**~~Практическое занятие №7 Визуальные компоненты для работы с текстовыми данными.~~**

**~~1 Цель и порядок работы~~**

~~Цель практического занятия – познакомиться с основными операциями работы с файлами различных форматов и компонентами для работы с текстовой информацией.~~

~~Порядок выполнения работы:~~

* ~~ознакомиться с описанием практического занятия;~~
* ~~получить задание у преподавателя;~~
* ~~написать программу и отладить ее на ЭВМ.~~
* **~~Задание~~**
  1. ~~Создать проект, согласно задания. Отладить и протестировать программу.~~
  2. ~~Защитить выполненную работу.~~
  3. ~~Базовый уровень (задание 1 -3), для студентов, претендующих на оценку 3. Повышенный уровень (задания 1 -4) для студентов, претендующих на оценку 4 или~~

~~5.~~

* 1. ~~Информационные ресурсы:~~

[~~https://metanit.com/sharp/windowsforms/~~](https://metanit.com/sharp/windowsforms/)

[~~https://metanit.com/sharp/tutorial/5.1.php~~](https://metanit.com/sharp/tutorial/5.1.php)

[~~https://metanit.com/sharp/tutorial/6.5.php~~](https://metanit.com/sharp/tutorial/6.5.php)

[~~https://metanit.com/sharp/tutorial/16.1.php~~](https://metanit.com/sharp/tutorial/16.1.php)

1. ~~Схема работы:~~

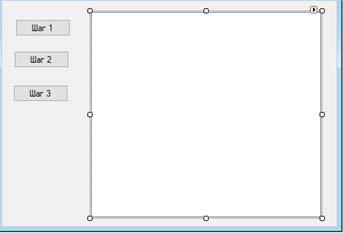
|  |  |  |
| --- | --- | --- |
| ~~№ занятия~~ |  | ~~Виды выполняемых работ.~~ |
| ~~6~~ | ~~в аудитории~~ | ~~Изучить теоретический материал согласно инфор-~~ |
|  |  | ~~мационных ресурсов.~~ |
|  |  | ~~Разработка приложений WindowsForm 1 и 2 зада-~~ |
|  |  | ~~ние.~~ |
|  | ~~самостоятельно~~ | ~~Разработка приложений WindowsForm 3 – базовый~~ |
|  |  | ~~уровень и 3-4 задания повышенный уровень.~~ |
| ~~7~~ | ~~в аудитории~~ | ~~Ответить на теоретические вопросы преподава-~~ |
|  |  | ~~теля, защитить выполненные проекты практиче-~~ |
|  |  | ~~ского занятия. После защиты прикрепить архивы~~ |
|  |  | ~~проектов в Microsoft Teams. Название архивов~~ |
|  |  | ~~должно содержать ФИО студента № практиче-~~ |
|  |  | ~~ского занятия и номер проекта. Например: Ива-~~ |
|  |  | ~~нов\_ Иван\_Иванович\_ПЗ\_7\_задание\_1.~~ |
|  |  | ~~Защита проектов подразумевает понимание сту-~~ |
|  |  | ~~дентом всех особенностей создания приложений и~~ |
|  |  | ~~всех их внутренних элементов.~~ |
|  |  |  |

**~~3 Формулировка заданий~~**

**~~Проект 1. «Запись и чтение данных в текстовых и бинарных файлах»~~**

**~~Шаг 1~~**~~. Создайте новый проект. Дайте форме заголовок «Пр.занятие. 7\_1. Петров Сергей» с Вашими фамилией и именем.~~

**~~Шаг 2~~** ~~. На форме расположите три кнопки и компонент~~ **~~textBox1~~**~~, как показано на рисунке:~~

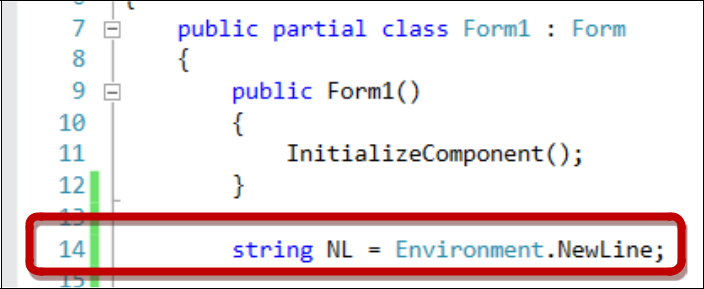
~~~~

~~Компонент типа~~ **~~TextBox~~** ~~будем использовать для вывода различной текущей ин-формации (по типу консоли). Чтобы можно было отображать несколько строк устано-вите свойство~~ **~~MultiLine~~** ~~равное~~ **~~true~~**~~. Сделайте 2-ю и 3-ю кнопки невидимыми.~~

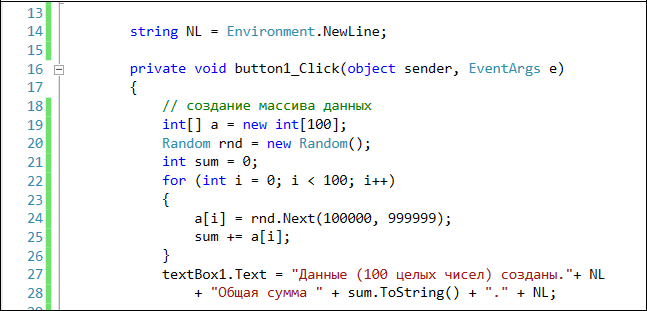
**~~Шаг 3~~**~~. Для работы с файлами, в блоке используемых пространств имён «using» в начале файла «Form1.cs» необходимо добавить пространство имён~~ **~~System.IO~~**~~.~~

~~~~

**~~Шаг 4~~**~~. Для того чтобы выводить информацию построчно в textBox1, необходимо определить переменную, представляющую разделитель строк в текстовых окнах (строка 14). Символа «\n» в данном случае недостаточно.~~

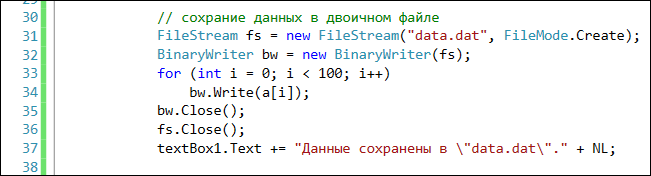
~~~~

**~~Шаг 5~~**~~. В обработчике события Click первой кнопки запишите операторы, создаю-щие два файла, содержащие один и тот же набор из 100 случайных чисел, равномерно расположенных в диапазоне от 100000 до 999999 (шестизначные натуральные числа).~~

~~~~

* ~~начале создаётся в куче массив~~ **~~a~~** ~~из 100 элементов целого типа. Далее в цикле~~ **~~for~~** ~~генерируются эти числа, по очереди сохраняя их в массиве и накапливая их сумму в переменной~~ **~~sum~~**~~. В конце формируется строка с информацией о созданных числах и осуществляется ее вывод в textBox1.~~

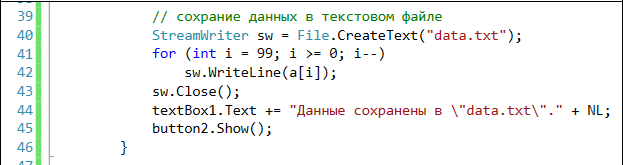
**~~Шаг 7~~**~~. Организуем запись созданных чисел в бинарный файл.~~

~~~~

* ~~начале создаём файл~~ **~~«data.dat»~~** ~~как поток двоичных данных и сохраняем ссылку на объект типа~~ **~~FileStrem~~** ~~для доступа к этому файлу (потоку данных) в переменную~~ **~~fs~~**~~. Создаём объект типа~~ **~~BinaryWriter~~** ~~для записи двоичных данных в этот файл (перемен-ная~~ **~~bw~~**~~). В цикле с помощью~~ **~~bw~~** ~~сохраняем в созданном файле по очереди все числа массива~~ **~~a~~**~~. В конце (обязательно!) закрываем доступ к обоим объектам (освобождаем ресурсы Windows, обеспечивающие работу с файлом).~~
  + ~~тексту в textbox дописываем (+=) ещё одну строку о проделанной работе.~~

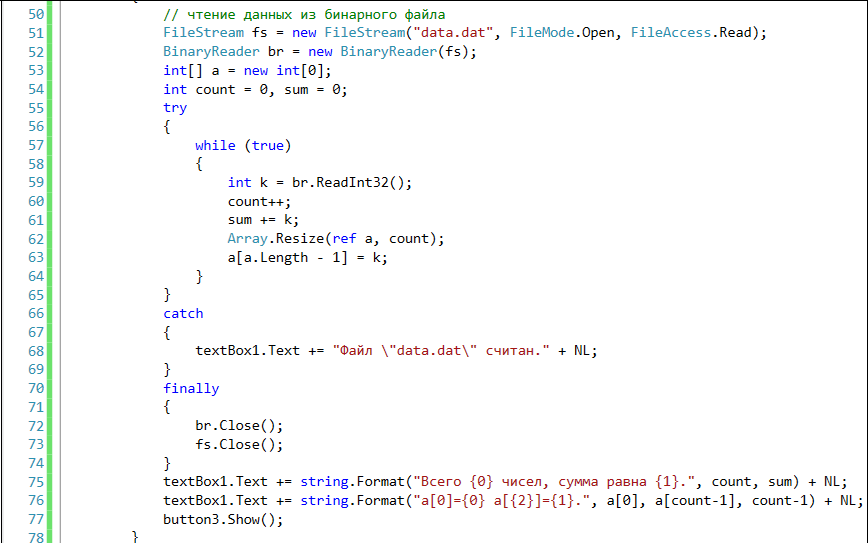
**~~Шаг 8~~**~~. Во многом аналогичные действия организуем для сохранения данных в тек-стовый файл~~ **~~«data.txt»~~**~~. В этом случае дстаточно толь одного объекта типа~~ **~~Stream-~~**

**~~Writer.~~**

~~~~

~~В конце метода сделаем видимой вторую кнопку.~~

**~~Шаг 9~~**~~. В В обработчике события Click второй кнопки реализуем чтение бинарных данных.~~

~~~~

~~Создаём поток типа~~ **~~FileStream~~** ~~уже не для создания файла, а для его открытия для чтения (строка 51), а читать данные будет объект типа~~ **~~BinaryReader~~**~~.~~

~~Само чтение организуем в защищенном блоке~~ **~~try~~** ~~–~~ **~~catch~~** ~~–~~ **~~finally~~**~~. В разделе~~ **~~try~~** ~~записываются операторы, в которых возможно исключение (в нашем случае в беско-нечном цикле считываем целые числа из файла и попытка чтения данных после того когда будет считано 100-е число приведёт к исключению). После чтения очередного числа мы увеличиваем на 1 счётчик чисел~~ **~~count~~**~~, добавляем число к сумматору~~ **~~sum~~**~~, изменяем размер массива (изначально он равен нулю) и в последнем элементе массива сохраняем считанное число.~~

~~После генерации исключения (попытка чтения несуществующего 101-го числа), управление будет передано в блок~~ **~~catch~~**~~, где в textBox1 выводится текущая информа-ция. В конце работы с файлом необходимо закрыть оба объекта.~~

* ~~завершении метода выводим в textBox1 справочную информацию и показываем третью кнопку. Обратите внимание на метод string.Format(), который формирует тек-стовую строку по указанному шаблону, заполняя вставки типа «{1}» значениями сле-дующих за шаблоном параметров данного метода.~~

~~При преобразовании чисел в строку символов во вставках можно использовать фор-матирование, например~~

~~double number3 = 25.07;~~

~~string result3 = String.Format("{0:f1}", number3); Console.WriteLine(result2); // 25,1~~

~~или~~

~~long number = 19876543210;~~

~~Console.WriteLine(number.ToString("+# (###) ###-##-##"));// +1 (987)~~

~~654-32-10~~

~~В последних версиях языка C# можно использовать интерполяцию строк~~

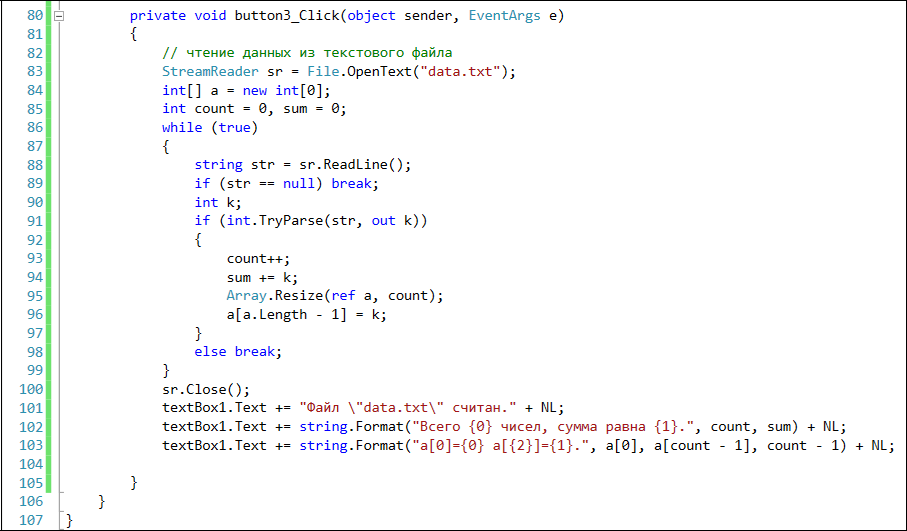
~~int x = 8;~~

~~int y = 7;~~

~~string result = $"{x} + {y} = {x + y}";~~

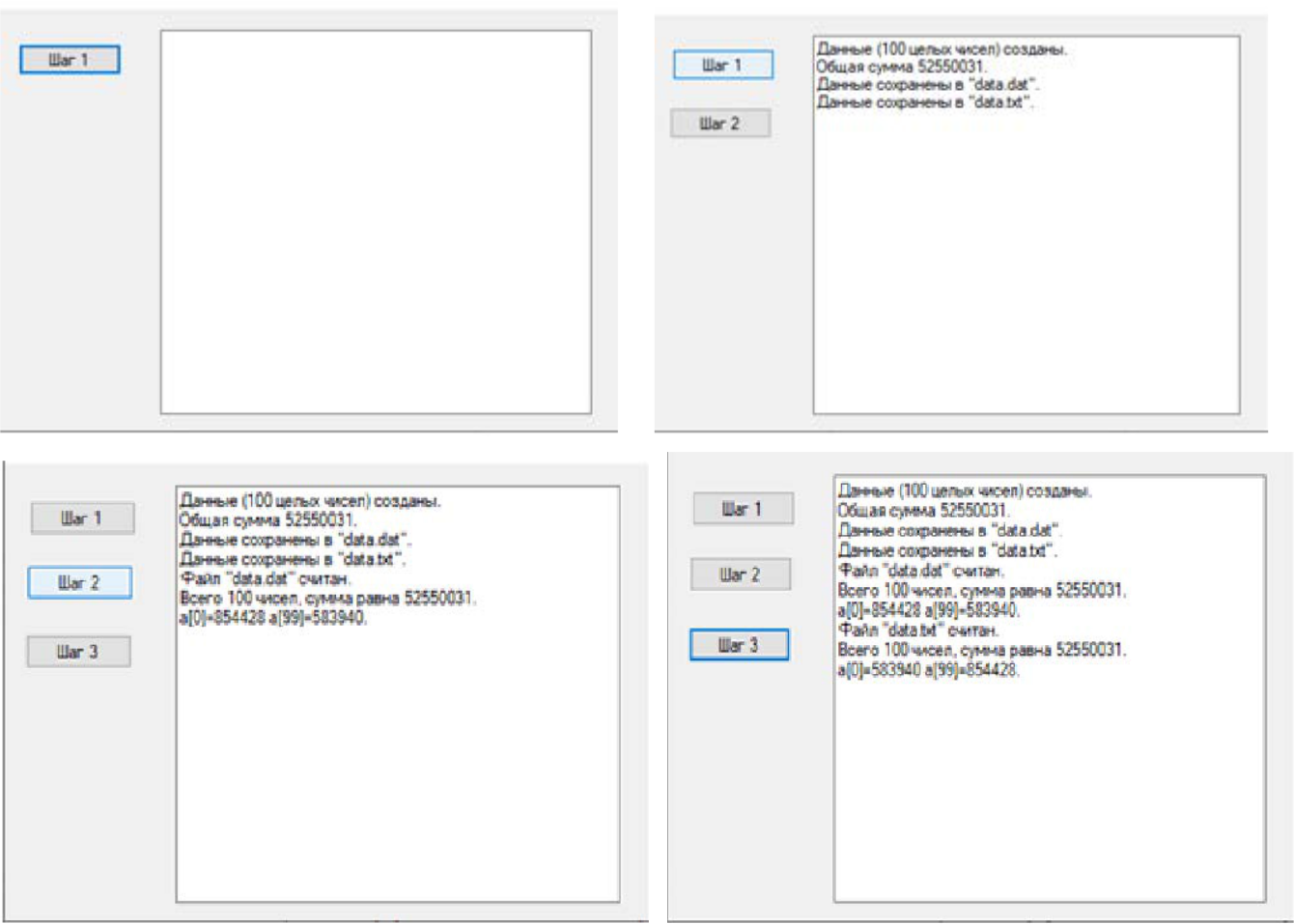
~~Console.WriteLine(result); // 8 + 7 = 15~~

**~~Шаг 10~~**~~. В обработчике события Click третьей кнопки аналогично считываем данные из текстового файла (разберитесь сами).~~

~~~~

* + ~~этом случае вместо защищённого блока в бесконечном цикле считываем текст по-строчно (в переменную str). Если мы считаем null (файл закончился), то выходим из цикла (break). Считанную строку пытаемся распознать как целое число (~~**~~int.tryParse~~**~~)~~
* ~~записать в целочисленную переменную (k). Если попытка будет удачной, число до-бавляется к массиву. Если неудачной – то завершаем цикл. В конце опять выводим справочную информацию.~~

**~~Шаг 11~~**~~. Запустим программу~~

~~~~

~~Как видно, количество и сумма записанных и считанных данных совпадают. Нуле-вой элемент в массиве данных текстового файла совпадает с последним элементом двоичного файла и наоборот.~~

~~Числа в текстовом файле хранятся как текст посимвольно. В конце строки записы-ваются два символа, которые означают в Windows переход к новой строке (то , что в шаге 4 мы получили как~~ **~~Environment.NewLine~~**~~). Фактически, файл~~ **~~«data.txt»~~** ~~содер-жит 100 строк, в которой записано по одному числу в обычной текстовой записи и 101 пустую строку.~~

~~Файл~~ **~~«data.dat»~~** ~~в два раза короче. Его длина 400 байт. В этом файле числа хранятся~~

* ~~том же формате, как в памяти (при записи такого файла в него просто копируется участок памяти, соответствующий сохраняемой переменной). Т.к. переменная типа int занимает в памяти 4 байта, то и получаем 400 байт для 100 переменных.~~
  + ~~бинарном файле (как и в памяти компьютера) между соседними числами нет ни-каких разделителей.~~

**~~Проект 2. «Простая обработка текстовых данных»~~**

~~Сформируйте папку из нескольких текстовых файлов разной длины (это могут быть списки студентов, тексты лекций, книг и т.д.). Один из файлов должен быть длины 8-~~

1. ~~строк, а один – больше 1000 строк.~~

~~Организуйте проект с одной формой на которой разместите три кнопки сверху в ряд и под ними друг под другом два больших многострочных компонента типа~~ **~~textbox~~**~~.~~

~~1-я кнопка предназначена для чтения текста из файла в массив строк. Для выбора файла из списка необходимо использовать компонент~~ **~~OpenDialog~~**~~. Считанный текст отобразить в верхний~~ **~~TextBox~~**~~.~~

~~2-я кнопка заменяет содержимое нижнего текстбокса на справочную информацию вида:~~

* ~~Имя считанного файла,~~
* ~~Количество строк~~
* ~~Количество непустых строк~~
* ~~Самая короткая непустая строка~~
* ~~Самая длинная непустая строка~~
* ~~Медиана – строка в середине текста.~~

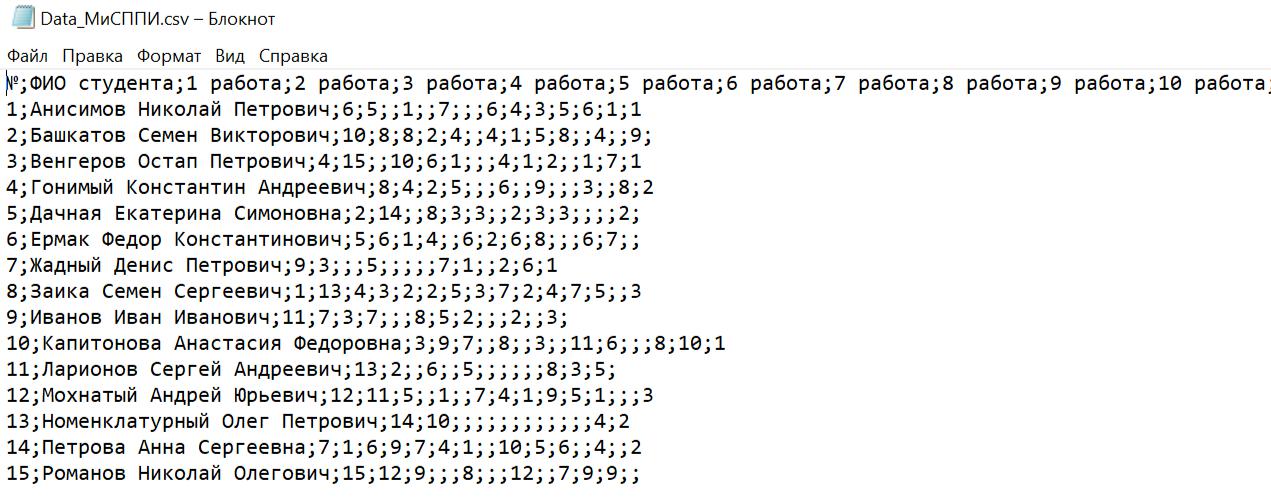
~~3-я кнопка создаёт массив всех слов текста по порядку. Словом будем считать непу-стой набор символов, не содержащий символов {<пробел>, <.>, <,>, <!>, <?>, <”>, <’>, <tab>, <enter>}, являющихся разделителями слов (для разбиения строки на слова ис-пользуйте метод~~ **~~Split~~** ~~класса~~ **~~String~~**~~). Содержимое нижнего~~ **~~TextBox~~** ~~заменяется спра-вочной информацией:~~

* ~~Имя считанного файла,~~
* ~~Количество слов~~
* ~~Длина самого длинного слова~~
* ~~Первое и последнее слово текста имеющих самую большую длину~~
* ~~Таблица из двух колонок. Первая – количество символов в слове по порядку от 1 до текущего максимума. Вторая – количество слов с указанной длиной в символах.~~

**~~Проект 3. «Обработка файлов формата CSV»~~**

~~Поместите в папку с рабочими файлами файл~~ **~~Data\_МиСППИ.csv~~** ~~из приложения к заданию.~~

~~Предварительно откройте его в текстовом редакторе и в EXCEL. Изучите его струк-туру, определите число полей и записей (первая строка – это названия полей данных).~~

~~~~

* ~~файле приведена информация о том, каким по порядку сдал очередное практиче-ское задание по дисциплине каждый из студентов некоторой группы. Внутри записи поля разделены символом «;». Последняя колонка «Бонусы» - это не название практи-ческого задания, а число набранных в семестре бонусных баллов.~~

~~Требуется создать приложение в формате Windows Forms, которое содержит ряд кнопок и многострочный TextBox достаточно большого размера. Функции кнопок (названия кнопок продумайте сами):~~

* **~~кнопка~~**~~:~~

~~А) Загружает данные из текстового файла~~ **~~Data\_МиСППИ.csv~~** ~~в память.~~

~~Б) Для каждого студента находит набранный им рейтинговый балл (за каждое сданную работу 5 баллов, дополнительно за каждое 1-е место в порядке сдачи оче-редной работы 0.5 балла, за 2-е место 0.4 и т.д до 0.1 балла за 5 место, места с 6 и ниже дополнительных баллов не даётся, плюс набранные бонусные баллы). Например, Петрова Анна должна получить: 50 баллов за 10 сданных заданий (10\*5=50), 1 балл за 2 первых места (2\*0.5=1), 0.4 балла за 2 четвертых места (2\*0.2=0.4), 0.1 балл за одно пятое место и 2 бонусных балла. Всего 53.5 баллов.~~

~~В) Формирует и сохраняет текстовый CSV файл~~ **~~Result\_ МиСППИ.csv~~**~~, дополняя строки исходного файла в конце значением рейтингового балла (не забудьте о пер-вой строке).~~

**~~2 кнопка~~**~~: выводит в TextBox содержимое исходного файла~~ **~~Data\_ МиСППИ.csv~~**

**~~3 кнопка~~**~~: выводит в TextBox содержимое созданного файла~~ **~~Result\_ МиСППИ.csv~~**

**~~4 кнопка~~**~~: выводит в TextBox список студентов в виде таблицы с разделителями (как на предыдущей странице), отсортированный по уменьшению рейтингового балла. Поля (графы) таблицы: порядковый номер, Фамилия и Имя студента, число сданных работ, рейтинговый балл (вещ. число с одним разрядом в десятичной части).~~

**~~Проект 4. «Работа с файлами других форматов»~~**

~~Задан текстовый файл, в строках которого через разделитель (определить какой разде-~~

~~литель используется самостоятельно) содержится информация согласно заданию (вариант задания определяется согласно списка студентов в группе).~~

~~Необходимо предусмотреть следующие операции:~~

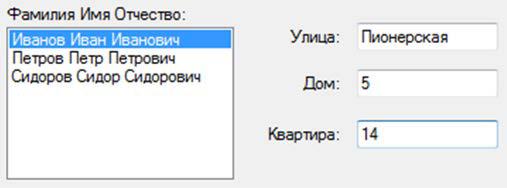
**~~1 кнопка~~**~~: возможность конвертации данных из текстового файла в xml или json файл.~~

**~~2 кнопка~~**~~: возможность загружать данные из xml или json файла следующим образом:~~

~~Пример: Текстовый файл содержит в строках (через пробел) следующую информацию:~~

~~Фамилия Имя Отчество Улица Дом Квартира~~

~~Форма для отображения данных из xml или json файла выглядит следующим образом:~~

~~~~

~~При выделении соответствующего элемента из ListBox должны отображаться данные соответствующей записи из xml или json файла.~~

**~~3 кнопка~~**~~: добавление новой записи в xml или json файл.~~

1. **~~кнопка~~**~~: редактирование записи в xml или json файле.~~
2. **~~кнопка~~**~~: удаление записи в xml или json файле.~~

~~Все изменения в файле должны автоматически отражаться на форме.~~

~~Добавление и редактирование записи реализовать через отдельную форму.~~

**~~6 кнопка~~**~~: формирование карточки выделенной записи в word, excel и pdf файлы.~~

~~Варианты для 4 задания:~~

**~~Задание 14.~~**

* ~~текстовом файле храниться следующая информация:~~

~~– Ф.И.О. пациента,~~

~~– дата рождения,~~

~~– дата посещения врача,~~

~~– диагноз,~~

~~– пол.~~

~~Определить количество пациентов, которым был поставлен указанный диагноз. Информацию выдавать в виде сообщения.~~