

## Практическая работа №24,25

### Тема: Разработка интерфейса приложения. Создание базы данных и работа с ней

**Цель работы:** изучение приемов создания пользовательского интерфейса и работы с базами данных в Visual Studio

**Задачи:**

- изучение и использование элементов создания интерфейса пользователя;
- изучение приемов подключения, создания базы данных и работы с ней.

**Материально-техническое обеспечение:**

**Место проведения:** Компьютерный класс.

**Время на выполнение работы:** 4 часа.

**Оборудование:** ПК

**Средства обучения:** операционная система, текстовый процессор MS Word, программные средства определенного вида Visual Studio

**Исходные данные:**

1. Конспект занятия.
2. Задание для практической работы.

**Перечень справочной литературы:**

1) Программирование на языке высокого уровня. Программирование на языке C++: учеб. пособие / Т. И. Немцова, С. Ю. Голова, А. И. Терентьев ; под ред. Л. Г. Гагариной. – М. : ИД «ФОРУМ» : ИНФРА-М, 2018. – 512 с. – (Среднее профессиональное образование).

**Краткие теоретические сведения:**

**ADO.NET** — это технология доступа к данным от Microsoft [.Net Framework](#), которая обеспечивает связь между реляционными и нереляционными системами через общий набор компонентов.

**Программирование баз данных на C#**

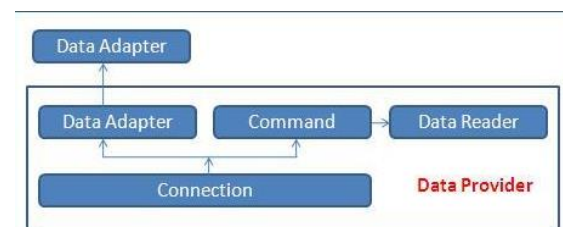
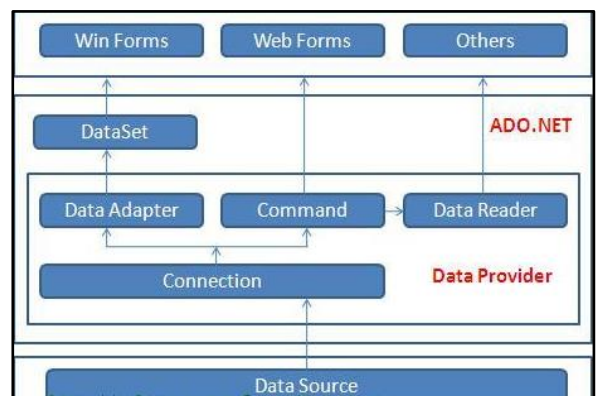
Двумя ключевыми компонентами ADO.NET являются **поставщики данных** и **набор данных**. .Net Framework включает в основном три поставщика данных для ADO.NET. Microsoft **SQL Server**, **OleDb** и **ODBC** являются основными поставщиками данных в .Net Framework. На следующих страницах вы можете подробно ознакомиться с каждым компонентом ADO.NET и программированием баз данных с помощью **исходного кода C#**.

Четыре объекта из .Net Framework обеспечивают функциональность поставщиков данных в ADO.NET. Это объект **подключения**, объект **команды**, объект **DataReader** и объект **DataAdapter**. Объект подключения обеспечивает физическое подключение к источнику данных. Командный объект используется для выполнения оператора SQL или хранимой процедуры, которые должны

выполняться в источнике данных. Объект **DataReader** представляет собой поточное, прямое и только для чтения извлечение результатов запроса из источника данных, которые не обновляют данные. Наконец, объект **DataAdapter**, который заполняет объект набора данных результатами из источника данных.

**Набор данных**

**DataSet** обеспечивает автономное представление наборов результатов из источника данных и полностью независимо от источника данных



## Создание базы данных и добавление таблиц в Visual Studio

Visual Studio можно использовать для создания и обновления файла локальной базы данных в SQL Server Express LocalDB. можно также создать базу данных, выполнив инструкции Transact-SQL в окне инструментов **SQL Server обозреватель объектов** в Visual Studio.

### Предварительные требования

Для выполнения этого пошагового руководства вам потребуются рабочие нагрузки **.NET для настольных систем и хранения и обработки данных**, установленные в Visual Studio. чтобы установить их, откройте **Visual Studio Installer** и выберите пункт **изменить** (или **несколько > изменений**) рядом с версией Visual Studio, которую требуется изменить.

### Примечание

процедуры, рассмотренные в практической работе, применимы только к проектам платформа **.NET Framework Windows Forms**, а не к проектам **.net Core Windows Forms**.

### Ход работы:

#### Требования к содержанию отчета:

- Номер и название практической работы.
- Цель работы.
- По каждому заданию (задаче/примеру) экранные формы (при наличии) и листинг исходного программного кода и кода с изменением в процессе рефакторинга, показывающие порядок выполнения практической работы, и результаты, полученные в ходе её выполнения.
- Ответы на контрольные вопросы в тетради.

#### Порядок выполнения работы:

Все проекты практической работы размещать в своей сетевой в новой папке

**Пр24,25\_Фамилия.**

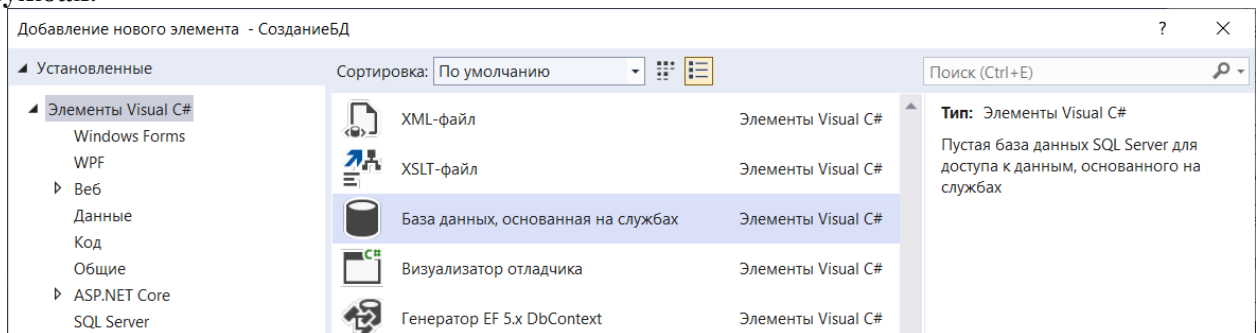
**В начале каждого файла проекта установить комментарии: пр.р.№\_\_\_\_\_ (указать номер), свою Фамилию. Формулировку задания**

**Задание 1. Создание базы данных и добавление таблиц в Visual Studio.** Для этого выполните следующие этапы:

#### 1 этап. Создание проекта и файла локальной базы данных

1. создайте новый проект **Windows Forms приложения** (платформа **.NET Framework**) и назовите его **Пр24,25\_Фамилия**.
2. в строке меню выберите **Проект - Добавить новый элемент... (компонент)**.
3. В списке шаблонов элементов прокрутите вниз и выберите **база данных, основанная на службах**.

**на службах.**

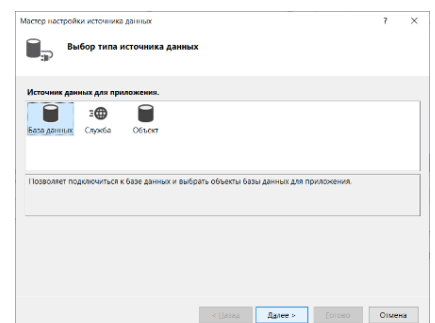


4. Присвойте базе данных имя **PK\_shop** и нажмите кнопку **Добавить**.

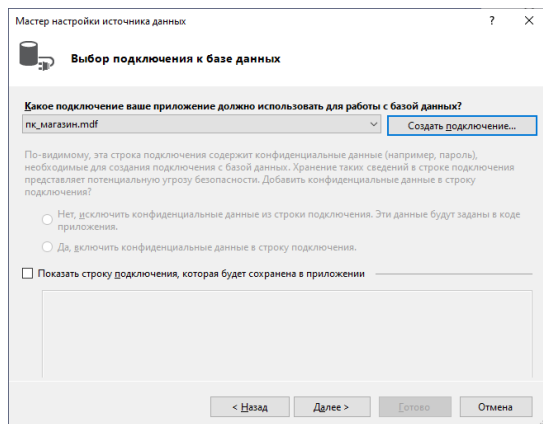
#### 2 этап. Добавление источника данных

1. если окно **Источники данных** не открыто, откройте его, нажав клавиши **Shift+Alt+D** или выбрав **Вид - Другие окна - Источники данных** в строке меню.
2. В окне **Источники данных** выберите **Добавить новый источник данных**.

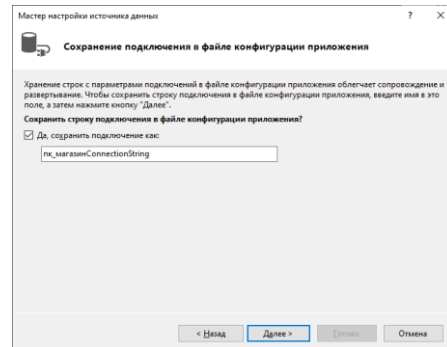
Откроется **Мастер настройки источника данных**.



- На странице **Выбор типа источника данных** выберите **база данных** , а затем нажмите кнопку **Далее**.
- На странице **Выбор модели базы данных** нажмите кнопку **Далее** , чтобы принять значение по умолчанию (**набор данных**).
- На странице **Выбор подключения к данным** выберите файл **PK\_shop.mdf** в раскрывающемся списке и нажмите кнопку **Далее**.



- На странице **сохранение подключения в файле конфигурации приложения** нажмите кнопку **Далее**.

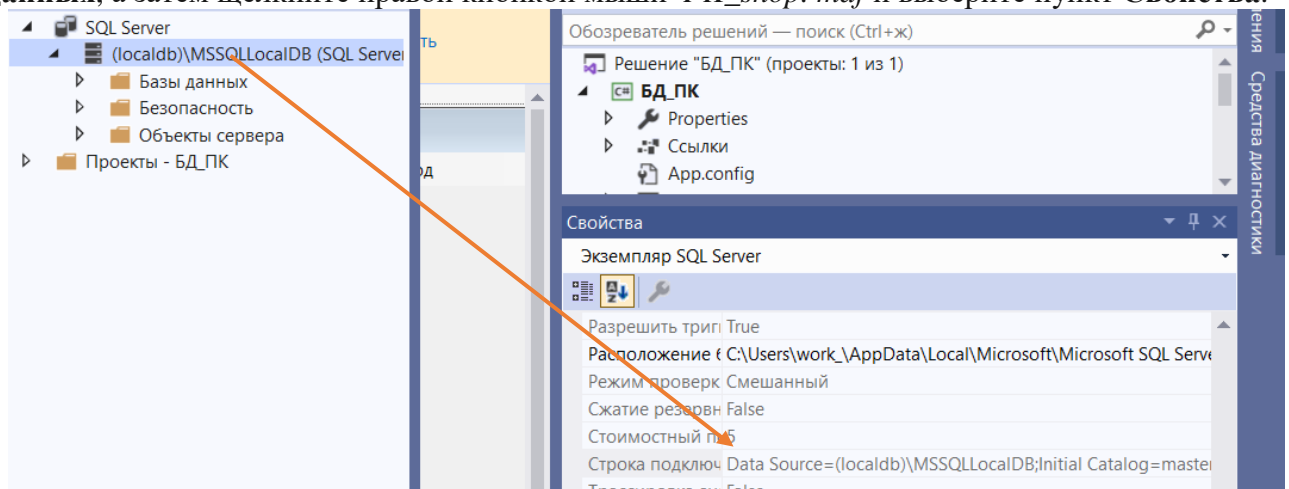


- На странице **Выбор объектов базы данных** появится сообщение о том, что база данных не содержит объектов. Нажмите кнопку **Готово**.

### 3 этап. Просмотр свойств подключения к данным

Знание строки подключения необходимо для реализации подключения и работы с базой данных на программном уровне. Чтобы просмотреть строку подключения для файла *PK\_shop.mdf*, откройте окно свойств подключения к данным:

- обозреватель объектов SQL Server** (чтобы открыть окно **SQL Server обозреватель объектов** меню **Вид**). Разверните узел **SQL Server - (LocalDB) \MSSQLLocalDB - базы данных**, а затем щелкните правой кнопкой мыши *PK\_shop.mdf* и выберите пункт **Свойства**.



### Примечание

если не удастся развернуть узел подключения к данным или отсутствует подключение *PK\_shop.mdf*, нажмите кнопку **Подключение к базе данных** на панели инструментов обозревателя сервера. в диалоговом окне **добавление соединения** убедитесь, что в поле **источник данных** выбран **Microsoft SQL Server файл базы данных**, а затем найдите и выберите файл *PK\_shop.mdf*. Завершите добавление подключения, нажав кнопку **ОК**.

### 4 этап. Создание таблиц и ключей с помощью конструктор таблиц

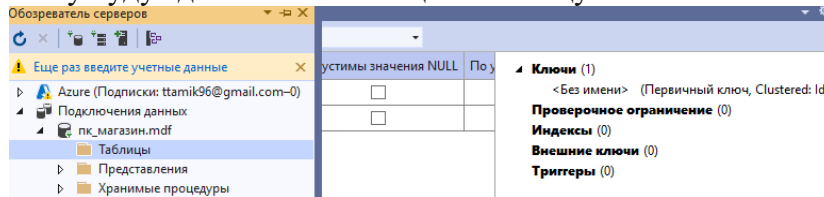
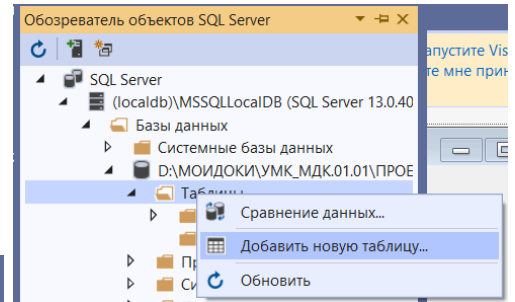
В этом разделе вы создадите две таблицы, первичный ключ в каждой таблице и несколько строк образца данных. Вы также создадите внешний ключ, чтобы указать, как записи в одной таблице соответствуют записям в другой таблице.

### Создание таблицы Products (Товары)

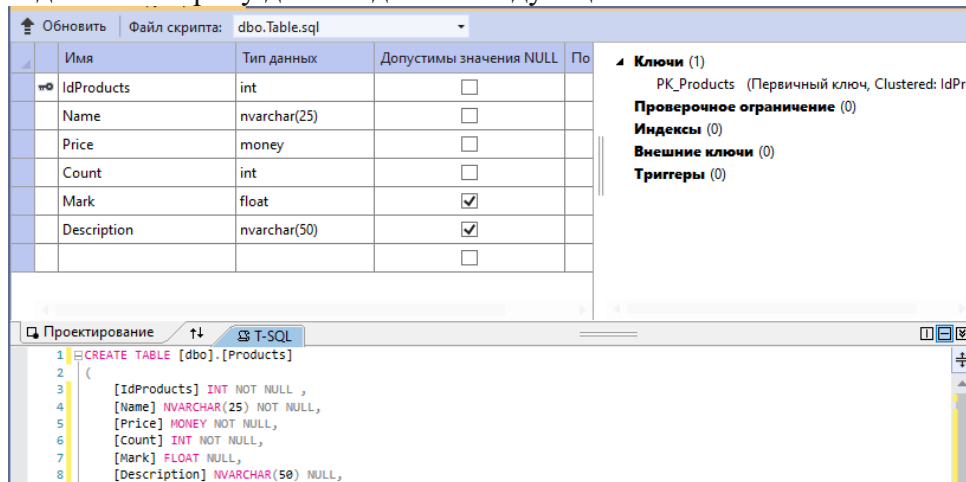
- В **Обозреватель сервера** разверните узел **SQL Server – Базы данных...**, и до узла *PK\_shop.mdf* .

2. Щелкните правой кнопкой мыши **таблицы** и выберите команду **Добавить новую таблицу...**

Будет открыт **Конструктор таблиц**, отобразится сетка с одной строкой по умолчанию, которая представляет один столбец в создаваемой таблице. Путем добавления строк в сетку будут добавлены столбцы в таблицу.



3. В сетке добавьте строку для каждой из следующих записей.



### Примечание

***nvarchar ( n / max )***

Строковые данные переменного размера. *n* определяет размер строки в парах байтов и может иметь значение от 1 до 4000. Значение **max** указывает, что максимальный размер при хранении составляет  $2^{30}-1$  символов (2 ГБ). Размер при хранении определяется как дважды *n* байт + 2 байта. В случае с кодировкой UCS-2 размер при хранении определяется как дважды *n* байт + 2 байта, а количество хранимых символов равно *n*. Для кодировки UTF-16 размер при хранении также равен дважды *n* байт + 2 байта, но количество хранимых символов может быть меньше *n*, так как дополнительные символы используют две пары байтов (также называются суррогатными парами). Синонимами типа **nvarchar** по стандарту ISO являются типы **national char varying** и **national character varying**.

4. Проверьте, чтобы поле **IdProducts** было установлено ключевым (правой кнопкой мыши и выберите пункт **Задать первичный ключ**).

### Примечание

строка по умолчанию (*Id*) должна быть удалена (правой кнопкой мыши и выберите пункт **Удалить**).

5. Назовите таблицу **Products** путем обновления первой строки в области скриптов, как показано в скриншоте окна.

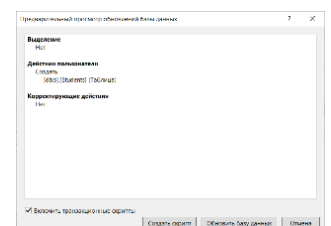
6. Добавьте ограничение индекса в таблицу **Products**. Добавьте запятую в конце [Description] строки, а затем добавьте следующий пример перед закрывающей круглой скобкой:  
**CONSTRAINT [PK\_Products] PRIMARY KEY CLUSTERED ([IdProducts] ASC)**

7. В левом верхнем углу **Конструктор таблиц** выберите **Обновить** или нажмите клавиши **SHIFT + ALT + U**.

8. В диалоговом окне **Предварительный просмотр обновлений базы данных** выберите **обновить базу данных**.

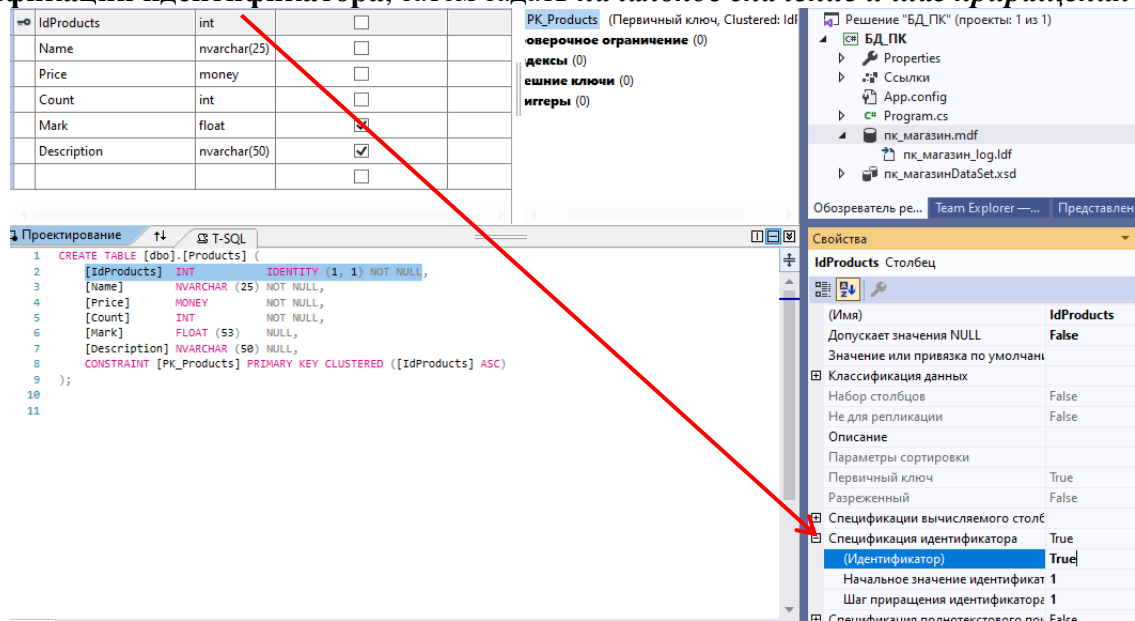
Таблица **Products** создается в файле локальной базы данных.

**Создание поля автоинкремента (счетчика) в таблице базы данных MS SQL Server**





Для того чтобы поле **Идентификатор продукта** имел автоматическую нумерацию (аналогии типа **Счетчик** в СУБД Access), необходимо активировать поле **IdProducts**, тип поля **int** и в окне свойств «**Properties**» установить значение **True** для свойства **Идентификатор** в Спецификации идентификатора, затем задать *начальное значение и шаг приращения*



### Создание таблицы Users (Покупатели)

1. Аналогично создайте еще одну таблицу, а затем добавьте строку для каждой записи следующей таблицы:
2. Задайте **IdUser** в качестве первичного ключа, а затем удалите строку по умолчанию.
3. Назовите таблицу **Users** путем обновления первой строки в области скриптов
4. Добавьте ограничение индекса в таблицу **Users** для ключа **IdUser**
5. Выполните **обновление** базы данных.

Если развернуть узел **таблицы** в обозревателе сервера, отобразятся две таблицы:

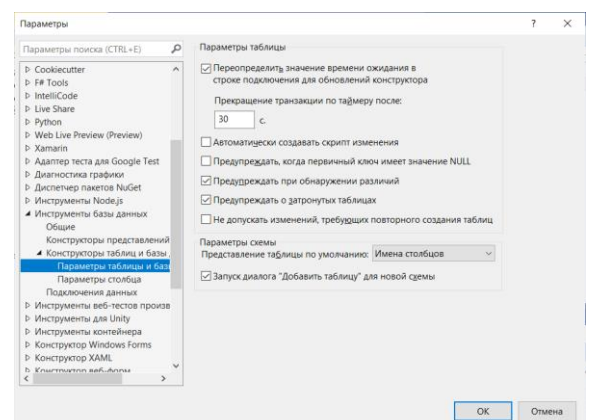
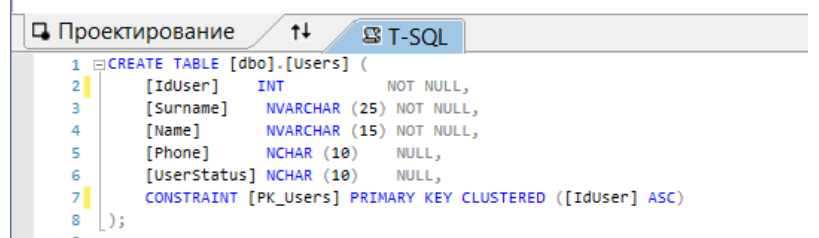
Если вы не видите его, нажмите кнопку **Обновить** на панели инструментов.

### Редактирование структуры таблиц.

Бывают случаи, когда нужно изменить структуру таблицы базы данных.

Для того, чтобы вносить изменения в таблицы базы данных в MS Visual Studio, сначала нужно снять опцию **“Не допускать изменений, требующих повторного создания таблицы”** как показано на рисунке. Иначе, MS Visual Studio будет блокировать внесения изменений в ранее созданную таблицу. Окно **Параметры**, вызывается из меню **Средства** в такой последовательности: **Средства** - **Параметры**

Имя	Тип данных	Допустимы значения NULL
IdUser	int	<input type="checkbox"/>
Surname	nvarchar(25)	<input type="checkbox"/>
Name	nvarchar(15)	<input type="checkbox"/>
Phone	nchar(10)	<input checked="" type="checkbox"/>
UserStatus	nchar(10)	<input checked="" type="checkbox"/>



## - Инструменты базы данных - Параметры таблицы и базы данных

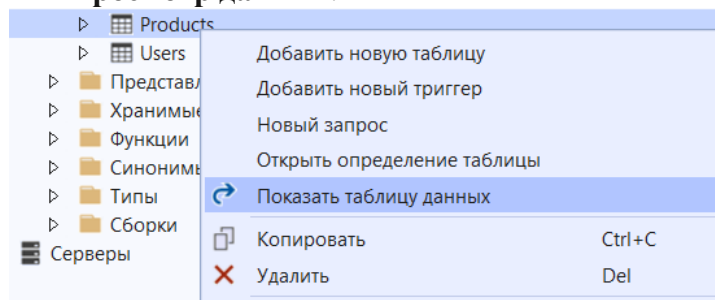
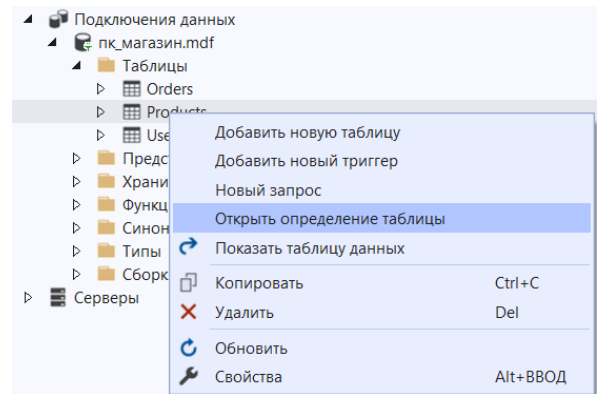
После настройки можно изменять структуру таблицы. Для этого используется команда «Открыть определение таблицы» из контекстного меню окна **Обозреватель серверов** или через **SQL Server – Базы данных... - Показать конструктор**, которая вызывается для выбранной таблицы (правый клик мышкой).

### 5 этап. Заполнение таблиц данными

1. в обозреватель сервера или **SQL Server обозреватель объектов** разверните узел образца базы данных.

2. Откройте контекстное меню для узла **таблицы**, выберите **Обновить**, а затем разверните узел **таблицы**.

3. Откройте контекстное меню таблицы **Products** и выберите **Показать данные таблицы** или **Просмотр данных**.



4. Добавьте необходимые данные для 7-8 товаров, например

IdProducts	Name	Price	Count	Mark	Description
4	процессор Inte...	45000,0000	150	4,3	10400F
5	ОЗУ 8GB	26000,0000	789	5	Adata XPG 3200Mhz
6	ОЗУ 4GB	18000,0000	100	4	Adata XPG 2400Mhz
7	Видеокарта	10000,0000	478	4,3	GIGABYTE RX 6500 XT

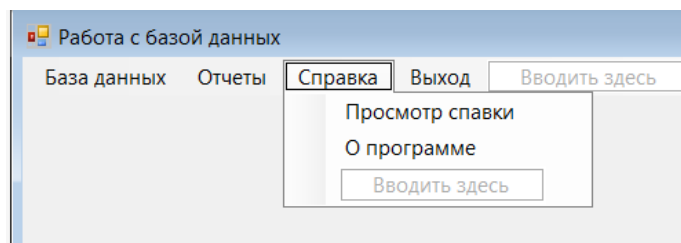
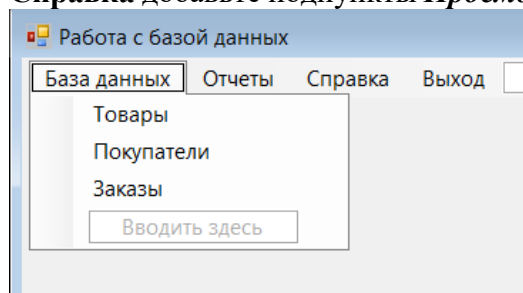
5. Аналогично заполните вторую таблицу покупателей.

**Важно!**

Убедитесь, что все идентификатор таблицы Покупатели и количества товаров — целые числа.

### Задание 2. Создание интерфейса приложения

Отобразите главную форму в рабочей области среды разработки. Создайте меню. В пункт **Справка** добавьте подпункты **Просмотр справки**, **О программе**.



### Задание 3. Доступ к данным в Visual Studio

#### Создание формы Покупатель

- Добавьте новую форму в проект
- Для того чтобы поместить на новую форму поля таблицы их необходимо перетащить из панели «Источники данных» на форму. Из таблицы «Users» перетащите мышью на форму поля «Фамилия», «Имя», «Телефон» и «Статус». Форма примет вид:

Обратите внимание, что после перетаскивания полей с панели «Источники данных» на форму в верхней части формы появилась навигационная панель, а в нижней части рабочей области среды разработки появились пять невидимых объектов. Эти объекты предназначены для связи нашей формы с таблицей «Покупатели», расположенной на сервере. Рассмотрим функции этих объектов:

- **имяБДDataSet (Набор данных)** – обеспечивает подключение формы к конкретной БД на сервере (в нашем случае это БД pk\_shop);

- **UsersBindingSource (Источник связи для таблицы «Users»)** – обеспечивает подключение к конкретной таблице (в нашем случае к таблице покупателей), а также позволяет управлять таблицей;

- **UsersTableAdapter (Адаптер таблиц для таблицы «Users»)** – обеспечивает передачу данных с формы в таблицу и наоборот.

- **TableAdapterManager (Менеджер адаптера таблиц)** – управляет работой объекта USERSTableAdapter;

- **UsersBindingNavigator (Панель управления таблицей «Users»)** – голубая панель с кнопками управления таблицей, расположенная в верхней части формы.

- Измените названия полей на: «Фамилия», «Имя», «Телефон» и «Статус».

Мы не помещаем поле «Код пользователя» на нашу форму, так как данное поле является первичным полем связи и заполняется автоматически. Конечный пользователь не должен видеть такие поля.

Теперь нам необходимо проверить работоспособность новой формы. Для отображения формы «Покупатели» её необходимо подключить к главной форме, а затем запустить проект и открыть форму «Покупатели».

Для этого реализуйте на главной форме интерфейс приложения.

### Подключение формы «Покупатели» к главной форме

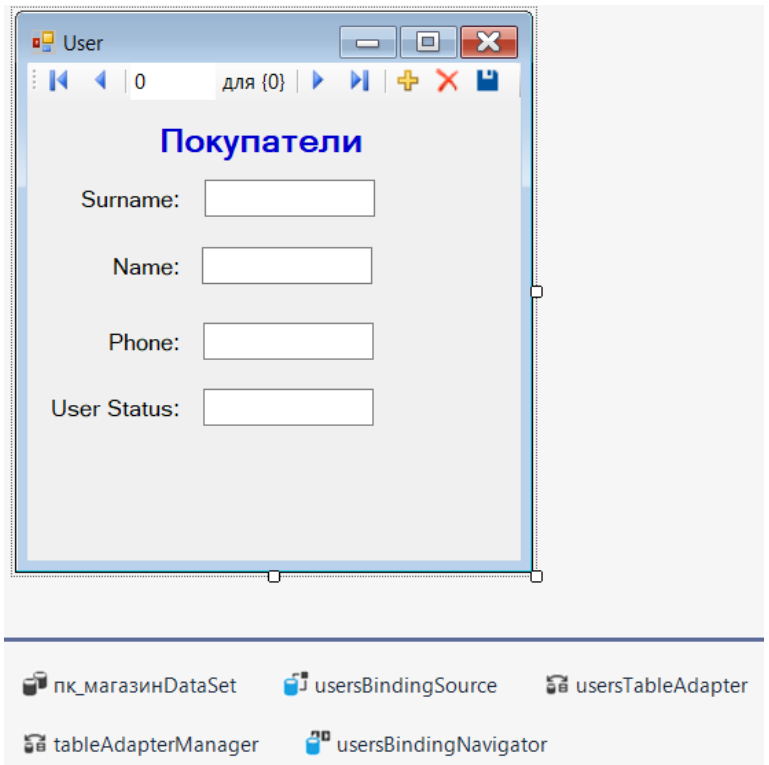
- Для этого дважды щёлкните ЛКМ по соответствующему элементу меню, расположенному на главной форме.
- В появившемся окне кода формы в процедуре **ПокупателиToolStripMenuItem\_Click** наберите код, предназначенный для открытия формы «Таблица «Покупатели»» (Users):

```
private User users;
```

```
ссылка: 1
public Titul()
{
    InitializeComponent();
}

ссылка: 1
private void ПокупателиToolStripMenuItem_Click(object sender, EventArgs e)
{
    users = new User();
    users.Visible = true;
}
```

- Проверьте работу проекта. Для открытия формы, отображающей форму «Покупатели» на

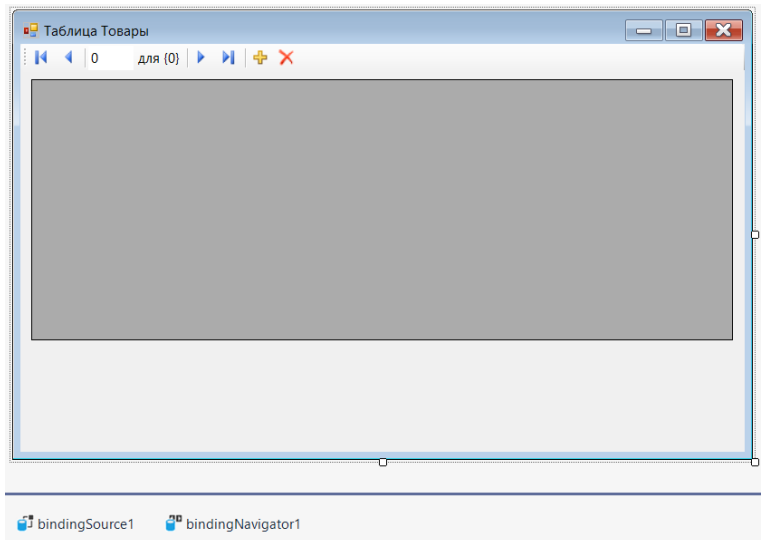
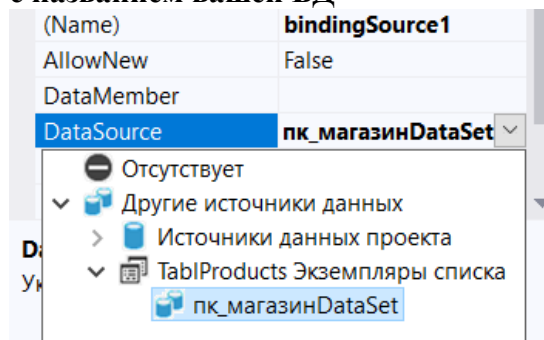


главной форме выберите меню Покупатели.

### Создание табличной формы Товары

- Добавьте вторую форму **Товары**
- На форму из вкладки «Данные» панели элементов добавить компоненты **BindingSource**, **DataGridView**, **BindingNavigator** (два последних являются визуальными компонентами).
- Настроить свойства компонента **bindingSource1**

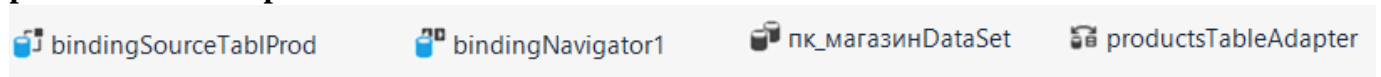
1) в элементе **bindingSource1** выбрать в свойстве **DataSource** именно тот **Dataset**, который расположен на одной форме с этим компонентом **Другие источники данных** – **Источники данных** – развернуть таблицу и выбрать **Dataset** с названием вашей БД



Измените name **bindingSource1** на **bindingSourceTablProd**

2) в свойстве **DataMember** нужно выбрать имя таблицы, связанной с этим компонентом – **Products**.

Если всё сделано, верно, то в списке невидимых компонентов должен появиться **TableAdapter** – **productsTableAdapter**.



3) в редакторе кода формы найдите строку, которая загружает данные в адаптер таблиц. Этот код был сгенерирован при установке привязки данных. Код должен иметь следующий вид:

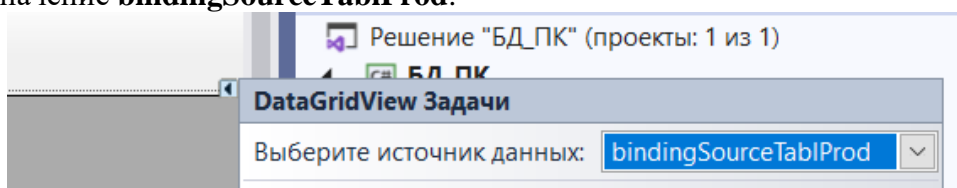
**!!! Обратите внимание, что название БД должно соответствовать вашему названию файла**

```
private void TablProducts_Load(object sender, EventArgs e)
{
    // TODO: данная строка кода позволяет загрузить данные в таблицу "пк_магазинDataSet.Products".
    // При необходимости она может быть перемещена или удалена.
    this.productsTableAdapter.Fill(this.пк_магазинDataSet.Products);
}
```

Он находится в событии формы **Form1\_Load**.

4) Настроить свойства компонента **dataGridView1**

В **dataGridView1** нужно изменить name **dataGridViewProduct**, свойству **DataSource** установить значение **bindingSourceTablProd**.



Появятся заголовки столбцов таблицы, как они заданы в базе данных.

5) Настроить свойства компонента **bindingNavigator1**

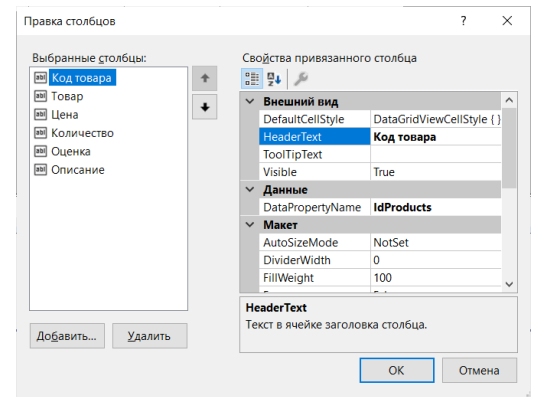
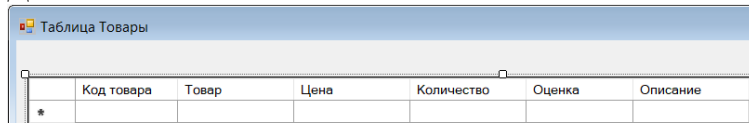


Установить свойству **BindingSource** соответствующий компонент формы - **bindingSourceTablProd**

- Изменение названий столбцов таблицы

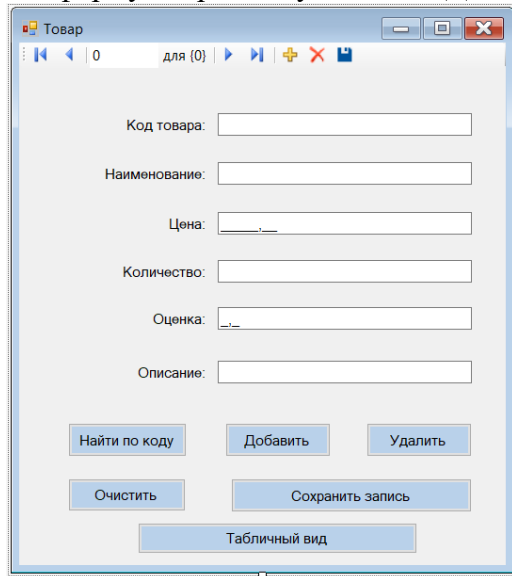
В **dataGridViewProduct** разверните задачи и выберите **Правка столбцов...**

Для каждого поля изменить свойство **HeaderText**



#### Задание 4. Создание сложных ленточных форм для работы с данными

- Добавьте новую ленточную форму для таблицы «Товары». Аналогично созданию формы *Покупатели* перенесите поля таблицы **Products** на форму. Переименуйте поля. Добавьте необходимые кнопки:



в редакторе кода формы будет сгенерирован код при установке привязки данных:

```
public partial class Product : Form
{
    ссылка: 1
    public Product()
    {
        InitializeComponent();
    }

    ссылка: 1
    private void productsBindingNavigatorSaveItem_Click(object sender, EventArgs e)
    {
        this.Validate();
        this.productsBindingSource.EndEdit();
        this.tableAdapterManager.UpdateAll(this.пк_магазинDataSet);
    }

    ссылка: 1
    private void Product_Load(object sender, EventArgs e)
    {
        // TODO: данная строка кода позволяет загрузить данные в таблицу "пк_магазинDataSet.Products".
        // При необходимости она может быть перемещена или удалена.
        this.productsTableAdapter.Fill(this.пк_магазинDataSet.Products);
    }
}
```

- Для каждой кнопки дайте name соответствующее ее назначению (см. далее образцы программного кода событий кнопок)
- События нажатия кнопок **Добавить** и **Удалить**

```
private void buttonAddProd_Click(object sender, EventArgs e)
{
    productsBindingSource.AddNew();
}
```

```
ссылка: 1
private void buttonDelProd_Click(object sender, EventArgs e)
{
    productsBindingSource.RemoveCurrent();
}
```

- Событие кнопки **Сохранить**

```
private void buttonSaveProd_Click(object sender, EventArgs e)
{
    //проверяет введенные в поля данные на соответствие типам данных полей
    this.Validate();
    //закрывает подключение с сервером
    this.productsBindingSource.EndEdit();
    //обновляет данные на сервере
    this.tableAdapterManager.UpdateAll(this.пк_магазинDataSet);
}
```

- Событие кнопки **Найти по коду**

Подключить директиву:

```
using System.Data.SqlClient;
```

и далее

```
private void buttonPoiskProd_Click(object sender, EventArgs e)
{
    string sql = String.Concat("SELECT * FROM Products WHERE IdProducts=", idProductsTextBox.Text);

    // подключение к серверу Sql
    string connectionString;
    connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\nk_магазин.mdf;Integrated Security=True";

    //создать объект подключения и открыть подключение к источнику данных
    SqlConnection connection = new SqlConnection(connectionString);
    connection.Open();

    //назначение открытого соединения свойству соединения объекта Command
    SqlCommand command = new SqlCommand(sql, connection);
    SqlDataReader dataReader = command.ExecuteReader(); //поточное извлечение результатов запроса только вперед и только для чтения из источника данных

    // Очистка от старых записей
    //Вывод найденной записи
    nameTextBox.Text = "";
    pricemaskedTextBox.Text = "";
    countTextBox.Text = "";
    markmaskedTextBox.Text = "";
    descriptionTextBox.Text = "";

    while (dataReader.Read())
    {
        nameTextBox.Text = nameTextBox.Text + dataReader["Name"];
        pricemaskedTextBox.Text = pricemaskedTextBox.Text + dataReader["Price"];
        countTextBox.Text = countTextBox.Text + dataReader["Count"];
        markmaskedTextBox.Text = markmaskedTextBox.Text + dataReader["Mark"];
        descriptionTextBox.Text = descriptionTextBox.Text + dataReader["Description"];
    }
    // После завершения работы с базой данных соединение должно быть закрыто
    // и освобождены ресурсы источника данных
    dataReader.Close();
    connection.Close();
}
```

- Кнопку **Очистить** запрограммировать самостоятельно
- Событие кнопки **Табличный вид** - отображение формы 2:

```
private TablProducts TablProduct;
```

ссылка: 1

```
private void button2_Click(object sender, EventArgs e)
{
    TablProduct = new TablProducts();
    TablProduct.Visible = true;
}
```

- Для проверки работы созданных кнопок запустите проект откройте форму Товары и нажмите каждую из кнопок.
- Далее измените объекты, отображающие поля для более удобного ввода информации. Для начала удалите текстовые поля ввода (TextBox), отображающие следующие поля таблицы «Products»: «Цена», «Оценка».
- Для отображения полей «Цена» и «Оценка» будем использовать текстовые поля ввода по маске (**MaskedTextBox**). Объект текстовое поле ввода по маске отсутствует в выпадающем списке объектов для отображения полей в окне «Источники данных», поэтому необходимо создавать данные объекты при помощи панели объектов (Toolbox), а затем подключать их к соответствующим полям вручную. Создайте текстовые поля ввода по маске

 **MaskedTextBox** справа от надписей «Цена» и «Оценка».

- Настройте маски ввода: полю «**Цена**» выберите пункт «**Установка маски...**»



**Примечание:** Символы для создания масок ввода приведены в приложении таблица 1.


В окне задания маски «**Маска ввода**» выберите маску «**Специальный**» и введите маску позволяющую пользователю вводить до 5 цифр до запятой и 2 цифры после запятой(причем один знак должен быть введен обязательно) нажмите кнопку «**ОК**».

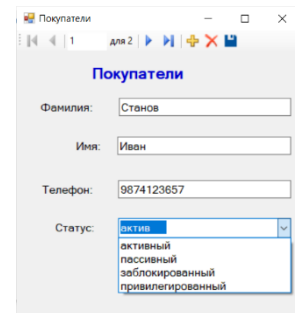
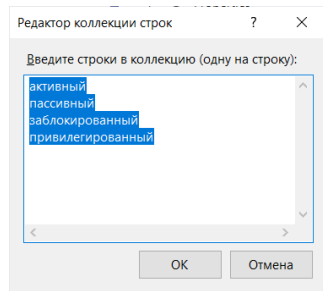
Для поля **Оценка** введите маску «**0.0**».

- Теперь необходимо подключить созданные текстовые поля ввода по маске к соответствующим полям. Для этого с панели «Источники данных» (DataSources) перетащите поле «Price» на текстовое поле ввода по маске, расположенное справа от надписи «Цена». Прделайте такую же операцию с полем «Оценка».

#### Задание 5. Заполнение параметров поля «Выпадающий список»

Поле «Статус» в таблице «Покупатели» должно иметь выпадающий список с predetermined значениями.

- Для этого удалите поле ввода справа от строки «Статус» на форме «Покупатели».
- Добавьте поле типа выпадающий список  **ComboBox** и свяжите с полем БД, как Вы это делали с полями «Цена» и «Оценка».
- В свойстве поля **Items** укажите значения, которые будут содержаться в выпадающем списке:



- Создайте маску для ввода телефона. Не забудьте поле маски связать с полем БД.

#### Задание 6. Работа с записями

Перейдём теперь к формированию сложных табличных форм для работы с записями в табличной форме данных. Рассмотрим приемы фильтрации и сортировки данных, а также реализуется поиск информации в таблице.

- 1) Откройте таблицу «Товары» в режиме конструктора и отредактируйте ее внешний вид, т.е. поместите на неё следующие объекты (табличную часть **не удалять** и ставить без изменения):

- четыре надписи (Label);
- пять кнопок (Button);
- выпадающий список (ComboBox);
- текстовое поле ввода (TextBox);
- группирующую рамку (GroupBox);
- список (ListBox);
- два переключателя (RadioButton).

Расположение элементов:

- 2) Выполните настройку свойств объектов:

- Задайте **свойства формы** следующим образом:

- **FormBorderStyle** (Стиль границы формы): **Fixed3D**;
- **MaximizeBox** (Кнопка развёртывания формы во весь экран): **False**;
- **MinimizeBox** (Кнопка свёртывания формы на панель задач): **False**;
- **Text** (Текст надписи в заголовке формы): **Таблица Товары**.

- Задайте **свойства надписей** (Label1, Label2, Label3 и Label4) как (рисунок 48):

- **AutoSize** (Авторамер): **False**;
- **Text** (Текст надписи): «Товары (Табличный вид)», «Поле для сортировки», «Наименование» и «Критерий».

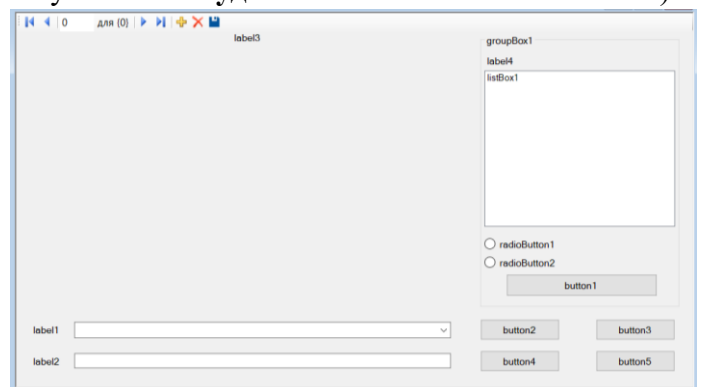
«Наименование» и «Критерий».

- Задайте **надписи на кнопках** как: «Сортировать», «Фильтровать», «Показать все», «Найти» и «Закрыть».

- Для того чтобы нельзя было произвести сортировку не выбрав поля изначально заблокируем кнопку «Сортировать» (свойство **Enabled = false**).

- У группирующей рамки задайте заголовок «Сортировка».

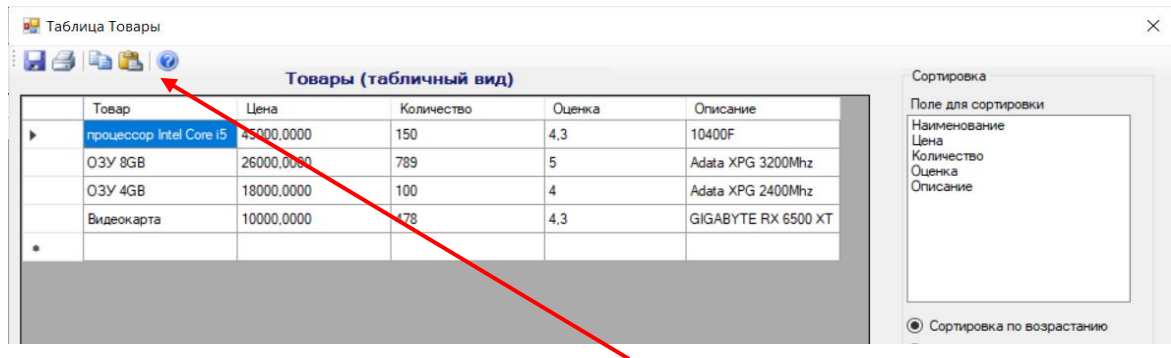
- У переключателей (Объекты **RadioButton1** и **RadioButton2**) задайте надписи как



«Сортировка по возрастанию» и «Сортировка по убыванию», а у переключателя «Сортировка по возрастанию» (**RadioButton1**) задайте свойство **Checked** (Включён) равное **True** (Истина).

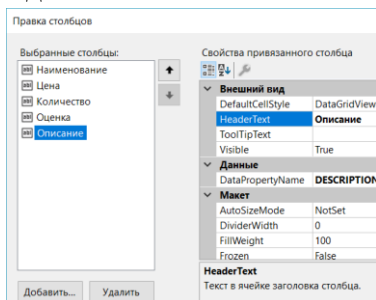
➤ Заполните список (**ListBox1**) значениями:

- Наименование
- Цена
- Количество
- Оценка
- Описание



3) Также на форму добавьте второй элемент **bindingNavigator2** - панель навигации, в которую добавьте стандартные элементы и удалите не нужные кнопки.

4) При редактировании табличной части **dataGridViewProduct** в окне «Правка столбцов...» из списка полей удалите поле «**IdProduct**» (Код товара), выделив его и нажав кнопку **Удалить**.

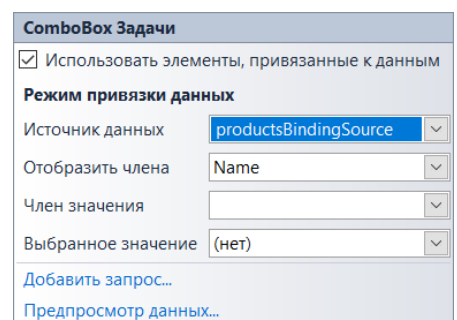
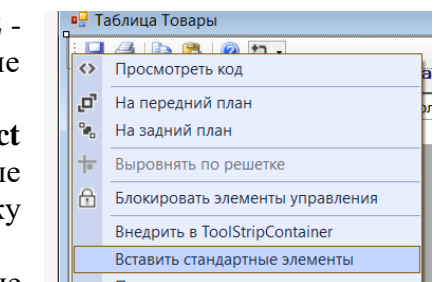


5) Настроим заполнение выпадающего списка наименованием товаров из таблицы **Товары**. Отобразите меню действий выпадающего списка.

Включите опцию «Использовать элементы, привязанные к данным». В строке **Источник данных** выберите **Другие источники данных**, **Источники данных**

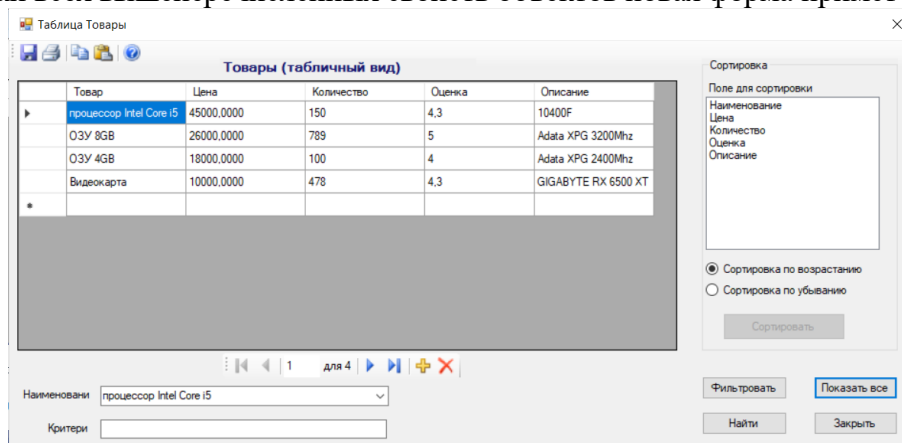
проекта,

**ProductBindingSource**. В строке **Отобразить члена** выберите **Name**.



6) На панели невидимых объектов появится дополнительный объект связи «**ProductBindingSource**», предназначенный для заполнения выпадающего списка.

После настройки всех вышеперечисленных свойств объектов новая форма примет вид:



7) Написание кода обработчиков событий объектов:

➤ **Код для разблокирования кнопки «Сортировать»**, при выборе пункта списка (ListBox1).

Для создания процедуры события дважды щёлкните ЛКМ по списку. Появится процедура обработки события, происходящего при выборе пункта списка (**ListBox1\_SelectedIndexChanged**).

```
//разблокировки кнопки «Сортировать»
```

ссылка: 1

```
private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    buttonSort.Enabled = true;
}
```

➤ **Код сортирующего нашу таблицу** в зависимости от выбранного поля и порядка сортировки при нажатии кнопки «Сортировать».

Дважды щёлкните ЛКМ по кнопке «Сортировать». В процедуре наберите код:

**!!! В коде наименования столбцов должны соответствовать наименованиям dataGridViewProduct**

```
private System.Windows.Forms.DataGridColumn COL;
ссылка: 1
private void buttonSort_Click(object sender, EventArgs e)
{
    //создает переменную COL для хранения имени выбранного столбца таблицы
    COL = new System.Windows.Forms.DataGridColumn();

    //блок switch, присваивающий в переменную Col имя выбранного столбца таблицы
    //в зависимости от номера выбранного пункта списка (ListBox1.SelectedIndex).
    // Если выбран первый пункт списка, то в переменную Col записывается столбец
    //DataGridViewTextBoxColumn2, если второй, то - DataGridViewTextBoxColumn3
    //и так далее. Хотелось бы отметить тот факт, что нумерация пунктов списка
    //начинается с нуля, а нумерация столбцов с единицы. Первый столбец «Наименование»
    //носит имя DataGridViewTextBoxColumn2, так как имя
    //DataGridViewTextBoxColumn1 имеет столбец заголовков строк;

    switch (listBox1.SelectedIndex)
    {
        case 0:
            COL = nameDataGridViewTextBoxColumn;
            break;
        case 1:
            COL = priceDataGridViewTextBoxColumn;
            break;
        case 2:
            COL = countDataGridViewTextBoxColumn;
            break;
        case 3:
            COL = markDataGridViewTextBoxColumn;
            break;
        case 4:
            COL = descriptionDataGridViewTextBoxColumn;
            break;
    }

    //Блок If выполняет следующую операцию: если включён
    //переключатель «Сортировка по возрастанию» (RadioButton1), то отсортировать
    //таблицу по полю заданному в переменной Col по возрастанию
    //(PRODUCTSDataGridView.Sort (Col, System.ComponentModel.ListSortDirection.
    //Ascending)), иначе по убыванию (PRODUCTSDataGridView.Sort (Col, System.
    //ComponentModel.ListSortDirection. Descending)).
    if (radioButton1.Checked)
        dataGridViewProduct.Sort(COL,
            System.ComponentModel.ListSortDirection.Ascending);
    else
        dataGridViewProduct.Sort(COL,
            System.ComponentModel.ListSortDirection.Descending);
}
```

➤ **Код обработчика события нажатия кнопки «Фильтровать».**

Дважды щёлкните по кнопке «Фильтровать» и наберите код

```
private void buttonFiltr_Click(object sender, EventArgs e)
{
    bindingSourceTablProd.Filter = "Name='" + comboBox1.Text + "'";
}
```

У объекта **bindingSourceTablProd** имеется текстовое свойство **Filter**, которое определяет условие фильтрации. Условие фильтрации имеет синтаксис:

**“<Имя поля><Оператор>’<Значение>”**

В нашем случае значение поля «**Name**» приравняется к значению, выбранному в выпадающем списке (**ComboBox1.Text**).



- **Код обработчика события кнопки «Показать всё»**, отменяющей фильтрацию записей. Дважды щёлкните по указанной кнопке, наберите команду:

```
private void buttonViewAll_Click(object sender, EventArgs e)
{
    bindingSourceTblProd.Filter = "";
}
```

Заметим, что если присвоить свойству **«Filter»** значение пустой строки (“”), то его действие будет отменено.

- **Реализация поиска информации в таблице.**

Дважды щёлкните по кнопке «Найти». И наберите код:

```
private void buttonPoisk_Click(object sender, EventArgs e)
{
    //перебирает все ячейки таблицы и устанавливает в них белый цвет фона
    // и чёрный цвет текста, то есть отменяет результаты предыдущего поиска
    for (int i = 0; i < dataGridViewProduct.ColumnCount - 1; i++)
    {
        for (int j = 0; j < dataGridViewProduct.RowCount - 1; j++)
        {
            dataGridViewProduct[i, j].Style.BackColor = Color.White;
            dataGridViewProduct[i, j].Style.ForeColor = Color.Black;
        }
    }
    //перебирает все ячейки таблицы и если они содержат текст, введённый в поле ввода (TextBox1),
    // то устанавливает в них голубой цвет фона и синий цвет текста, чем выделяет искомые ячейки.
    for (int i = 0; i < dataGridViewProduct.ColumnCount - 1; i++)
    {
        for (int j = 0; j < dataGridViewProduct.RowCount - 1; j++)
        {
            if (dataGridViewProduct[i, j].Value.ToString().IndexOf(textBox1.Text) != -1)
            {
                dataGridViewProduct[i, j].Style.BackColor = Color.AliceBlue;
                dataGridViewProduct[i, j].Style.ForeColor = Color.Blue;
            }
        }
    }
}
```

- **Код для кнопки «Закрыть».**

Дважды щёлкните ЛКМ по этой кнопке и в появившейся процедуре наберите код:

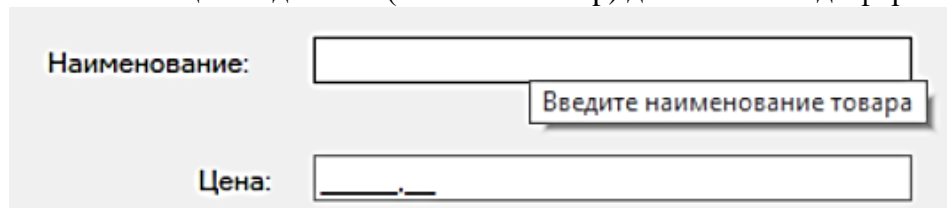
```
private void buttonClose_Click(object sender, EventArgs e)
{
    this.Close();
}
```

Проверьте работоспособность табличной формы. Проверьте, как работает поиск, фильтрация и сортировка записей в таблице, нажимая на соответствующие кнопки. После проверки работы формы для возвращения в среду разработки просто закройте все формы.

**Примечание:** Для проверки сохранения данных в таблицы базы данных, проект необходимо запустить как приложение из папки *Debug*. Внести новые записи, сохранить. И повторно перезапустить как приложение и посмотреть содержимое таблиц.

## Задание 7. Выполните настройку интерфейса пользователя:

1. Установите всплывающие подсказки (элемент *toolTip*) для полей ввода форм. Например,



Наименование:

Цена:

```

ToolTip tip = new ToolTip();

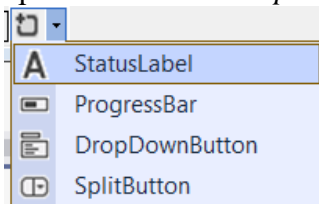
tip.SetToolTip(nameTextBox, "Введите наименование товара");
tip.SetToolTip(descriptionTextBox, "Введите пояснение к товару");
tip.SetToolTip(markmaskedTextBox, "укажите оценку товара по 5-тибалльной шкале");

```

- Создайте подсказку для пользователя в строке состояния для полей Наименование, Критерий, Поле для сортировки в табличной форме Товары

Пример создания строки состояния для поля **Наименование**:

- Перетащите элемент управления **StatusStrip** из панели элементов в табличную форму. Элемент управления **StatusStrip** автоматически располагается в нижней части формы.
- Нажмите кнопку раскрывающегося списка для элемента управления **StatusStrip** и выберите **StatusLabel** для добавления элемента управления **ToolStripStatusLabel** в элемент управления **StatusStrip**.



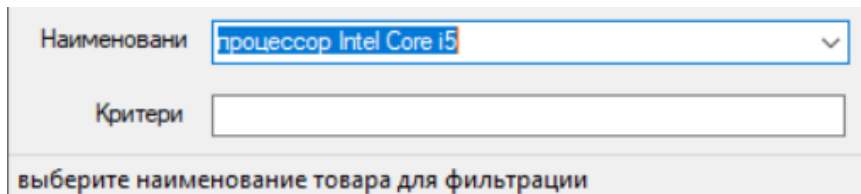
- Обработайте событие **MouseEnter**, чтобы оно реагировало на выделение пользователем поля Наименование. Для этого:
- Выделите поле, в окне «Свойства» щелкните «События», дважды щелкните событие **MouseEnter**. Напишите код в обработчик события:

```

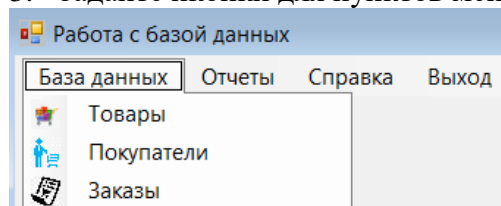
private void comboBox1_MouseEnter(object sender, EventArgs e)
{
    toolStripStatusLabel1.Text = "выберите наименование товара для фильтрации";
}

```

- Проверьте результат. Получим появление подсказки в строке состояния при наведении мыши на поле:

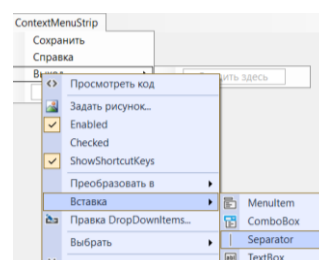
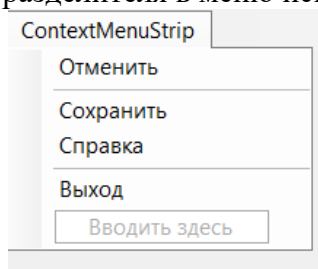


- Аналогично реализуйте остальные подсказки.
- Задайте иконки для пунктов меню **База данных** и **Справка**, например:

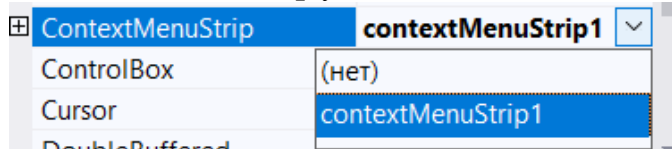


- Создайте контекстное меню для главной формы.

- Для этого перетащите элемент управления **ContextMenuStrip** из панели элементов на главную форму и создайте пункты меню (аналогично главным пунктам меню). Для добавления разделителя в меню используйте элемент **ToolStripSeparator**. Например:



- Далее необходимо контекстное меню подключить к форме: выделить форму, свойство **ContextMenuStrip** установить значение contextMenuStrip1



- Привязка пунктов контекстного меню к конкретным функциям осуществляется путем создания кода обработчика событий для каждого пункта меню. Для формирования обработчика необходимо выделить на форме класс **ContextMenuStrip** и сделать двойной щелчок на соответствующем пункте меню, например "Выход". В сгенерированном обработчике необходимо добавить вызов метода, для функции "Выход" - метод Close().

```
private void выходToolStripMenuItem1_Click(object sender, EventArgs e)
{
    this.Close();
}
```

## 5. Использование справочного файла

**Примечание:** Разделы справочного руководства создаются как отдельные HTML - файлы и могут быть созданы в любом редакторе, позволяющем работать с таким форматом. Справочные руководства часто хранятся в откомпилированном файле с уточнением ".chm".

- Добавьте элемент управления **HelpProvider** на форму.
- В папке с решением создайте файл справки, например, документ Microsoft Word. Текст укажите произвольный и в конце файла свою группу и ФИО.
- Для элемента **helpProvider1** в свойстве **HelpNamespace** укажите путь к файлу справки.
- Реализуйте возможность вызова файла справки из соответствующего пункт меню.
- Создайте обработчик события выбора файла справки. В теле обработчика укажите следующую строку:

```
Help.ShowHelp(this, helpProvider1.HelpNamespace);
```

- Проверьте результат. Проверьте, что открылся требуемый файл.
6. Установите запрет на ввод недопустимых символов в поля: в таблице Покупатели - фамилия, имя; в таблице Товары - количество.
  7. При необходимости запрограммируйте остальные необходимые элементы приложения

Класс **MaskedTextBox** является усовершенствованной версией элемента управления **TextBox**, которая поддерживает декларативный синтаксис, на основе которого принимаются или отклоняются данные, вводимые пользователем. С помощью свойства **Mask** можно задать следующие типы входных данных, не создавая в приложении специальный код для проверки.

**Таблица 1 – Символы для создания масок ввода**

Символ маски	Описание
0	Цифра, ввод обязателен. Пользователь вправе ввести любую цифру из диапазона 0-9
9	Цифра или пробел. Ввод необязателен.
#	Цифра, пробел, плюс или минус. Ввод необязателен.
L	Буква, ввод обязателен.
?	Буква, ввод необязателен.
&	Символ (в т.ч. Unicode), ввод обязателен. Любой не управляющий символ. При значении true свойства <b>AsciiOnly</b> данный символ маски совершенно аналогичен символу “L”
C	Символ (в т.ч. Unicode), ввод не обязателен. Любой не управляющий символ. При значении true свойства <b>AsciiOnly</b> данный символ маски совершенно аналогичен символу “?”
a	Буквенно-цифровой символ (Unicode не принимается), ввод обязателен. При значении true свойства <b>AsciiOnly</b> данный символ маски позволяет ввод лишь ASCII букв a-z и A-Z.
A	Буквенно-цифровой символ (Unicode не принимается), ввод опциональный. При свойстве <b>AsciiOnly</b> приравненным к true данный символ маски позволяет ввод лишь ASCII букв a-z и A-Z.
. (точка)	Место, где будет отображен десятичный разделитель. Какой символ будет использоваться, определяется свойством <b>Culture control-a</b> .
, (запятая)	Место, где будет отображен разделитель тысяч. Какой символ будет использоваться, определяется свойством <b>Culture control-a</b> .
: (двоеточие)	Место, где будет отображен разделитель времени. Какой символ будет использоваться, определяется свойством <b>Culture control-a</b> .
/ (прямой слеш)	Место, где будет отображен разделитель даты. Какой символ будет использоваться, определяется свойством <b>Culture control-a</b> .
\$	Место, где будет отображен символ валюты. Какой символ будет использоваться, определяется свойством <b>Culture control-a</b> .
<	Весь ввод пользователя после этого символа маски автоматически приводится к нижнему регистру.
>	Весь ввод пользователя после этого символа маски автоматически приводится к верхнему регистру.
	С этого места отменить действие любого из двух предыдущих символов маски. Принимать ввод пользователя в том регистре, в каком он его осуществляет.
!	Маска ввода заполняется слева направо, а не справа налево.
\	Знаки, следующие непосредственно за обратной косой чертой, отображаются без изменений.
""	Знаки, заключенные в двойные кавычки, отображаются без изменений.

## Элементы управления для построения пользовательского интерфейса

**richTextBox** Данный элемент управления дает возможность пользователю вводить и обрабатывать большие объемы информации (более 64 Кб). RichTextBox включает все возможности текстового редактора Microsoft Word.

**listBox** — простейший вариант пролистываемого списка. Он позволяет выбирать один или несколько хранящихся в списке элементов. Кроме того, ListBox имеет возможность отображать данные в нескольких колонках. Это позволяет представлять данные в большем объеме.

**toolTip** – всплывающая подсказка широко используется в Windows-приложениях, позволяет добавить подсказку практически любому компоненту, наследующему класс Control.

Элемент управления **BindingNavigator** является специализированным элементом управления **ToolStrip**, который предназначен для навигации и управления привязанными к данным элементами управления формы.

Для создания профессиональных интерфейсных элементов Windows-приложений, соответствующих последним требованиям эргономики и схожих с интерфейсными элементами приложений компании Microsoft, предлагается использовать набор интерфейсных компонентов:

- MenuStrip — компонент для создания основных меню приложений;
- ContextMenuStrip — компонент для создания контекстных меню приложений;
- StatusStrip — компонент для создания статусных панелей;
- ToolStrip — компонент для создания панелей инструментов.

### Компонент ToolStrip

Данный компонент служит в качестве базового класса для компонентов, реализующих меню и статусные панели, а также для создания панелей инструментов. Создаваемые с помощью компонента ToolStrip панели похожи на инструментальные панели в таких продуктах, как Microsoft Office, и обладают широкими возможностями настройки, включая поддержку визуальных шаблонов, а также поддерживают динамическое изменение расположения элементов.

Компонент ToolStrip может служить контейнером для следующих компонентов:

- ToolStripButton — реализует кнопку на инструментальной панели;
- ToolStripComboBox — реализует список на инструментальной панели;
- ToolStripSplitButton — реализует кнопку-разделитель на инструментальной панели;
- ToolStripLabel — реализует элемент инструментальной панели, который может содержать текст, графическое изображение или гиперссылку;
- ToolStripSeparator — реализует разделитель на инструментальной панели;
- ToolStripDropDownButton — реализует кнопку с раскрытием вложенных элементов на инструментальной панели;
- ToolStripTextBox — реализует строку ввода текста на инструментальной панели.

Помимо этого, можно использовать компонент ToolStripControlHost для размещения в инструментальной панели любых других компонентов Windows Forms.

Компонент ToolStrip обеспечивает базовую функциональность инструментальных панелей, включая отрисовку элементов, обработку событий от мыши и клавиатуры, поддержку операций drag-and-drop и т.п. Для расширения функциональности панелей инструментов можно воспользоваться классом ToolStripManager, чтобы объединить элементы панели в специальные горизонтальные или вертикальные области. Для получения более полного контроля над стилями и отрисовкой панелей можно использовать класс ToolStripRenderer.

### Компонент MenuStrip

Компонент MenuStrip представляет собой контейнер для создания структуры меню, каждый элемент которого реализуется с помощью компонента ToolStripMenuItem. Каждый такой элемент может либо представлять конкретную команду, либо содержать дочерние элементы.

Компонент MenuStrip может содержать компоненты ToolStripMenuItem, ToolStripComboBox,



ToolStripSeparator и ToolStripTextBox.

### **Компонент ContextMenuStrip**

Данный компонент используется для реализации контекстных меню, которые отображаются, когда пользователь щелкает правой кнопкой мыши по интерфейсному элементу или определенной области формы. Для связи интерфейсных элементов или формы с контекстным меню используется свойство ContextMenuStrip соответствующего элемента или формы.

Контекстное меню может состоять из следующих элементов: ToolStripMenuItem, ToolStripComboBox, ToolStripSeparator и ToolStripTextBox.

### **Компонент StatusStrip**

Этот компонент используется для создания статусных панелей и обычно состоит из нескольких элементов StatusStripPanel, добавляемых с помощью метода AddRange, каждый из которых может содержать либо текст, либо графическое изображение, либо и то и другое. Статусная панель также может содержать компоненты ToolStripLabel и ToolStripProgressBar.