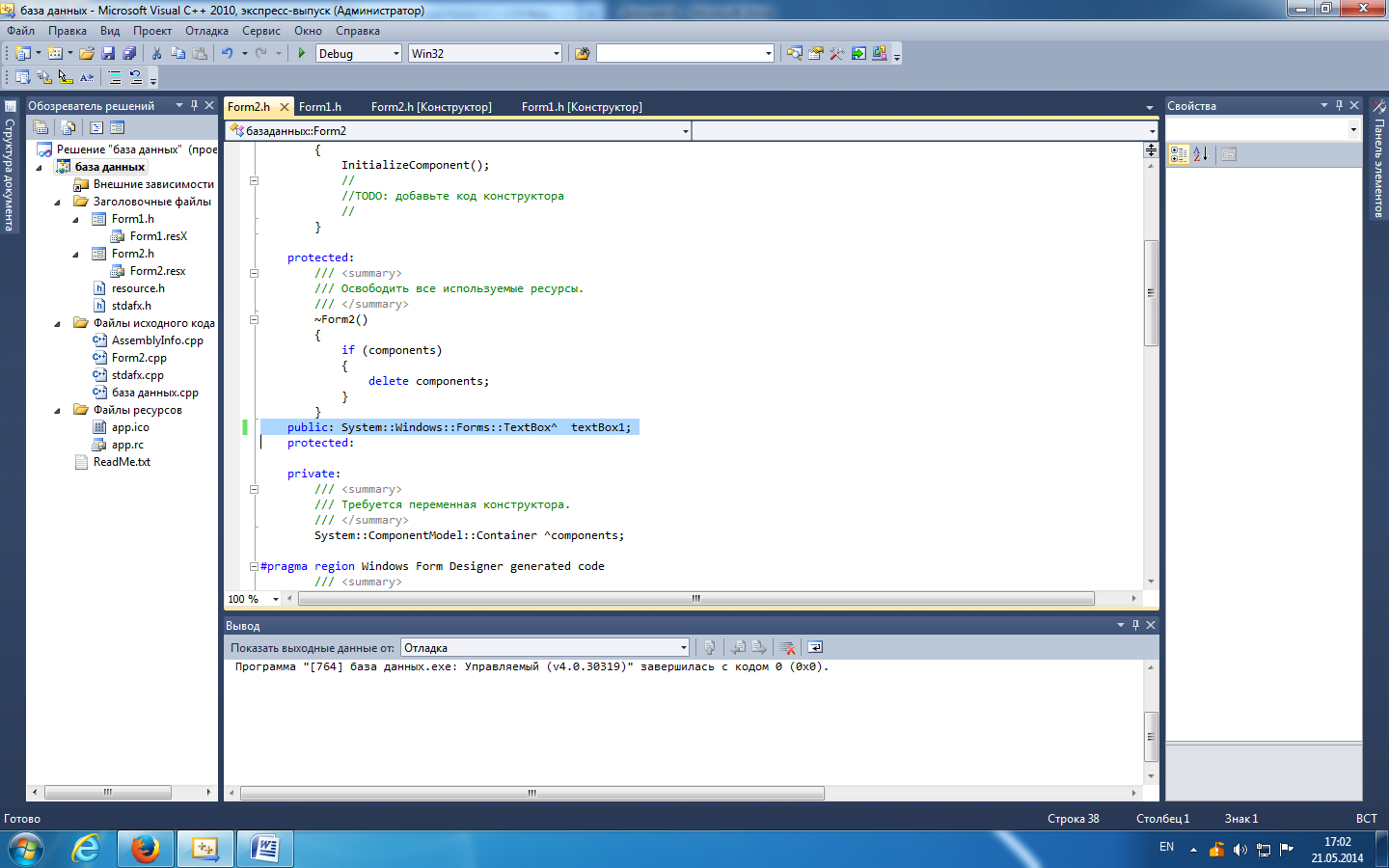
[System::Object](http://msdn.microsoft.com/ru-ru/library/system.object%28v=vs.110%29.aspx)   
  [System::MarshalByRefObject](http://msdn.microsoft.com/ru-ru/library/system.marshalbyrefobject%28v=vs.110%29.aspx)  
    [System.ComponentModel::Component](http://msdn.microsoft.com/ru-ru/library/system.componentmodel.component%28v=vs.110%29.aspx)  
      [System.Windows.Forms::Control](http://msdn.microsoft.com/ru-ru/library/system.windows.forms.control%28v=vs.110%29.aspx)  
        [System.Windows.Forms::ScrollableControl](http://msdn.microsoft.com/ru-ru/library/system.windows.forms.scrollablecontrol%28v=vs.110%29.aspx)  
          [System.Windows.Forms::ContainerControl](http://msdn.microsoft.com/ru-ru/library/system.windows.forms.containercontrol%28v=vs.110%29.aspx)  
            System.Windows.Forms::Form

**3. Объединение двух форм**

Создайте проект в приложении **«Windows Forms».**Теперь нужно создать вторую форму в этом же проекте: для этого необходимо щелкнуть правой кнопкой по имени проекта, выбрать **«Добавить»**, после чего откроется новою меню, в котором нужно будет выбрать «**Создать элемент»**, после этого появится новое окно, в котором уже будет выделен элемент **«Форма Windows Forms»**, вам нужно будет назвать его –**«Form2»**.



После этого на панели появится вторая форма конструктора. Суть учебного примера следующая – на **"Form1"** находится текстовое поле и кнопка, в текстовое поле вводится число, нажимается кнопка, введенное число возводится в квадрат, после чего появляется **"Form2"** на котором тоже есть текстовое поле, где будет показан результат возведения числа в квадрат. Важно понять следующее – поскольку элемент **"textBox1"** будет находиться на обеих формах, то его нужно объявить как **public** в коде формы **"Form2"**, а в коде формы **"Form1"** в самом верху кода подключить библиотеку второй формы **#include "Form2":**



Начало файла “Form1.h”:

#pragma once

#include "Form2.h"

Обработчик событий «Кнопка»:

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

Single a, b = 0;

Single A = Single::TryParse(textBox1->Text, System::Globalization::NumberStyles::Number, System::Globalization::NumberFormatInfo::CurrentInfo, a);

b = a\*a;

textBox1->Text = String::Format("Равно: {0:F5}", b);

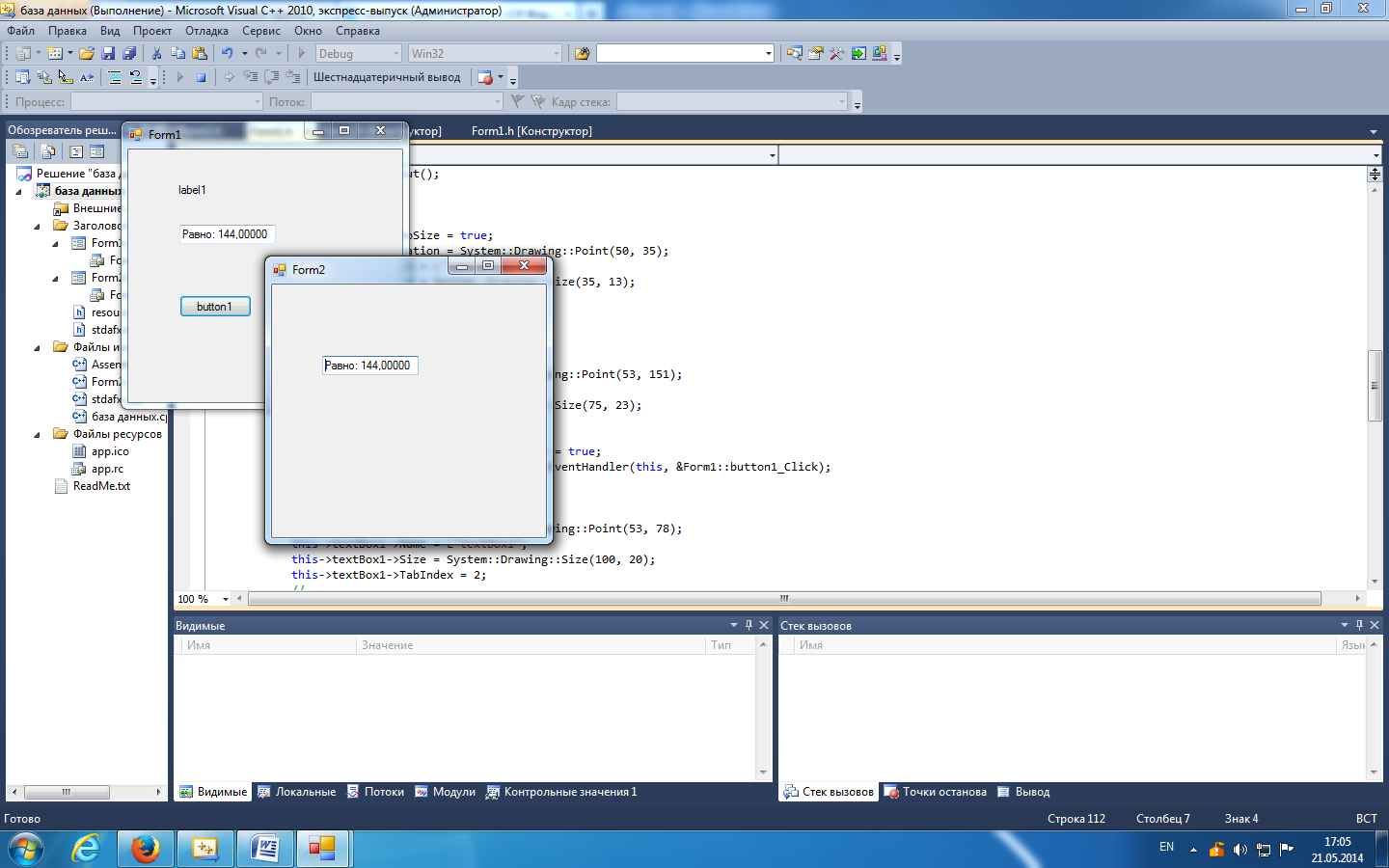
Form2^ gform2 = gcnew Form2;

gform2->Show();

gform2->textBox1->Text = this->textBox1->Text;

}

Пример работы:



**Элемент DataGrindView**

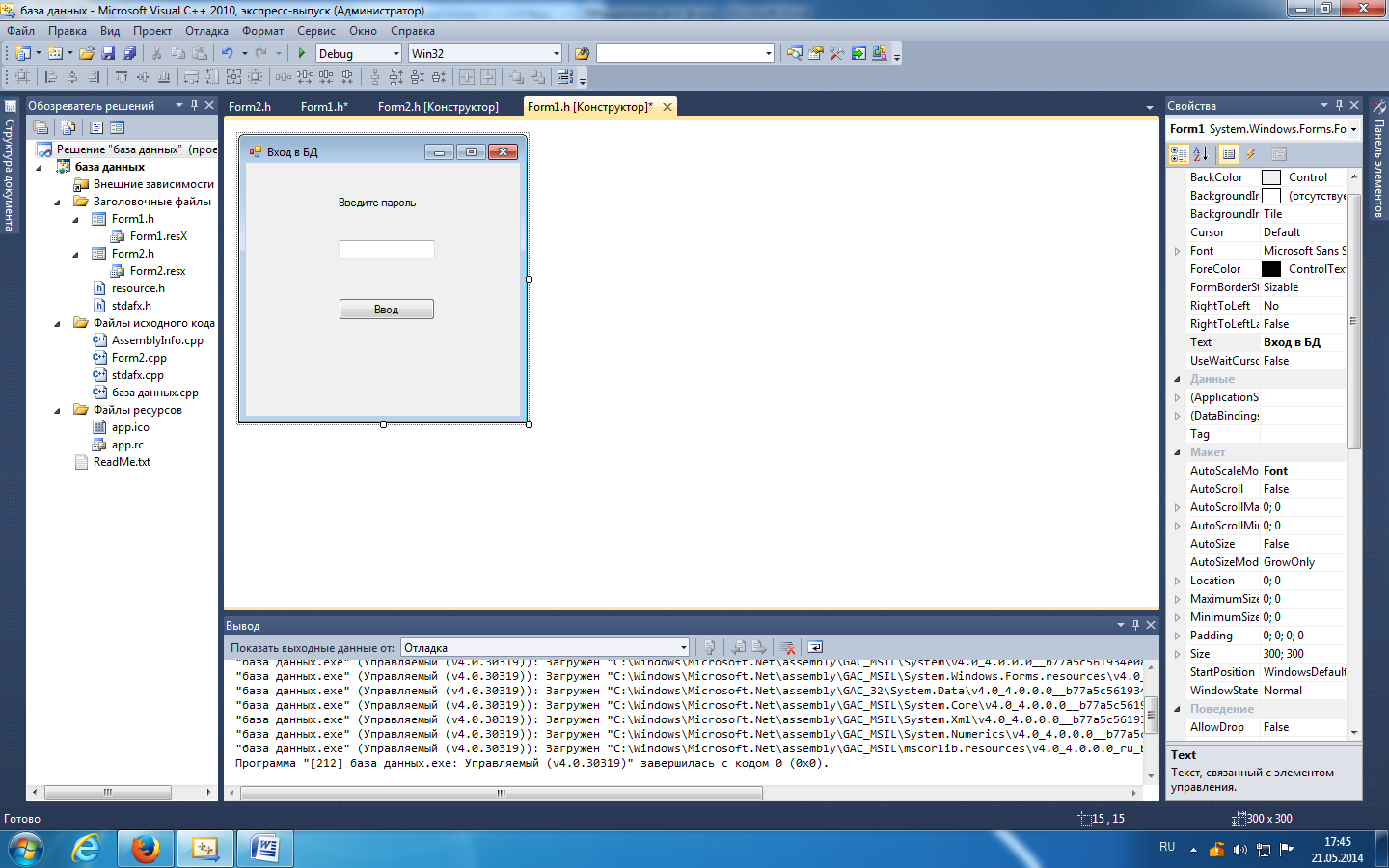
**Задание:**

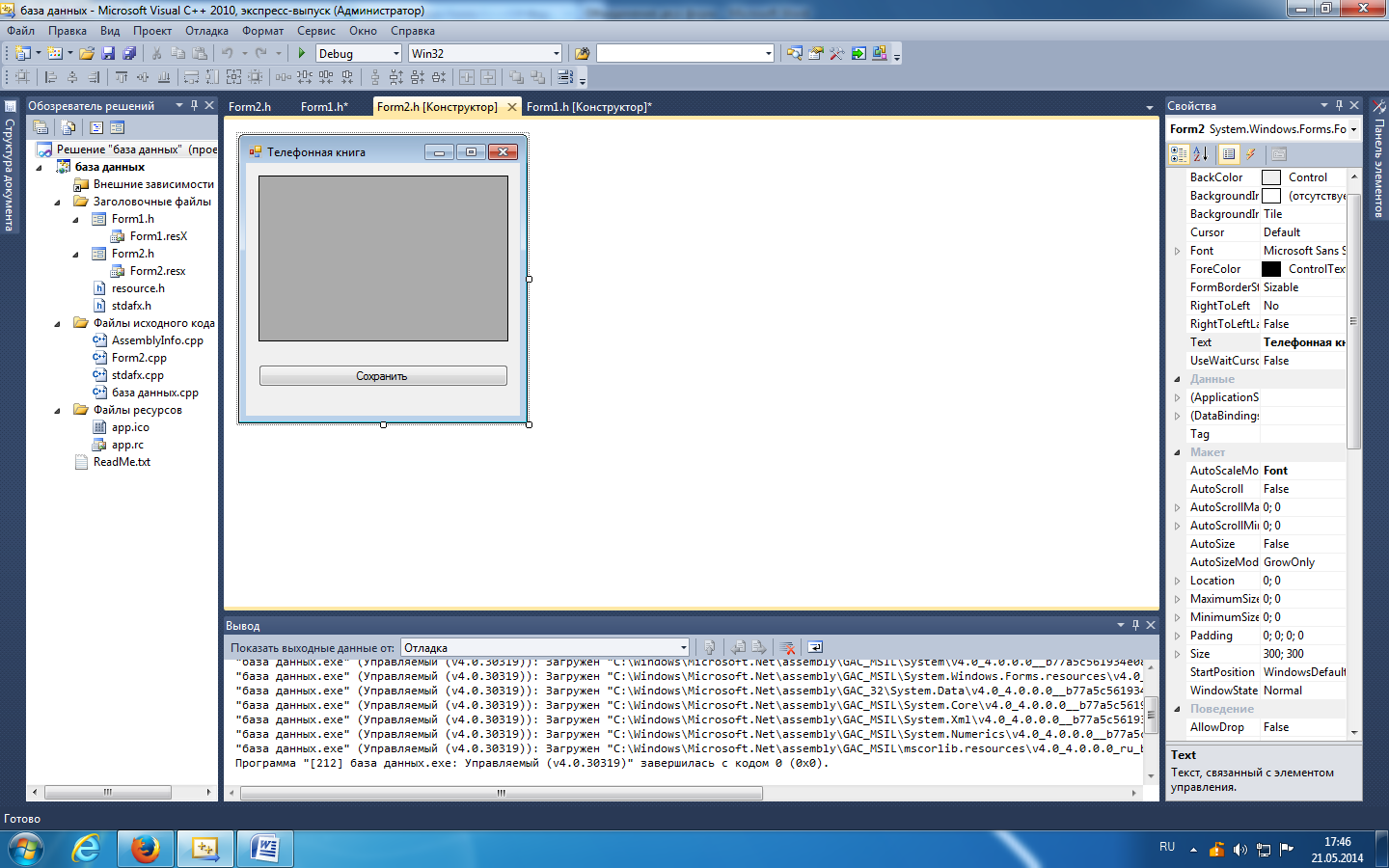
Создать телефонную книгу: на **"Form1"** находится тестовое поле, куда нужно ввести пароль, и кнопка, после нажатия которой программа проверяет введённый пароль. Если пароль **admin** то появится вторая форма, где будет находится таблица и активная кнопка **"Сохранить"**. В таблицу вводятся данные, нажимается кнопка – после чего в папке проекта создаётся документ **baza.xml**, в котором будут храниться данные, которые можно дополнять или удалять, сохраняя изменения. Если пользователь вводит пароль “reader”, то появляется вторая форма **"Form2"** с той же таблицей и данными, но кнопка “Сохранить” неактивна – пользователь может только просматривать данные, но не изменять. Если же пароль неверный – то появляется окно **"Incorrect password"**.

**Выполнение:**

Создайте проект в приложении **"Windows Forms"**: на **"Form1"** перенесите – **"label"**, **"textBox"** и **"button"**; Добавьте новую форму **"form2"** и перенесите на неё – **"DataGrindView"** и **"button"**.

Созданные формы:

****

****

**Файл “Form1.h”:**

#pragma once

#include "Form2.h"

.

.

.

#pragma endregion

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

Form2^ gform2 = gcnew Form2;

gform2->Show();

if (textBox1->Text == "reader")

{

gform2->Visible = true;

this->Visible = false;

gform2->button1->Enabled = false;

}

if (textBox1->Text == "admin")

{

gform2->Visible = true;

this->Visible = false;

gform2->button1->Enabled = true;

}

if (textBox1->Text != "reader" && textBox1->Text != "admin")

{

gform2->Visible = false;

MessageBox::Show("Incorrect password!");

}

}

private: System::Void Form1\_Load(System::Object^ sender, System::EventArgs^ e) {

this->Text = "Вход в БД";

textBox1->PasswordChar = '\*';

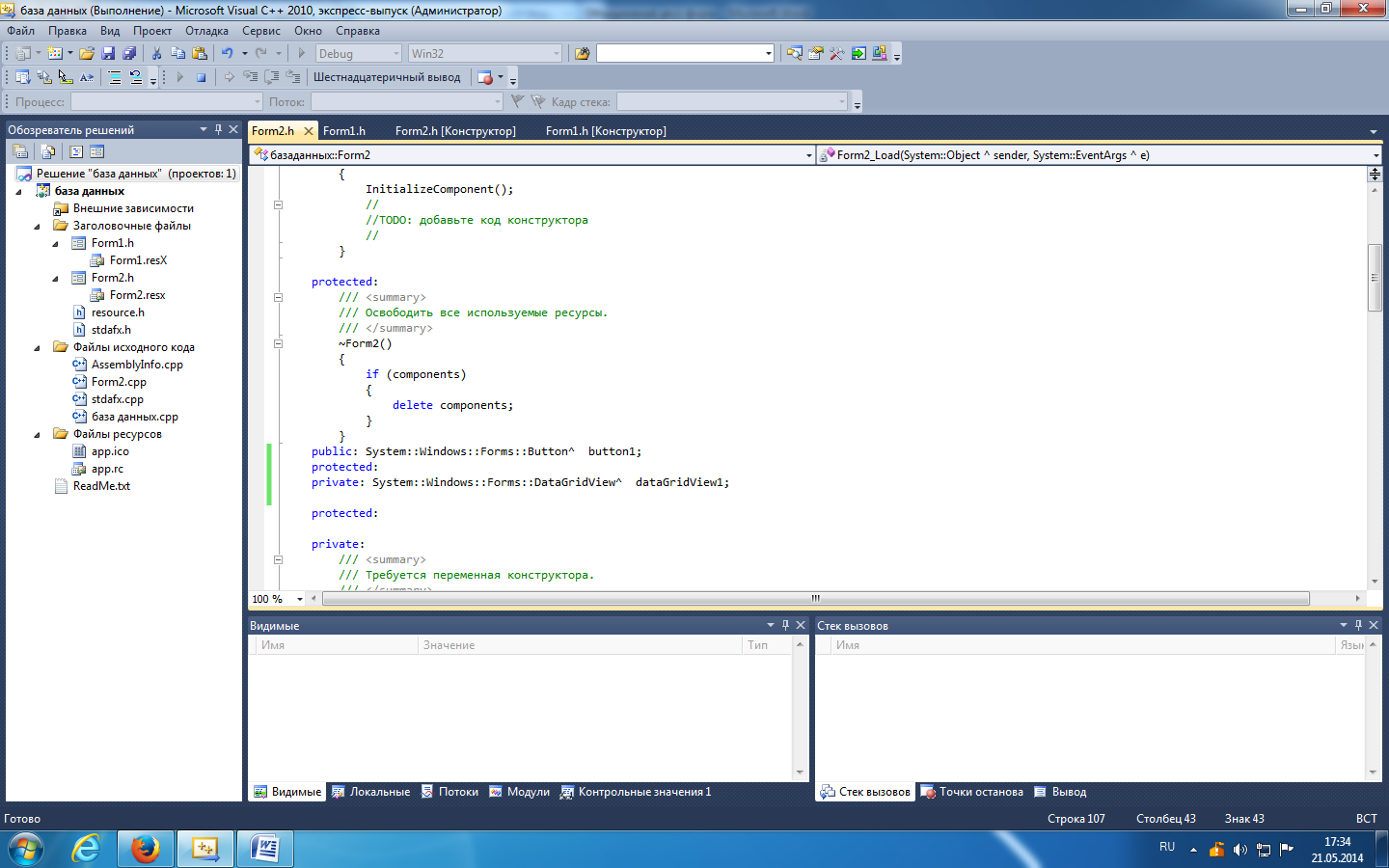
}

};

}

**Файл “Form2.h”:**

**Проверьте установку свойства public для button1:**

****

#pragma endregion

DataTable ^ Таблица;

DataSet ^ ВводДанных;

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

Таблица->TableName = "Название таблицы";

ВводДанных->WriteXml("baza.xml");

}

private: System::Void Form2\_Load(System::Object^ sender, System::EventArgs^ e) {

this->Text = "Телефонная книга";

button1->Text = "Сохранить";

this->Visible = false;

Таблица = gcnew DataTable();

ВводДанных = gcnew DataSet();

if (IO::File::Exists("baza.xml") == false)

{

dataGridView1->DataSource = Таблица;

Таблица->Columns->Add("Имена");

Таблица->Columns->Add("Номера Телефонов");

ВводДанных->Tables->Add(Таблица);

}

else

{

ВводДанных->ReadXml("baza.xml");

String ^ СтрокаXML = ВводДанных->GetXml();

dataGridView1->DataMember = ("Название таблицы");

dataGridView1->DataSource = ВводДанных;

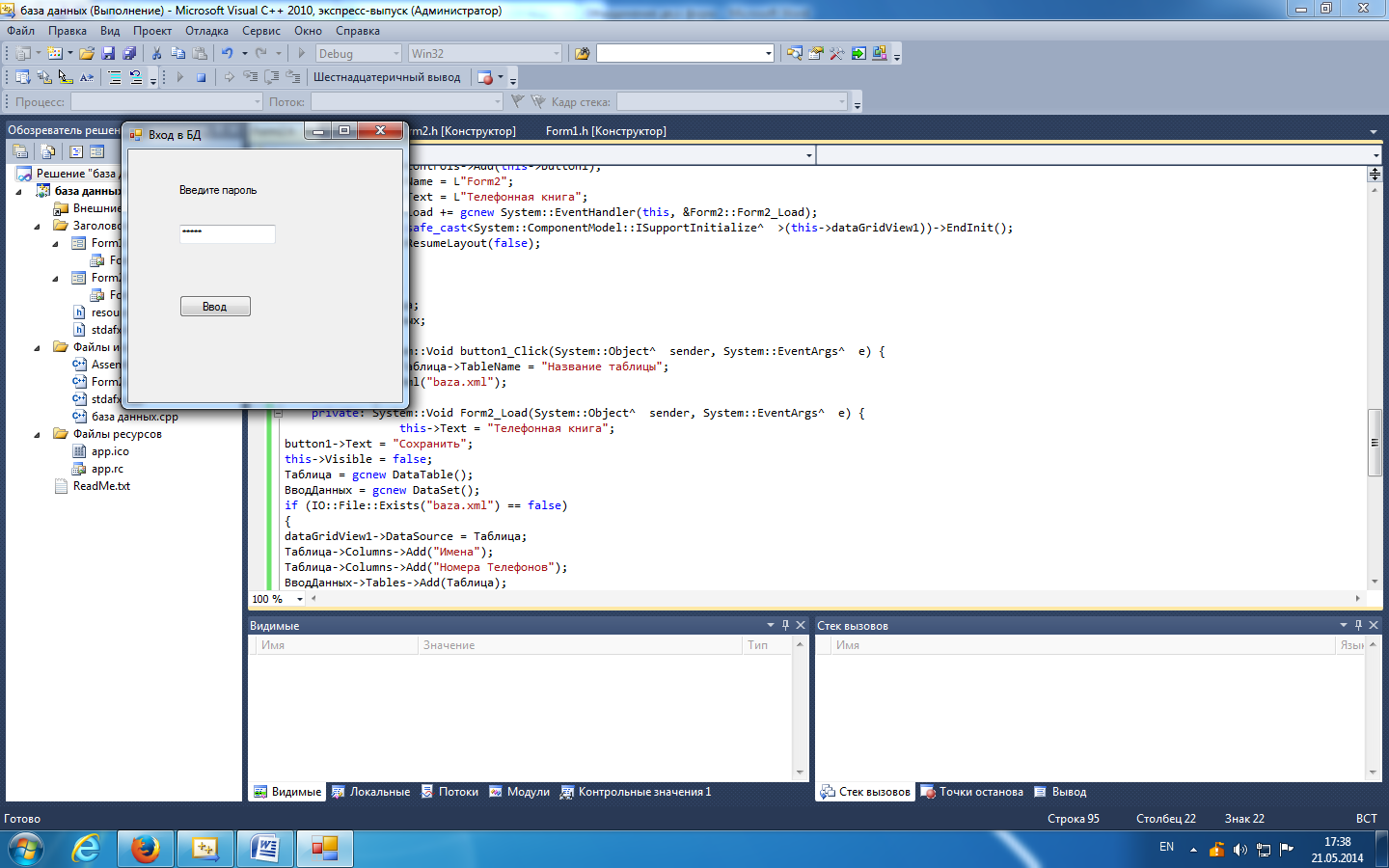
}

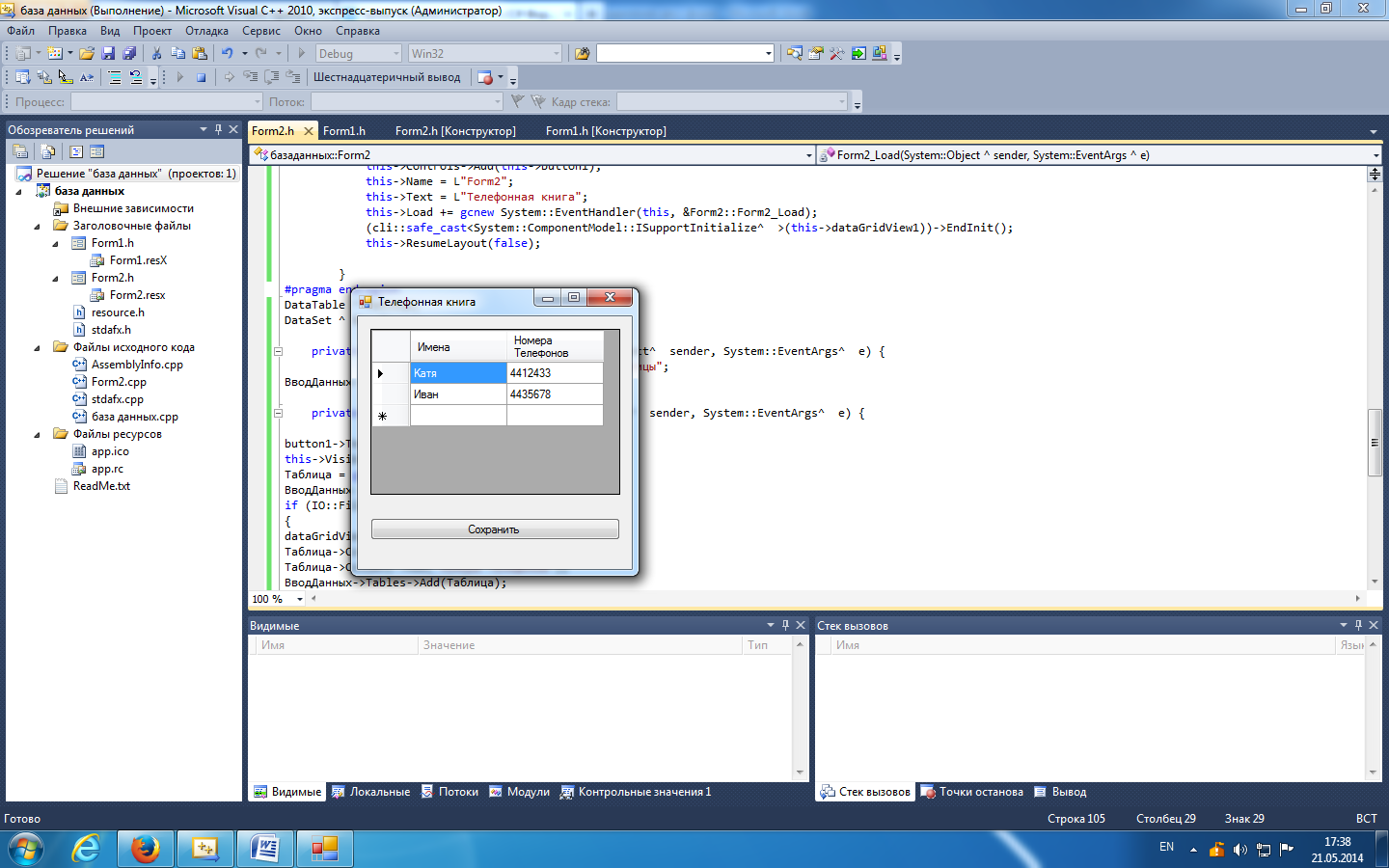
}

};

}

**Пример работы:**





Предупреждение: когда первая форма становится невидима, появляется вторая, но закрыв вторую форму первая не закроется.

**Задания для выполнения курсовой работы/проекта**

Создать приложение Windows Form для решения задачи:

1. решение квадратного уравнения.
2. вычисление основных тригонометричеких функций (sin, cos, tg) угла.
3. сортировка одномерного массива по возрастания и убыванию методом "пузырька". Массив заполняется случайным образом.
4. нахождение минимального и максимального элементов одномерного массива. Массив заполняется случайным образом.
5. вычисление значения *а^b.*
6. сложение и умножение двух матриц. Матрицы заполняются случайным образом.
7. нахождение минимального и максимального элементов двумерного массива. Массив заполняется случайным образом.
8. нахождение суммы и произведения элементов массива. Массив заполняется случайным образом.
9. найти сумму элементов матрицы, стоящих на главной диагонали. Матрица заполняется случайным образом.
10. найти сумму элементов матрицы, стоящих на побочной диагонали. Матрица заполняется случайным образом.
11. преобразовать матрицу А ( m×n) так, чтобы строки с нечетными индексами были упорядочены по убыванию, с четными – по возрастанию. Матрица заполняется случайным образом.
12. преобразовать матрицу А (m×n) таким образом, чтобы каждый столбец был упорядочен по убыванию. Матрица заполняется случайным образом.
13. написать программу, которая обрабатывает результаты экзамена. Для каждой оценки программа должна вычислить процент от общего количества оценок. Количество пятерок, четверок, троек и двоек задается.
14. решение кубического уравнения.
15. нахождение определителя квадратной матрицы. Матрица заполняется случайным образом.
16. сформировать массив, состоящий из минимальных элементов матрицы по строке. Матрица заполняется случайным образом.
17. сформировать массив, состоящий из максимальных элементов матрицы по строке. Матрица заполняется случайным образом.
18. вычислить произведение матрицы на вектор. Матрица и вектор задаются случайным образом.
19. найти кубический корень x.
20. сортировка одномерного массива по возрастания и убыванию методом "вставок". Массив заполняется случайным образом.

Пояснительная записка к курсовой работе/проекту должна содержать:

1. Порядок выполнения работы.

2. Код программы.

3. Пример работы программы.

4. Работающее приложение выслать на электронную почту: hutorianka@yandex.ru