**Приложение «Склад»**

**Необходимо разработать программу ввода названия товара, единицы измерения (шт, кг, л), количества товара на складе и цены реализации за единицу товара, добавления сведений по множеству имеющихся товара в поле многострочного редактора и расчёта общего количества, общей стоимости и средней стоимости имеющихся на складе товаров.**

**Входные данные**:  
Данные о товаре **Product**:  
Название товара **name** – строка, длина 255 символов, ввод с клавиатуры;  
Единица измерения **unit** - перечислимый тип: шт, кг, л, выбор из списка;  
Цена **price** – целое число, больше или равно 0 (в копейках), ввод с клавиатуры;  
Количество **amount** – вещественное число, больше или равно 0, ввод с клавиатуры.  
**Выходные данные**:  
Список товара на складе **products** - список объектов товар **Product** (для каждого: название, единица измерения, цена, количество, стоимость), вывод на экран и в файл;  
Общее количество товаров **totalAmount** - вещественное число, больше или равно 0, вывод на экран и в файл;  
Общая стоимость товаров **totalPrice** - вещественное число, больше или равно 0 (в рублях), вывод на экран и в файл;  
Средняя стоимость товара **averagePrice** – вещественное число, больше или равно 0 (в рублях), вывод на экран и в файл.

1. Функциональные характеристики проекта (что должна делать программа), начиная с получения входных данных (возможно несколько вариантов получения).
2. Обработка входных данных (что вычисляется, каким методом), выдачу результатов, дополнительные сервисные функции.

Для рассматриваемого примера:

Приложение должно выполнять следующие функции:

* + ввод данных об одном товаре с клавиатуры с помощью дружественного интерфейса пользователя;
  + добавление введенной информации о товаре в список товаров;
  + удаление из списка товара с указанным номером;
  + удаление всех товаров из списка (очистка списка);
  + расчет общего количества товаров в списке товаров;
  + расчет общей стоимости товаров в списке товаров;
  + расчет средней стоимости товара в списке товаров;
  + сохранение в файл информации о списке товаров;
  + чтение из файла сохраненной информации;
  + отображение информации о программе;
  + предоставление возможности вызова функций несколькими способами: с помощью кнопок, с помощью главного меню, с помощью контекстного меню, с помощью панели инструментов.

1. Создайте новый проект, выполнив для этого команду **Файл / Создать / Проект ...**. В открывшемся окне выберите **Шаблоны / Visual C#** и далее выберите **Приложение Windows Forms**.
2. Проверьте, что у вас **отмечено** поле "Создавать каталог для решения" (в этом случае поле "Имя решения" будет не доступно для задания) и **отмечено** поле "Добавить в систему управления версиями"
3. Укажите в качестве системы управления версиями систему **Git**
4. Разработайте спецификации отдельных компонент приложения.
   * Для того чтобы разработать спецификации классов приложения, продумайте, какие сущности имеются в задании? Дайте им названия.
   * Какие характеристики можно указать для каждой сущности? Какого они типа? Введите обозначения полей класса и укажите их тип.
   * Подумайте, какие параметры должны передаваться в конструктор класса? Сколько их? Какого типа они должны быть?
   * Какие функции должен выполнять класс? Введите обозначения методов класса. Для каждого метода решите, должен ли метод возвращать результат? Какого типа? Должны ли передаваться в метод параметры? Сколько их? Какого они типа? Должен ли быть доступ к методу из другого класса?
   * К каким полям должен быть доступ из вне, а какие являются внутренними?
   * Создайте диаграмму классов, указав для каждого класса:
     + поля с указанием типа и спецификатора доступа
     + конструктор с указанием передаваемых параметров
     + методы с указанием входных и выходных параметров и спецификатора доступа
     + Укажите отношения между классами

Для рассматриваемого примера можно выделить две сущности: **Product** (Товар) и **Stock** (Склад).

Сущность **Product** характеризуется следующими параметрами (поля класса):

* + **name: string** - название,
  + **amount: float**- количество,
  + **price: int**- цена,
  + **unit: enum Units** - единица измерения.

Конструктор должен содержать 4 параметра - характеристики товара:  
**Product(string name, float amount, int price, Units unit)**

Выполняемые функции (методы класса):

* + Представление товара в строковом виде:  
    **ToString()**- возвращает строку, параметров нет

Сущность **Stock** характеризуется следующими параметрами (поля класса):

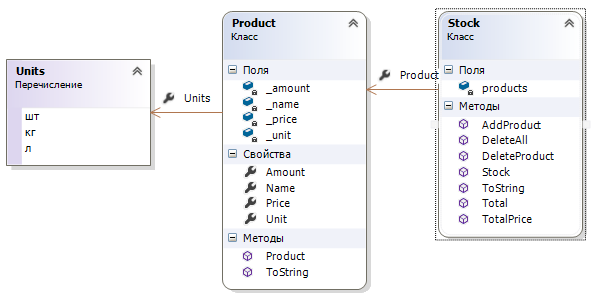
* + **products: List<Product>** - список товара

Конструктор не содержит параметров и создает пустой список товаров

Выполняемые функции (методы класса):

* + Добавление товара:  
    **AddProduct(string name, float amount, int price, Units unit)** - ничего не возвращает, принимает 4 параметра - характеристики товара
  + Удаление указанного товара:  
    **bool DeleteProduct(int index)**- возвращает логический тип (удачное удаление), принимает 1 параметр целого типа - номер удаляемой записи
  + Удаление всех товаров:  
    **DeleteAll()** - ничего не возвращает, не принимает параметры
  + Представление списка товаров в строковом виде:  
    **ToString()** - возвращает строку, параметров нет
  + Расчет общей стоимости товаров в списке товаров:  
    **double TotalPrice()** - возвращает вещественное число, не принимает параметры
  + Расчет количества товаров в списке товаров:  
    **double TotalAmount()** - возвращает вещественное число, не принимает параметры
  + Расчет средней стоимости товара в списке товаров:  
    **double AveragePrice()** - возвращает вещественное число, не принимает параметры

Диаграмма классов для рассматриваемого примера может иметь вид

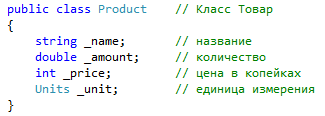
[](https://www.sites.google.com/site/erinacsharp/labs/lab-01/c_0525.PNG?attredirects=0)

1. Добавьте в проект файлы с классами, необходимыми для выполнения задания вашего варианта. При создании классов предусмотрите, чтобы **ваш код соответствовал принципам самодокументируемости** (т.е. все поля и методы имели значимые имена). Предусмотрите в классах:
   * поля для хранения значений,
   * свойства для работы с полями,
   * конструктор для создания объектов класса,
   * методы для работы с классом

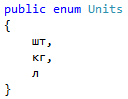
Рассмотрим выполнение этого пункта для приведенного выше примера.

1) Создадим необходимые классы:

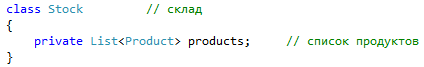
**Класс Product (Товар):**

[](https://www.sites.google.com/site/erinacsharp/labs/lab-01/c_0501.PNG?attredirects=0)

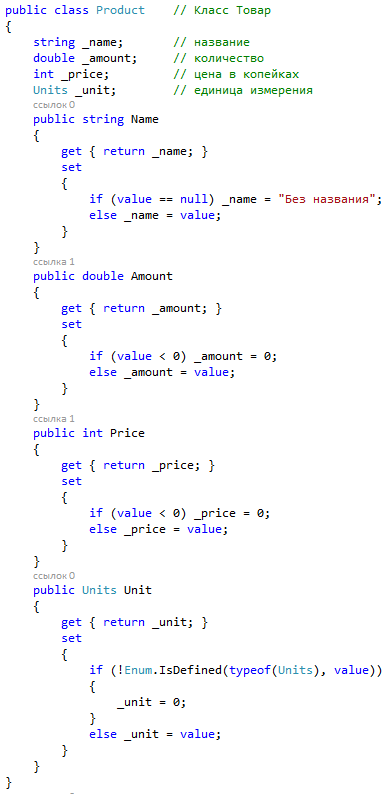
Поскольку у нас единица измерения имеет четко указанные конкретные значения, создадим для нее перечислимый **тип данных Units**:

**[](https://www.sites.google.com/site/erinacsharp/labs/lab-01/c_0502.PNG?attredirects=0)**

**Класс Stock (Склад):**

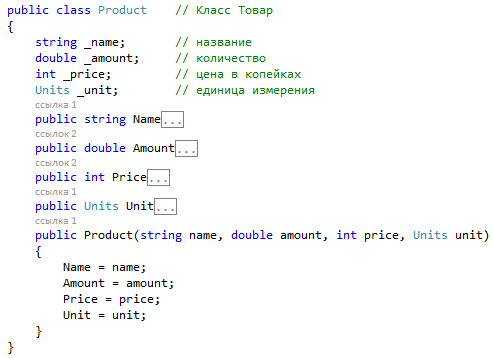
[](https://www.sites.google.com/site/erinacsharp/labs/lab-01/c_0503.PNG?attredirects=0)

2) Добавим в класс **Product** свойства, предусмотрев в них проверку правильности данных:

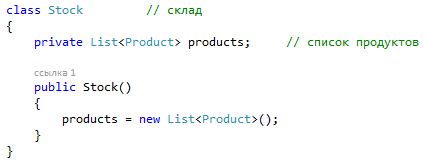
[](https://www.sites.google.com/site/erinacsharp/labs/lab-01/c_0504.PNG?attredirects=0)

Пояснение: в свойстве **Unit** проводится проверка, что переданное значение принадлежит нашему типу данных

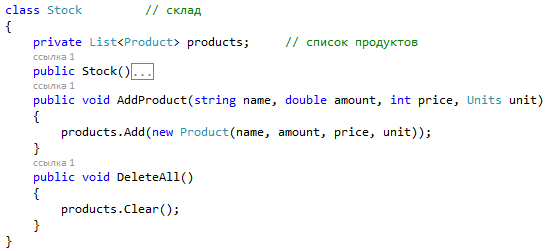
3) Добавим в класс **Product** конструктор:

[](https://www.sites.google.com/site/erinacsharp/labs/lab-01/c_0505.PNG?attredirects=0)

4) Добавим в класс **Stock** конструктор:

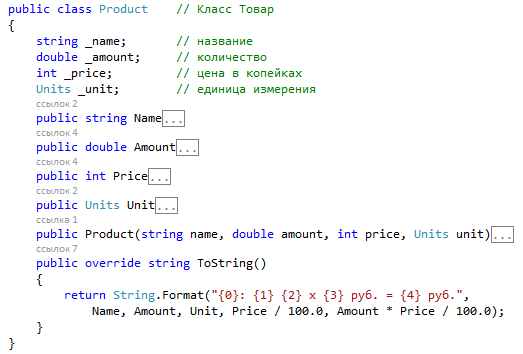
[](https://www.sites.google.com/site/erinacsharp/labs/lab-01/c_0506.PNG?attredirects=0)

5) Добавим в класс **Stock** методы, позволяющие добавлять товар на склад и удалять все товары со склада:

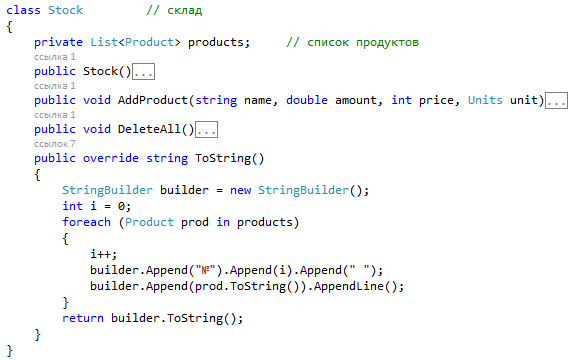
[](https://www.sites.google.com/site/erinacsharp/labs/lab-01/c_0507.PNG?attredirects=0)

6) Добавим в оба класса перегруженные методы ToString(), которые позволят выдавать строковое представление объекта класса.

Для класса **Product** должна формироваться строка, содержащая всю информацию о товаре:

[](https://www.sites.google.com/site/erinacsharp/labs/lab-01/c_0508.PNG?attredirects=0)

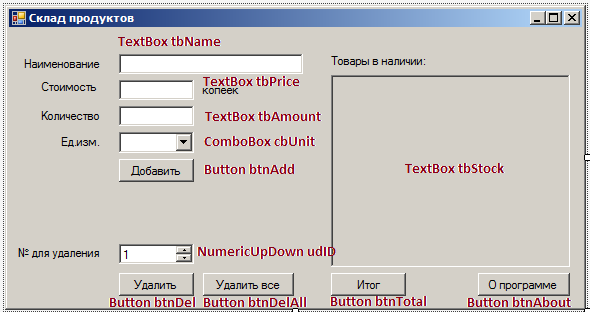
Для класса **Stock** должна формироваться строка, содержащая информацию о каждом товаре, имеющемся на складе:

[](https://www.sites.google.com/site/erinacsharp/labs/lab-01/c_0509.PNG?attredirects=0)

В методе **ToString(**) класса **Stock** в цикле перебираются все товары, имеющиеся на складе, и для каждого товара добавляется его номер в списке и информация о нем.

1. В процессе создания классов не забывайте периодически выполнять сборку проекта для своевременного устранения ошибок компиляции (меню СБОРКА / Собрать решение). На данный момент у вас должен быть корректный проект, который проходит сборку.
2. Продумайте, какие элементы интерфейса необходимо разместить на форме, чтобы обеспечить удобную и наглядную работу пользователя. (Внимание, исходя из учебных целей, интерфейс ОБЯЗАТЕЛЬНО должен содержать список **ListBox** или **ComboBox** и многострочный редактор для вывода информации). Перенесите на форму необходимые компоненты.

Создадим форму для отображения информации в рассматриваемом примере. Внешний вид формы для данного этапа работы показан ниже. На форме уже предусмотрены компоненты для пока еще не реализованных методов удаления конкретного элемента, для вывода итоговой информации и для отображения формы "О программе". Для каждого компонента ОБЯЗАТЕЛЬНО задайте ЗНАЧИМЫЕ названия (свойство **Name**) и необходимые значения для отображения (свойство **Text**). В таблице после рисунка показаны дополнительные свойства, которые были настроены для элементов в рассматриваемом примере.

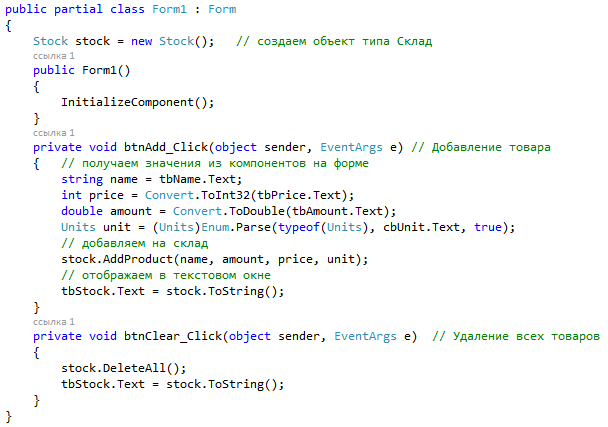
[](https://www.sites.google.com/site/erinacsharp/labs/lab-01/c_0510.png?attredirects=0)

***Таблица 1. Изменяемые свойства компонентов***

|  |  |  |
| --- | --- | --- |
| **Название компонента (свойство Name)** | **Тип компонента** | **Измененное свойство и его значение** |
| cbUnit | ComboBox | **Items = шт, кг и л.** - значения, которые будут отображаться в выпадающем списке, должны совпадать с перечислимым типом **DropDownStyle = DropDownList** - стиль компонента, задаем только выбор из списка, без возможности вводить свое значение |
| tbStock | TextBox | **MultiLine = True** - возможность работы с многострочным текстом **ReadOnly = True** - только для чтения, без возможности редактирования |
| udID | NumericUpDown | **Min = 1** - минимальное значение номера товара |

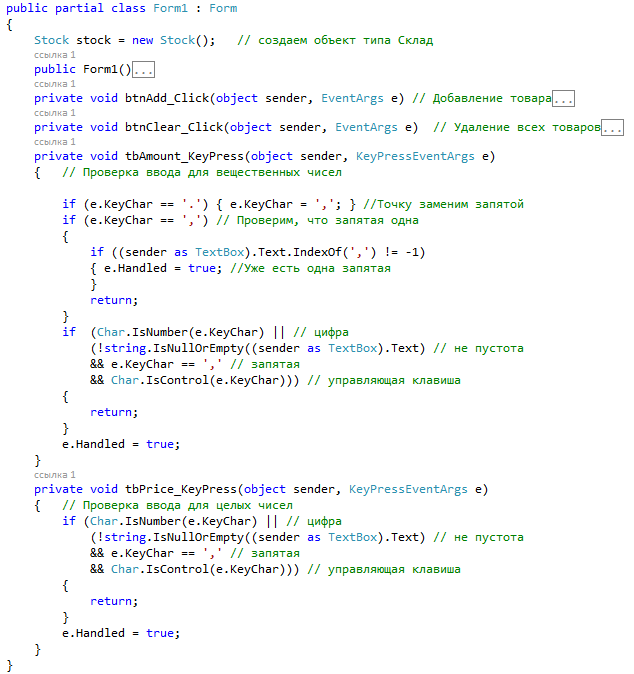
1. Создайте необходимые обработчики событий для формы, чтобы организовать взаимодействие с пользователем

Для рассматриваемого примера код класса формы может выглядеть так:

[](https://www.sites.google.com/site/erinacsharp/labs/lab-01/c_0511.PNG?attredirects=0)

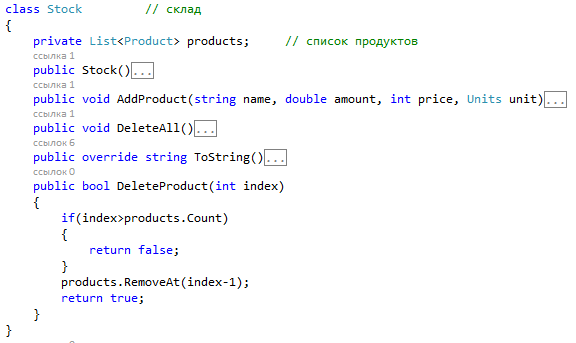
1. Для того, чтобы защитить приложение от некорректного ввода пользователя, создайте методы проверки для числовых полей.

Метод проверки ввода может быть реализован как обработчик события **KeyPress** для текстовых полей. Ниже приведены два примера - для ввода вещественных и ввода целых чисел.

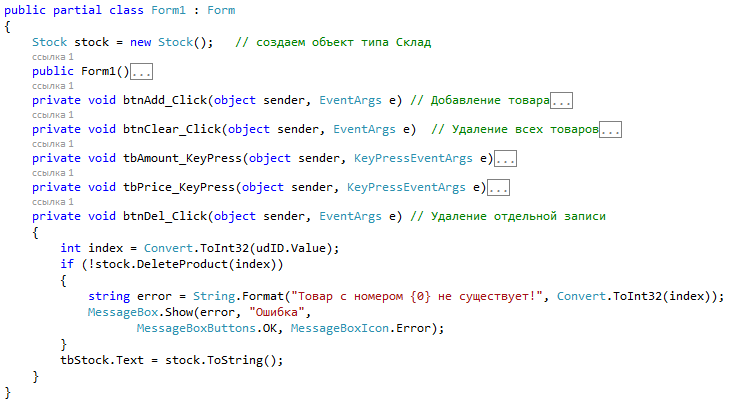
[](https://www.sites.google.com/site/erinacsharp/labs/lab-01/c_0513.PNG?attredirects=0)

1. Отладьте приложение. В нем должен быть полностью рабочим функционал добавления одной записи и удаления всего содержимого. Для всех полей должен обеспечиваться корректный ввод.
2. Добавьте возможность удалять только указанную позицию.

Для рассматриваемого примера необходимо добавить новый метод в класс **Stock**, в который будет передаваться номер удаляемого элемента. Данный метод проверяет наличие записи с указанным номером. В случае успешного удаления метод возвращает значение **true**. Если такой записи нет, то возвращается **false**

[](https://www.sites.google.com/site/erinacsharp/labs/lab-01/c_0512.PNG?attredirects=0)

Вызов данного метода может быть реализован следующим образом:

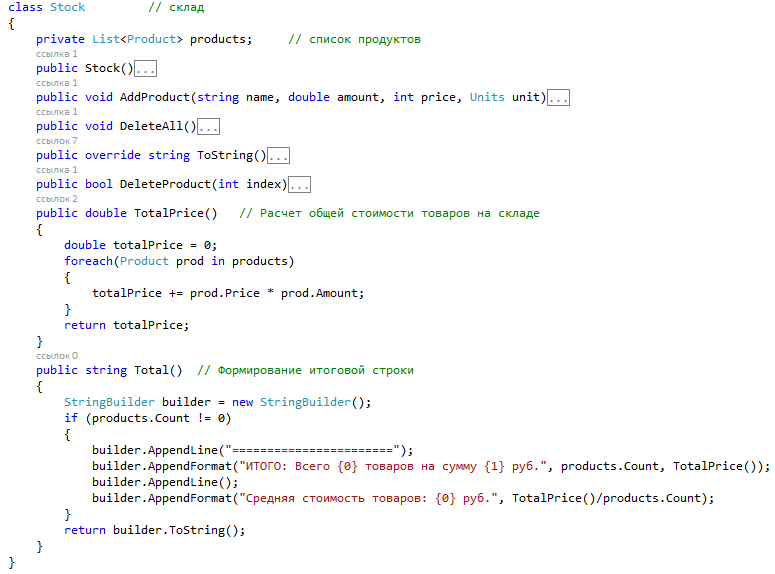
[](https://www.sites.google.com/site/erinacsharp/labs/lab-01/c_0514.PNG?attredirects=0)

В данном обработчике события в случае неудачной попытки удаления выдается сообщение об ошибке с помощью класса **MessageBox**.

1. Добавьте в приложение вывод расчетной информации.

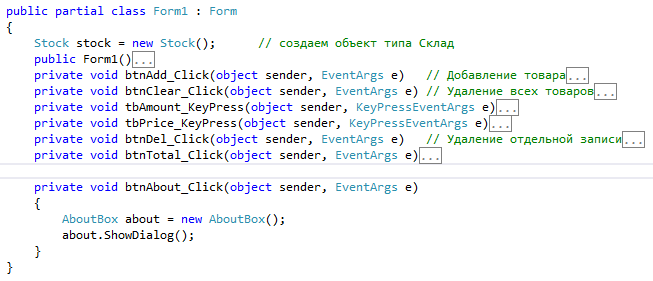
В рассматриваемом примере по заданию требуется выполнить расчёт общего количества, общей стоимости и средней стоимости имеющихся на складе товаров.

Для реализации этой возможности добавим в класс **Stock**дополнительные методы:

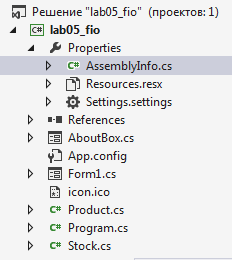
[](https://www.sites.google.com/site/erinacsharp/labs/lab-01/c_0515.PNG?attredirects=0)

В методе **TotalPrice()** выполняется расчет общей стоимости продуктов на складе, а в методе **TotalPrice()** формируется итоговая строка.

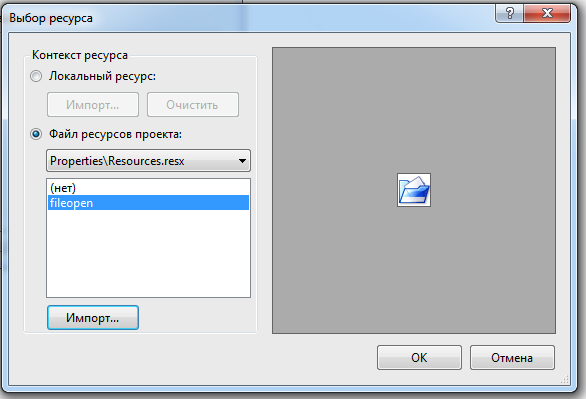
1. Реализуйте отображение формы "О программе". Для этого выполните команду из главного меню **ПРОЕКТ / Добавить форму Windows**, найдите пункт Окно "О программе". **НЕ ВНОСИТЕ пока ИЗМЕНЕНИЯ в форму!** Вызов формы "О программе" реализуйте в обработчике события кнопки **btnAbout**

[](https://www.sites.google.com/site/erinacsharp/labs/lab-01/c_0516.PNG?attredirects=0)

1. Для того, чтобы изменить данные, отображаемые на форме "О программе", откройте файл **AssemblyInfo.cs** и внесите изменения в глобальные параметры проекта. Посмотреть назначение каждого параметра можно в справке: <https://msdn.microsoft.com/ru-ru/library/284c1c4s(v=vs.90).aspx>

[](https://www.sites.google.com/site/erinacsharp/labs/lab-01/c_0517.PNG?attredirects=0)

1. Добавьте на форму "О программе" недостающую информацию, чтобы она соответствовала  заданию. (***Цитата****: В приложении должна быть форма "О программе", на которой должна отображаться следующая информация: Фамилия, имя, фотография автора, номер группы, номер варианта, номер и название лабораторной работы*).
2. Добавьте две кнопки, которые позволят сохранить информацию в файле и вывести содержимое файла в окно. Настройте изображения на кнопках с помощью свойства **Image**:

[](https://www.sites.google.com/site/erinacsharp/labs/lab-01/c_0518.PNG?attredirects=0)

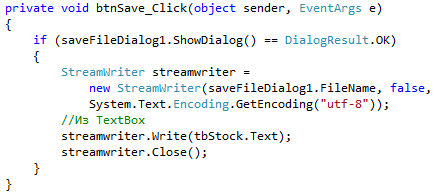
1. Добавьте на форму компоненты **OpenFileDialog** и **SaveFileDialog**. Компоненты **OpenFileDialog** и **SaveFileDialog** не визуальные и, потому, разместились ниже формы. Настройте необходимые свойства компонентов, чтобы обеспечить комфортную работу пользователя. **ОБЯЗАТЕЛЬНО НАСТРОЙТЕ** фильтр файлов и имя файла по умолчанию.

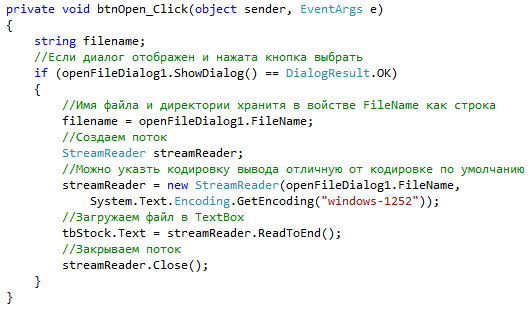
***Таблица 2. Основные свойства OpenFileDialog и SaveFileDialog***

|  |  |
| --- | --- |
| **Свойство** | **Описание** |
| AddExtension | разрешает или запрещает автоматическое добавление расширение, указанное в свойстве DefaultExt |
| DefaultExt | расширение, принятое по умолчанию для автоматического добавления к имени файла при AddExtension=true |
| CheckFileExists | используется для получения или установки значения, указывающего отображать или нет диалоговое окно предупреждения, если пользователь указал в свойстве FileName имя файла, которого не существует в данной директории и нажал кнопку "Открыть" при невыбранном кликом мышки файле. При CheckFileExists=true вместо прерывания будет выдано сообщение, что такого файла нет и исключения не вызывается. |
| FileName | имя файла по умолчанию для выборки если была нажата кнопка OK и не выбран кликом мышки файл в окне диалога |
| CheckPatchExists | используется для получения или установки значения, указывающего отображать или нет диалоговое окно предупреждения, если пользователь указал в свойстве FileName имя файла с несуществующим именем директории |
| DereferenceLinks | используется для указания контроллу что именно возвращать при выборе файла ссылки - файл по ссылке (true) или файл самой ссылки (false) |
| Filter, FilterIndex | фильтр для выбираемых файлов и индекс строки, отображаемой в окошечке "Имя файла". Например значение свойства заданного строкой при FilterIndex=1 rtf файлы (\*.rtf)|\*.rtf|txt файлы(\*.txt)|\*.txt позволит выбрать только текстовые файлы в формате rtf или txt. В окошечке "Тип файла" будет только две строки: rtf файлы (\*.rtf) txt файлы(\*.txt) а в окошечке "Имя файла" будет отображена первая |
| InitialDirectory | директория, которая выбирается при старте OpenFileDialog |
| MultiSelect | при значении true позволяет выбрать мышкой при нажатой кнопке Shift или Ctrl несколько файлов и сохранить их имена в свойстве FileNames в виде массива строк |
| ReadOnlyChecked | при значении true позволяет открывать команде OpenFile выбранные файлы только в режиме чтения |
| RestoreDirectory | при true диалоговое окно восстанавливает текущий каталог к первоначальному значению если пользователь изменил каталог при поиске файлов, при false - нет. Свойство работает только если закрыт поток Stream, созданный методом openFileDialog1.OpenFile(). |
| ShowHelp | при значении true в окне диалога отображается кнопка "Справка" |
| ShowReadOnly | при значении true в окне диалога отображается переключатель "Только для чтения" |
| Title | заголовок диалогового окна |
| ValidateNames | при true проверяет допустимость имени для файла применительно к Win32. Никакое заданное по умолчанию в этом случае не добавляется |

Отметим, что все свойства могут быть заданы на этапе проектирования и программно - путем присвоения соответствующих значений.

1. Организовать запись в файл и чтение из файла можно с помощью следующего кода:

[](https://www.sites.google.com/site/erinacsharp/labs/lab-01/c_0519.PNG?attredirects=0)

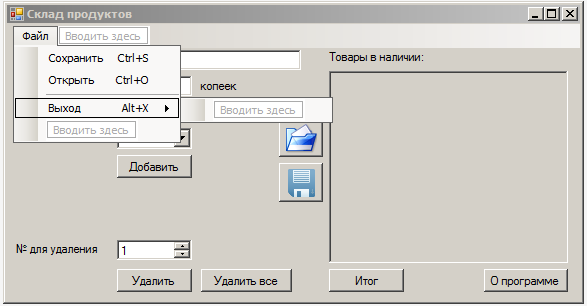
[](https://www.sites.google.com/site/erinacsharp/labs/lab-01/c_0520.PNG?attredirects=0)

1. Добавить возможность выполнения основных функций формы с помощью главного и контекстного меню.

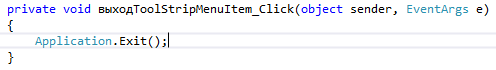
Для работы с меню необходимо добавить на форму элемент управления **MenuStrip**. Это не визуальный компонент, поэтому он отобразится в области внизу формы. В самом верху формы отображается поле, позволяющее создавать пункты меню. Создайте первый пункт главного меню "*Файл*" со следующей структурой, показанной на рисунке.

Для создания полосы-разделителя достаточно задать символ "-" в качестве текста пункта меню (один-единственный минус!).

Для задания горячих клавиш для пунктов меню, настройте для них свойство **ShortcutKeys**.

[](https://www.sites.google.com/site/erinacsharp/labs/lab-01/c_0521.PNG?attredirects=0)

1. Добавьте пункт главного меню и подменю для него, позволяющие работать с информацией (например, добавлять, удалять, вычислять итог...). И далее еще пункт меню "*Помощь*" с подпунктом "*О программе*".
2. Создайте обработчик события для пункта меню *Выход* (сделайте двойной щелчок на пункте меню):

[](https://www.sites.google.com/site/erinacsharp/labs/lab-01/c_0522.PNG?attredirects=0)

Обратите внимание, как называется пункт меню. Да, не удивляйтесь, Visual Studio позволяет давать русские имена переменным, классам, компонентам и т.д. Вы вполне можете этим пользоваться, создавая самодокументируемые приложения на родном языке. (Одно неудобство, возникающее при этом - это необходимость постоянно переключать язык ввода на клавиатуре...)

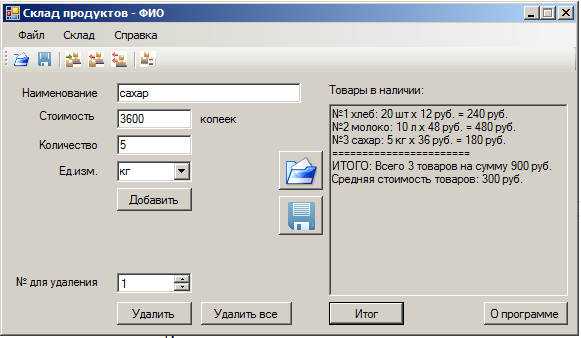
1. Назначим пунктам меню те же действия, которые выполняют кнопки. **НИКОГДА НЕ ПИШИТЕ ОДИН И ТОТ ЖЕ КОД ДВАЖДЫ!**  Выберите пункт меню *Сохранить*, перейдите в **окно Свойств** и перейдите на **вкладку События** (вверху окна Свойства кнопочка с молнией :-) ), найдите событие **Click**и в выпадающем списке выберите уже существующий обработчик события сохранения в файл. Этим действием вы назначили один обработчик двум компонентам. В итоге у вас код написан один раз, но работает по двум разным событиям. Выполните подобное назначение ***для всех пунктов меню***.
2. Добавим возможность выполнять сохранение в файл и чтение из файла с помощью контекстного меню. Запустите свое приложение и посмотрите, что у формы на данный момент нет контекстного меню, однако у текстовых окон контекстное меню имеется по умолчанию. Контекстных меню, в отличие от главного, на форме может быть много, и они присоединяются к различным компонентам и могут меняться в зависимости от ситуации. Для создания контекстного меню добавьте компонент **ContextMenuStrip**. Этот компонент также является невизуальным и располагается ниже формы. Принцип работы с компонентом такой же, как и с главным меню **MenuStrip**. Выделение компонента отображает вверху формы визуальный построитель меню. Присоедините существующие обработчики событий к созданным пунктам меню.
3. Для того, чтобы присоединить контекстное меню к форме, необходимо для формы задать свойство **ContexMenuStrip** равным нужному компоненту. Назначение контекстного меню можно динамически менять в процессе выполнения приложения. Проверьте работоспособность вашего контекстного меню.
4. (***Не обязательный, но очень желательный. За выполнение +0,5 балла***) Добавим возможность выполнения всех действий с помощью панели инструментов. Для этого разместите компонент **ToolStrip**. Для добавления на него кнопок воспользуйтесь визуальным построителем. Для разделения кнопок на логические группы добавляйте элемент **Separator**. Добавьте кнопки для всех имеющихся в приложении действий. Настройте изображения на кнопках (свойство **Image**).
5. Ну и еще несколько "штрихов" к приложению. У главной формы задайте заголовок, отражающий суть задания и содержащий вашу фамилию (например, Склад - Автор ФИО) (свойство **Text** у формы). Значение свойства **MaximizeBox** сделайте **false** - чтобы заблокировать кнопочку разворота окна на весь экран. Свойство **FormBorderStyle** сделайте **FixedSingle** - это нужно для того, чтобы нельзя было менять размер окна. Свойство **StartPosition=Center** - для отображения формы по центру экрана.
6. (***Не обязательно, но желательно. +0,5 балла за выполнение***) **По многочисленным просьбам**: ***как добавить окно с запросом подтверждения удаления перед удалением записи или очисткой всего содержимого***?

Простейшее модальное диалоговое окно можно создать на базе класса **MessageBox**. Вы уже использовали этот класс для вывода системного сообщения. Полный перечень параметров метода **MessageBox.Show**, позволяющих задать необходимый внешний вид диалоговой панели, а также количество и тип расположенных на ней кнопок, можно посмотреть здесь: <https://msdn.microsoft.com/ru-ru/library/szwxe9we(v=vs.110).aspx>.

Изменим обработчик события кнопки Удалить, добавив в него предупреждение перед удалением данных по товару:

[](https://www.sites.google.com/site/erinacsharp/labs/lab-01/c_0523.PNG?attredirects=0)

1. Внешний вид полученного готового приложения показан ниже:

[](https://www.sites.google.com/site/erinacsharp/labs/lab-01/c_0524.PNG?attredirects=0)

1. Составление тестовых данных на основе потока данных (тестирование методом "белого ящика").
   * Для того чтобы составить тестовый набор на основе потока данных, необходимо разбить входной поток данных на классы эквивалентности. Каждому классу присваивается номер. Затем подготавливаются такие наборы входных данных, которые позволяют проверить выполнение программы по каждому классу эквивалентности хотя бы один раз.
   * Внимательно изучите условие задачи. Составьте структуру входных и выходных данных (введите обозначения, укажите типы данных). Подумайте, какие граничные условия накладываются на входные данные.
   * Разбейте входной поток данных на классы эквивалентности. Присвойте каждому классу эквивалентности свой номер.
   * Занесите описание каждого класса в таблицу, в графу «Классы эквивалентности».
   * Занесите в таблицу наборы входных данных, соответствующих каждому классу эквивалентности, в графу «Тестовый набор».

Пример оформления таблицы тестовых данных на основе потока данных:

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Классы эквивалентности** | **Тестовый набор** | **Ожидаемые результаты** |
| 1 | Класс допустимых значений при вводе | *Наименование*: Молоко  *Стоимость*: 4800 *Количество*: 10 *Ед.изм.*: л Нажать кнопку *Добавить* Нажать кнопку *Итог* | В поле со списком товара отобразилась одна строка: №1 Молоко: 10 л х 48 руб. = 480 руб. ====================== ИТОГО: Всего 1 товаров на сумму 480 руб. Средняя стоимость товаров: 480 руб. |
| 2 | Класс допустимых значений при вводе | *Наименование*: Хлеб *Стоимость*: 1200 *Количество*: 20 *Ед.изм.*: шт. Нажать кнопку *Добавить* Нажать кнопку *Итог* | В поле со списком товара отобразились две строки: №1 Молоко: 10 л х 48 руб. = 480 руб. №2 Хлеб: 20 шт х 12 руб. = 240 руб. ====================== ИТОГО: Всего 2 товаров на сумму 720 руб. Средняя стоимость товаров: 360 руб. |
| 3 | Класс допустимых значений при вводе | *Наименование*: Сахар *Стоимость*: 3600 *Количество*: 5 *Ед.изм.*: кг Нажать кнопку *Добавить* Нажать кнопку *Итог* | В поле со списком товара отобразились три строки: №1 Молоко: 10 л х 48 руб. = 480 руб. №2 Хлеб: 20 шт х 12 руб. = 240 руб. №3 Сахар: 5 кг х 36 руб. = 180 руб. ====================== ИТОГО: Всего 3 товаров на сумму 900 руб. Средняя стоимость товаров: 300 руб. |
| 4 | Класс допустимых значений при вводе, граничные условия | *Наименование*: Масло *Стоимость*: 0 *Количество*: 10 *Ед.изм.*: шт Нажать кнопку *Добавить* Нажать кнопку *Итог* | В поле со списком товара отобразились четыре строки: №1 Молоко: 10 л х 48 руб. = 480 руб. №2 Хлеб: 20 шт х 12 руб. = 240 руб. №3 Сахар: 5 кг х 36 руб. = 180 руб. №4 Масло: 0 шт х 0 руб. = 0 руб. ====================== ИТОГО: Всего 4 товаров на сумму 900 руб. Средняя стоимость товаров: 250 руб. |
| 5 | Класс недопустимых значений при вводе | *Наименование*:  *Стоимость*: 1 *Количество*: 1 *Ед.изм.*: шт Нажать кнопку *Добавить* Нажать кнопку *Итог* | В поле со списком товара отобразились четыре строки: №1 Молоко: 10 л х 48 руб. = 480 руб. №2 Хлеб: 20 шт х 12 руб. = 240 руб. №3 Сахар: 5 кг х 36 руб. = 180 руб. №4 Масло: 0 шт х 0 руб. = 0 руб. №5 : 1 литр х 0,01 руб = 0,01 руб ====================== ИТОГО: Всего 5 товаров на сумму 900,01 руб. Средняя стоимость товаров: 180,00 руб. |
| 6 | Класс недопустимых значений при вводе | *Наименование*: Лимонад *Стоимость*:  *Количество*: 10 *Ед.изм.*: шт Нажать кнопку *Добавить* Нажать кнопку *Итог* | ??? (при необходимости добавить обработчик исключения!!!) |
| 7 | Класс недопустимых значений при удалении | Указать *№ для удаления*: 7  Нажать кнопку *Удалить* | Отображается окно с текстом: "Товар с номером 4 не существует" В поле со списком товара отобразились четыре строки: №1 Молоко: 10 л х 48 руб. = 480 руб. №2 Хлеб: 20 шт х 12 руб. = 240 руб. №3 Сахар: 5 кг х 36 руб. = 180 руб. №4 Масло: 0 шт х 0 руб. = 0 руб. №5 : 1 литр х 0,01 руб = 0,01 руб |
| 8 | Класс допустимых значений при удалении | Указать *№ для удаления*: 2  Нажать кнопку *Удалить* | В поле со списком товара отобразились три строки: №1 Молоко: 10 л х 48 руб. = 480 руб. №2 Сахар: 5 кг х 36 руб. = 180 руб. №3 Масло: 0 шт х 0 руб. = 0 руб. №4 : 1 литр х 0,01 руб = 0,01 руб |
| 9 | Класс допустимых значений при удалении, граничные условия | Указать *№ для удаления*: 4  Нажать кнопку *Удалить* | В поле со списком товара отобразились две строки: №1 Молоко: 10 л х 48 руб. = 480 руб. №2 Сахар: 5 кг х 36 руб. = 180 руб. №3 Масло: 0 шт х 0 руб. = 0 руб. |
| 10 | Класс допустимых значений при удалении, граничные условия | Указать *№ для удаления*: 1  Нажать кнопку *Удалить* | В поле со списком товара отобразилась одна строка: №1 Сахар: 5 кг х 36 руб. = 180 руб. №2 Масло: 0 шт х 0 руб. = 0 руб. |
| 11 | Класс допустимых значений при удалении, граничные условия | Указать *№ для удаления*: 2  Нажать кнопку *Удалить* | В поле со списком товара отобразилась одна строка: №1 Сахар: 5 кг х 36 руб. = 180 руб. |
| 12 | Класс допустимых значений при удалении, граничные условия | Указать *№ для удаления*: 1  Нажать кнопку *Удалить* | В поле со списком нет строк (пусто) |
| 13 | Класс недопустимых значений при удалении | Указать *№ для удаления*: 1  Нажать кнопку *Удалить* | ??? (при необходимости добавить обработчик исключения) |

1. Составление протокола испытаний.
   * Для того чтобы составить протокол испытаний, разработайте структуру входных и выходных данных, укажите, из каких источников будут получены входные данные, в какой форме и куда будут выводится результаты.
   * Подробно распишите функциональные характеристики проекта.
   * Все функциональные характеристики необходимо проверить в ходе испытаний и внести в протокол. Сначала обычно проверяется способность программы обеспечить ввод исходных данных. Если возможны несколько вариантов ввода (например, в разной форме, из разных источников), то проверяется каждый способ ввода.
   * Затем обычно проверяется способность программы осуществлять контроль вводимых данных, а также описывается поведение программы в случае, если данные не получены (не введены).
   * Далее в протокол испытаний вносятся все варианты выполнения расчётов и обработки данных. Здесь необходимо проверить работоспособность программы при вводе допустимых и не допустимых входных данных.
   * В последнюю очередь проверяется способность программы осуществлять вывод результатов. Если возможны несколько вариантов вывода (например, в разной форме, на разные носители), то проверяется каждый способ вывода.

Пример оформления протокола испытаний:

|  |  |  |
| --- | --- | --- |
| **Проверяемые требования** | **Действия пользователя и вводимые значения** | **Результаты и сообщения программы** |
| Способность программы обеспечить ввод исходных данных | 1) Ввести в поля ввода исходные данные: *Наименование*: Молоко  *Стоимость*: 4800 *Количество*: 10 *Ед.изм.*: л | В поля ввода можно ввести данные в соответствии с типом поля:  *Наименование*: допустимы любые символы *Стоимость*: допустимы только целые положительные числа и 0 *Количество*: допустимы только положительные числа (целые и вещественные) и нуль Можно стирать символы в поле ввода, используя клавиши BackSpace и Delete *Ед.изм.*: допустим только выбор из списка |
| 2) Ввести в поля ввода исходные данные: *Наименование*: Хлеб *Стоимость*: 1200 *Количество*: 2,5 *Ед.изм.*: шт. |  |  |
| Способность программы обеспечить контроль вводимых данных. | 1) В поле ввода *Наименование*: ввести следующие значения: а) Сахар б) любые символы с клавиатуры в) удалить символы | В поле ввода *Наименование*: можно ввести данные в соответствии с типом поля *Строка*: Допустимы любые символы. Можно стирать символы в поле ввода, используя клавиши BackSpace и Delete |  |  |
| 2) В поле ввода *Стоимость*: попробовать ввести следующие значения: а) символов; б) отрицательного числа: -10; в) вещественного числа: 36.3; г) нулевого значения: 0 | В поля ввода *Стоимость*: можно ввести данные в соответствии с типом поля *Целое число*: Допустимы только цифры, недопустимые символы игнорируются. Можно стирать символы в поле ввода, используя клавиши BackSpace и Delete |  |  |
| 3) В поле ввода *Количество*: попробовать ввести следующие значения: а) символов; б) отрицательного числа: -5; в) вещественного числа: 5.3; г) нулевого значения: 0 д) две десятичные точки: 5.3.4 | В поле ввода *Количество*: можно ввести данные в соответствии с типом поля *Вещественное число*: Допустимы только цифры и точка, недопустимые символы игнорируются. Точка вводится только один раз. Можно стирать символы в поле ввода, используя клавиши BackSpace и Delete |  |  |
| 4) В поле ввода *Ед.изм.*: попробовать ввести следующие значения: а) шт. б) ввод значения, не имеющегося в списке | В поле ввода *Ед.изм.*: можно только выбрать данные из списка Допустим только выбор из списка |  |  |
| Способность программы обеспечить контроль пустых полей ввода. | Оставить одно или несколько полей ввода пустыми и нажать на кнопку *Добавить*. | Должно быть выведено сообщение об ошибке, приложение не закрывается, пользователь имеет возможность продолжить работу. |  |  |
| Проверить способность программы добавлять введенную информации о товаре в список товаров | В поля ввода ввести данные нескольких товаров, каждый раз нажимая кнопку *Добавить*. | В поле со списком товара должны отобразиться строки, содержащие информацию по каждому товару |  |  |
| Проверить способность программы определять итоговые данные при наличии товаров в списке. | 1) В поля ввода ввести данные товара: *Наименование*: Молоко  *Стоимость*: 4800 *Количество*: 10 *Ед.изм.*: л Нажать кнопку *Добавить* Нажать кнопку *Итог* | В поле со списком товара отобразились строки:  №1 Молоко: 10 л х 48 руб. = 480 руб. ====================== ИТОГО: Всего 1 товаров на сумму 480 руб. Средняя стоимость товаров: 480 руб. |  |  |
|  | 2) В поля ввода ввести данные товара: *Наименование*: Хлеб *Стоимость*: 1200 *Количество*: 20 *Ед.изм.*: шт Нажать кнопку *Добавить* Нажать кнопку *Итог* | В поле со списком товара отобразились строки:  №1 Молоко: 10 л х 48 руб. = 480 руб. №2 Хлеб: 20 шт х 12 руб. = 240 руб. ====================== ИТОГО: Всего 2 товаров на сумму 720 руб. Средняя стоимость товаров: 360 руб. |  |  |
|  | 3) В поля ввода ввести данные товара: *Наименование*: Сахар *Стоимость*: 3600 *Количество*: 5 *Ед.изм.*: кг Нажать кнопку *Добавить* Нажать кнопку *Итог* | В поле со списком товара отобразились строки:  №1 Молоко: 10 л х 48 руб. = 480 руб. №2 Хлеб: 20 шт х 12 руб. = 240 руб. №3 Сахар: 5 кг х 36 руб. = 180 руб. ====================== ИТОГО: Всего 3 товаров на сумму 900 руб. Средняя стоимость товаров: 300 руб. |  |  |
| Проверить способность программы удалять товар с указанным номером | Ввести несколько товаров. Указать номер существующего товара | Выводится окно с сообщением "Удалить данные по товару с №Х" и двумя кнопками: ОК и Отмена При нажатии на ОК выводится сообщение "Товар с номером Х удален" В списке товаров отсутствует строка с выбранным товаром При нажатии на Отмена выводится сообщение "Отмена удаления товара" В списке товаров все товары присутствуют |  |  |
| Проверить способность программы проверять наличие товара с указанным номер перед удалением | Ввести несколько товаров. Указать номер не существующего товара | Отображается окно с текстом: "Товар с номером Х не существует" В списке товаров все товары присутствуют |  |  |
| Проверить способность программы осуществлять вывод результатов в файл. | Ввести несколько товаров. Рассчитать итоговые значения. Нажать кнопку сохранения в файл. Ввести имя файла для сохранения | Содержимое текстового окна со списком товаров выводится в текстовый файл с указанным именем и расширением txt. |  |  |
| Проверить способность программы осуществлять чтение данных из файла | Нажать кнопку открытия файла. Выбрать файл на диске. Открыть файл | Содержимое файла выводится в текстовое окно со списком товара |  |  |