## Тема 10

## Создание приложений, работающих с потоковыми файлами

### 4.10.1. Основные понятия

Данные в приложениях VB можно не только вводить с клавиатуры и отображать на экране, но и читать из файла, где они хранятся, и записывать в файл.

***Файл* –** это поименованная совокупность любых данных, размещенная на внешнем запоминающем устройстве, хранимая, пересылаемая и обрабатываемая как единое целое. Файл может содержать программу, числовые данные, текст, закодированное изображение и пр. Физически файлы реализуются как участки памяти на [***внешних носителях***](http://www.tomsk.ru/Books/informatica/theory/chapter2/1_2_10.html)***,*** например, на магнитных дисках или CD-ROM. Каждый файл должен иметь имя***.***

К файловой системе имеет доступ любая прикладная программа, для этого во всех языках программирования, в том числе и в VB, имеются специальные средства в виде методы классов **.**NET Framework для работы с файлами.

По способу доступа к данным различают файлы ***последовательного доступа***, ***произвольного доступа*** и ***двоичные***. В файлах первого вида доступ к информации осуществляется последовательно от начала до конца файла, в файлах произвольного доступа – либо последовательно, либо в произвольном порядке, а в двоичных файлах – побайтно.

Текстовые файлы состоят из одной или более строк, слов или симво­лов. Обычные текстовые файлы определены в MS Windows как ***Текстовые документы*** и имеют расширение \*.txt.

Работа с любым файлом, в том числе с текстовыми файлами, состоит из трех этапов:

* открытие файлов;
* чтение или запись информации из файла или в файл;
* закрытие файла.

***Поток*** – это программный виртуальный канал для обмена данными между приложением и Windows или другим приложением. При взаимодействии приложения с Windows потоки, как правило, используются для работы с устройствами и представляют собой абстрактный уровень между приложением и устройством, что позволяет скрыть от приложения низкоуровневый механизм доступа к устройствам, упростить и унифицировать процесс написания программ, использующих ввод/вывод. Поток представляется программисту как логическое устройство приема-передачи информации. Потоки бывают входные и выходные. Данные, поступающие в поток, как правило, буферизуются, т.е. помещаются в специальный буфер данных, в котором они могут храниться. Передача данных из буфера получателю данных происходит при запросе от получателя или в случае заполнения буфера данных. Если получатель производит чтение данных, когда буфер потока пуст, то данная ситуация расценивается как достижение конца потока.

Поток для работы может быть отрыт в текстовом или бинарном режиме. С потоками можно производить следующие действия: открывать и закрывать потоки; посылать данные в поток и получать данные из потока; анализировать ошибки потокового ввода-вывода; управлять буферизацией потока и размером буфера; позиционировать указатель чтения-записи данных в потоке (если возможно). Все потоки и методы для работы с ними связаны с понятием файл.

Отличие потока от файла заключается в том, что поток представляет собой абстрактный уровень доступа программиста к физическому устройству, а файл представляет собой способ хранения информации на этом физическом устройстве. Операция открытия файла связывает поток с определенным файлом, операция закрытия разрывает эту связь. Каждый поток, связанный с файлом, имеет управляющую структуру типа **FILE**. Указатель на данную структуру, называемый также файловой переменной, является связующим звеном между потоком и файлом, работа с потоком, а значит, и со связанным с ним файлом, осуществляется через связанный с потоком указатель на файловую переменную.

Грубо обобщая, можно разделить функции ввода/вывода в каркасе .NET Framework на две широкие категории, не зависящих от устройства хранения данных – это запись и чтение.

Данные могут рассматриваться как поток байтов или символов. Данные также можно рассматривать как набор объектов.

Класс **Stream**(Поток, Абстрактный файл) – абстрактный класс, который является базовым для чтения и записи байтов в некоторое хранилище данных типа файла. Этот класс поддерживает синхронные и асинхронные чтение и запись. Асинхронные методы обсуждаются ниже в данной главе.

Класс**FileStream**, который является производным от класса**Stream,** предоставляет операции чтения и записи последовательности байтов в файл. Конструктор**FileStream**создает экземпляр потока. Перегруженные методы класса**Stream**(Поток, Абстрактный последовательный файл) осуществляют чтение и запись в файл.

Для загрузки в текстовое поле или строку содержимого текстового файла, а также для записи строковых данных в файл в VB можно использовать потоковый доступ:

* с использованием методов класса **System.IO** библиотеки   
  Framework.NET;
* с использованием объекта **My.Computer.System**.

При работе с файлами во время выполнения программы могут возникать различные непредвиденные ошибочные ситуации.

Рассмотрим средства VBдля ***обработки ошибок времени выполнения***.

***Ошибки времени выполнения*** (также называемые ***исключениями***) возникают при исключительных условиях работы - например, из-за отсутствия диска в дисководе или неправильного заданного пути при открытии файла. Блоки кода, обрабатывающие такие ошибки, называются структурными обработчиками ошибок (или структурными обработчиками исключений). Их можно использовать для распознавания ошибок времени исполнения при их возникновении в программе и настройки программы так, что она снова сможет получить управление и продолжить работу. Для этого имеется блок **Try…Catch…End Try** – синтаксическая конструкция VB для обработки ошибок.

Ошибка времени исполнения возникает тогда, когда VB пытается выполнить оператор, который по какой-то причине не может быть выполнен в том виде, в котором он написан. Это не значит, что VB недостаточно «умен», чтобы справиться с проблемой; это значит, что компилятору VB не сказали, что делать, если что-то пойдет не так.

Можно написать специальные процедуры VB, называемые ***структурными обработчиками ошибо*к**, которые будут реагировать на ошибки времени исполнения. Обработчик ошибок отслеживает ошибку времени исполнения и говорит программе, как продолжать работу при возникновении этой ошибки. Обработчики ошибок помещаются в процедуры событий там, где существует возможность возникновения проблемы, или в общие функции или подпрограммы, специально предназначенные для обработки ошибок. Как предполагает их название, обработчики ошибок обрабатывают ошибку с помощью оператора Try…Catch…End Try и специального объекта отслеживания ошибок – экземпляра класса **Exception**. Например, если ошибка времени исполнения произошла при загрузке файла с диска, обработчик ошибок может отобразить собственное сообщение об ошибке, которое укажет на проблему.

Оператор **Try** необходимо поместить в процедуру события непосредственно перед оператором, в котором может произойти непредвиденная ситуация, а блок **Catch** следует непосредственно за ним и содержит операторы, которые необходимо выполнить, если произойдет ошибка времени исполнения. Также можно использовать некоторые дополнительные операторы, такие, как **Finally**, а также вложенные блоки кода **Try…Catch…End Try**. Базовый синтаксис обработчика исключений **Try…Catch…End Try** может иметь следующий формат:

|  |
| --- |
| **Try**  *Операторы, которые могут вызвать ошибку времени исполнения*  **Catch**  *Операторы, которые выполняются,*  *если ошибка времени исполнения происходит*  **Finally**  *Дополнительные операторы,*  *выполняемые независимо от возникновения ошибки*  **End Try** |

где **Try, Catch** и **End Try** – это обязательные ключевые слова, а **Finally** и операторы, которые стоят за ним, необязательны.

В **Примерах 4.10.2-1** и **4.10.2-2** продемонстрирована ситуация возникновения ошибки времени выполнения, связанная с открытием файла.

### 4.10.2. Использование классов StreamReader, StreamWriter и объекта My.Computer.FileSystem для работы с текстовыми файлами

В последних версиях VSиспользуется эффективный ***потоковый доступ к файлам***, который является средством **.**NET Frameworkи определен в пространстве имен (библиотеке классов) **System.IO**.

***Поток* (Stream)** в VS– это абстрактный тип **Stream**, который представляет последовательность символов, передаваемых в среду или из среды для хранения информации. Потоки позволяют считывать и записывать данные на запоминающее устройство, которое в нашем случае соответствует файлу. Поэтому над потоками в основном выполняются ***операции чтения*** (**Read**), ***записи*** (**Write**)и ***поиска*** (**Seek**)***,*** которые реализованы в классе **System.IO.Stream*.*** Причем большинство потоковых объектов выполняют буферизацию данных, которая позволяет значительно повысить эффективность операций ввода/вывода. При этом данные потока не записываются сразу на диск, когда производится запись в файл, а помещаются в буфер и затем сбрасываются на диск, когда поток закрывается.

Таким образом, потоки класса **System.IO.Stream** осуществляют управление файлами на уровне отдельных байтов. Чтобы сделать операции чтения и записи более эффективными, в пространстве имен **System.IO** имеются средства для открытия и отображения текстовых файлов. Так, например, имеются классы **StreamReader** и **StreamWriter**, а также объект **My.Computer.FileSystem**.

|  |
| --- |
| *'Процедура чтения из текстового файла*  **Sub readfile1(ByVal filename As String, ByRef p As String)**  *'создание потока для чтения из текстового файла*  **Dim Rf As StreamReader**  **Rf = New StreamReader(filename)**  **Try**  **p = Rf.ReadToEnd()** *'чтение текста из файла в p*  **Rf.close()** *'закрытие потока*  **Catch ex As Exception**  **MsgBox("Невозможно прочитать данные из файла")**  **End Try**  **End Sub** |

*Рис. 4.10.2-1. Программный код процедуры* **readfile1()**

*чтения из текстового файла*

***Чтобы использовать классы* StreamReader *и* StreamWriter*, в начале программного кода проека следует добавить оператор доступа к этим классам:* Imports System.IO**.

|  |
| --- |
| *'Процедура построчного чтения данных из текстового файла,*  *'подсчета числа строк и формирования строки для вывода данных*  **Sub readfile2(ByVal filename As String, \_**  **ByRef n As Integer, ByRef p As String)**  ***'создание потока для чтения из текстового файл***  **Dim Rf As New StreamReader(filename)**  **Dim line As String**  **p = ""** *'строка с исходными данными*  **n = 0** *'количество строк в файле*  **Try**  **Do While Not Rf.EndOfStream()**  *'пока не достигнут конец потока*  **line = Rf.ReadLine( )***'чтение строки из файла в line*  **n = n + 1 : p = p & line & vbNewLine**  **Loop**  **Rf.Close() 'закрытие потока**  **Catch ex As Exception**  **MsgBox("Невозможно прочитать данные из файла")**  **End Try**  **End Sub** |

*Рис. 4.10.2-2. Программный код процедура* **readfile2()**

*построчного чтения данных из текстового файла и подсчета числа строк*

***Класс* StreamReader** из библиотеки .NET Framework позволяет откры­вать в программе и отображать текстовые файлы.На рис. 4.10.2-1 и   
рис. 4.10.2-2 представлены процедуры чтения содержимого всего текстового файла целиком в одну строку и построчного чтения данных из текстового файла. Обе эти процедуры имеют входной параметр **filename** (имя файла) и выходной параметр **p** (строка, содержащая значения, прочитанные из файла).

В примерах, представленных на рис. 4.10.2-1 и рис. 4.10.2-2, сначала создаются объекты **Rf** класса **StreamReader** – потоки открываются для чтения. Далее применяются методы созданных объектов: в примере на рис.4.10.3-1 метод **ReadToEnd()** считывает всю информацию из файла в строковую переменную **p**, начиная с текущей позиции (в данном примере с начала файла) и до конца файла; в примере на рис. 4.10.2-2 метод **ReadLine()** считывает из потока **Rf** только одну строку символов и присваивает ее содержимое строковой переменной **line**. Метод **ReadLine()** здесь применяется в цикле, пока не будет достигнут конец потока (**endOfStream**).

Метод **Close()** закрывает поток **Rf** и освобождает все ресурсы, которые были связаны с этим объектом.

***Класс* StreamWriter** из библиотеки классов позволяет создавать в программе новые текстовые файлы и записывать в них текстовые данные, а также добавлять текстовые данные в существующие текстовые файлы.

На рис. 4.10.2-3 и рис. 4.10.2-4 представлены процедура записи всей строки в заново созданный текстовый файл, и процедура добавления строки в уже существующий файл. Обе процедуры в качестве входных параметров принимают **filename** и **p–**строка, содержимое которой записывается в файл.

|  |
| --- |
| *' Процедура записи строки в заново созданный текстовый файл*  **Sub writefile1(ByVal filename As String, ByVal p As String)**  *'проверка существования текстового файла с таким же именем*  **If File.Exists(filename)Then**  **MsgBox("Файл "&filename " уже существует")**  **Exit Sub**  **End If**  *'создание нового текст. файла и потока wf для записи в файл*  *'1 способ Dim wf As StreamWriter = File.CreateText(filename)*  *'2 способ*  **Dim wf As New StreamWriter(filename)**  **wf.WriteLine(p)** *'запись строки в файл*  **wf.Flush()** *'очистка буфера потока*  **wf.Close() *'закрытие потока***  **End Sub** |

*Рис. 4.10.2-3. Программный код процедура* **writefile1()**

*записи строки в заново созданный текстовый файл*

|  |
| --- |
| *'Процедура добавления строки в уже существующий текстовый файл*  *' Если такого файла нет, то он создается*  **Sub writefile2(ByVal filename As String, ByVal p As String)**  *'проверка существования текстового файла с таким же именем*  **If File.Exists(filename) = False Then**  **MsgBox("Файл " & filename & " не существует")**  **End If**  *'1)Создание потока wf для добавления строки в конец файла*  *'Dim wf As StreamWriter = File.AppendText(filename)*  *'2)Значение True 2-го параметра означает, что будет*  *' производиться дозапись в конец файла*  **Dim wf As New StreamWriter(filename, True)**  **wf.WriteLine(p)** *'запись строки в файл*  **wf.Flush() *'очистка буфера потока+***  **wf.Close()** *'закрытие потока*  **End Sub** |

*Рис. 4.10.2-4. Программный код процедура* **writefile2()**

*добавления строки в уже существующий текстовый файл*

При создании потока **wf** класса **StreamWriter** вторым способом может использоваться второй аргумент, который определяет способ записи данных в файл. Если этот аргумент равен **True**, то строка **p** добавляется в конец файла к уже существующему содержимому. Если второй аргумент равен **False** или отсутствует, то все содержимое файла (если оно существует) полностью стирается, и с начала файла записывается строка **p**.

Таблица 4.10.2-1

|  |  |
| --- | --- |
| **Объект** | **Описание** |
| **My.Application** | Информация, связанная с текущим приложением, включая заголовок, директории, номер версии. |
| **My.Computer** | Информация об оборудовании, программном обеспечении и файлах, имеющихся на текущем (локальном) компьютере. **My.Computer** включает **My.Computer.FileSystem**, который ис­пользуется для открытия текстовых файлов и зашифрованных файлов системы. |
| **My.Forms** | Информация о формах текущего проекта VS. |
| **My.User** | Информация о текущем активном пользователе из **My.Computer**. |
| **My.WebServices** | Информация о веб-службах, активных в **My.Computer**, и ме­ханизм для доступа к новым веб-службам. |

***Пространство имен* My** *– это* средство быстрого доступа, которое упрощает доступ к .NET Framework при выполнении часто встречающихся задач, та­ких, как операции с формами, исследование содержимого компьютера и файловой системы, отображение информации о текущих приложениях или пользователях. С точки зрения пространства имен **My** выглядит примерно, как еще одна библиотека классов, которая дублирует в той или иной форме функциональность, уже присутствующую в .NET Framework**.**

Пространство имен **My** по функциональности разделено на несколько категорий, которые приведены в табл. 4.10.3-1.

Кроме того, пространство имен **My** *–* это просто способ быстрого доступа к элементам .NET Framework, и все его возможности можно увидеть в окне **Программный код** при помощи технологии ***IntelliSense***.

Например, чтобы отобразить окно сообщения, выводящее имя текущего компьютера и имя текущего пользователя, необходимо ввести:

|  |
| --- |
| **MsgBox(My.User.Name)** |

Эта строка осуществляет вывод, соответствующий рис. 4.10.2-5.



*Рис. 4.10.2-5. Результат выполнения функции* **MsgBox(My.User.Name)**

Объект **My.Computer** может отображать различную информацию о компьютере и его файлах. Например, следующий оператор отображает текущее системное время (локальная дата и время), определяемое ком­пьютером:

|  |
| --- |
| **MsgBox(My.Computer. Clock. LocalTime)** |

Чтобы открыть существующий текстовый файл и вывести его содержимое в объект текстового поля, можно использовать объект **My.Computer.FileSystem** и метод **ReadAllText( )**.Программный код будет иметь вид, как на рис. 4.10.3-6.

Метод **ReadAllText()** копирует все содержимое указанного текстового файла в строковую переменную или объект (в данном случае, переменную с именем **Str**)*,* что с точки зрения производительно­сти и времени написания кода быстрее, чем чтение файла по одной строке с помощью функции   
**LineInput().**

|  |
| --- |
| *'Процедура чтения данных из текстового файла*  *'целиком в одну строку и вывод данных в TextBox*  **Sub vvodFile(ByVal filename As String, \_**  **ByRef Str As String, ByRef T As TextBox)**  *'Открываем существующий файл и загружаем его текстовое*  *'содержимое целиком в одну строку Str*  **Try**  **Str = My.Computer.FileSystem.ReadAllText(filename)**  **T.Text = Str**  **Catch ex As Exception**  **MsgBox("ошибка при чтении файла")**  **End Try**  **End Sub** |

*Рис. 4.10.2-6. Программный код процедура* **vvodFile()**

*чтения данных из текстового файла целиком в одну строку*

Метод **ReadAllText()** копирует все содержимое указанного текстового файла в строковую переменную или объект (в данном случае в переменную с именем **Str**)*,* что с точки зрения производительно­сти и времени написания кода быстрее, чем чтение файла по одной строке с помощью функции   
**LineInput().**

Благодаря этому фактору объект **My.Computer.FileSystem** предоставляет великолепный способ быстро запрограммировать часто встречающиеся задачи. Относительно этой новой функции и возможных способов ее применения важно отметить, что здесь объект **My.Computer.FileSystem** полезен потому, что читается весь текстовый файл. Однако функция **LineInput( )** и класс **StreamReader** предлагают боль­ше возможностей, чем реализация объекта **My***,* и особенно это касается возможности обрабатывать файлы построчно. Эта возможность является очень важной при выполнении сортиров­ки и лексического разбора. Так что лучше всего знать все три метода открытия и создания текстовых файлов. Какой из них будет использоваться при программировании, зависит от ре­шаемой задачи и того, как ваш код будет использоваться.

Чтобы создать новый или добавить данные в существующий текстовый файл, можно использовать объект **My.Computer.FileSystem** и метод   
**WriteAllText( )**.Программный код будет иметь вид, как на рис. 4.10.2-7.

В методе **WriteAllText()** третий аргумент определяет способ записи данных. Если указать **True**, то данные добавляются в конец, существующих данных файла; если же указать **False**, то в случае существования данных в файле они полностью заменяются новыми данными (файл перезаписывается).

|  |
| --- |
| *'Процедура записи строки в текстовый файл*  **Sub vivodFile(ByVal filename As String, ByRef Str3 As String)**  *'Запись строки-результата Str3 в текстовый файл; (False)*  *'параметра задает,что файл будет перезаписан.*  **Try**  **My.Computer.FileSystem.WriteAllText(filename, Str3, False)**  **Catch ex As Exception**  **MsgBox("ошибка при сохранении (создании) файла")**  **End Try**  **End Sub** |

*Рис. 4.10.2-7. Программный код процедуры* **vivodFile()**

*записи строки в текстовый файл*

### 4.10.3 Примеры создания проектов (приложений),

### работающих с потоковыми файлами

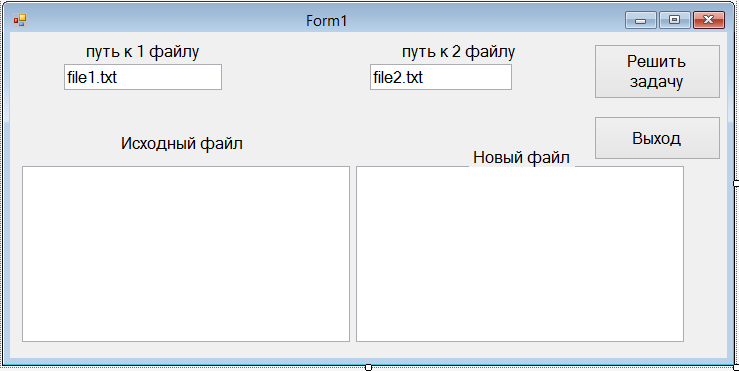
Пример 4.10.3-1. Дан текстовый файл, каждая строка которого содержит только слова из русских букв, разделенных одним или несколькими пробелами. Разработать программный код трех процедур:

* процедуры-функции с необходимыми формальными параметрами, которая получает строку и переставляет ее символы в обратном порядке;
* процедуры-подпрограммы с необходимыми формальными параметрами, которая построчно считывает текст из исходного файла с помощью потокового доступа и построчно записывает в новый файл преобразованные строки исходного файла;
* событийной процедуры, в которой осуществляется ввод имен файлов и вызов разработанных ранее процедур и функций.

Форма разрабатываемого приложения представлена на рис. 4.10.3-1.

На первом этапе необходимо создать процедуру-функцию **Perevorot()**, которая, получив через параметр строку **s**, с помощью регулярного цикла с отрицательным шагом записывает в новую строку temp все символы исходной строки **s** и возвращает созданную строку **temp** в качестве результата.

На втором этапе необходимо создать процедуру-подпрограмму **Resh()**, которой через параметры передаются две строки, с именами исходного и нового файлов. В **Resh()** необходимо создать два потоковых объекта: **Rf** – типа StreamReader, служащий для чтения исходного файла, и **wf** – типа **StreamWriter**, необходимый для записи в новый файл.



*Рис. 4.10.3-1. Форма проекта* **Пример 10.3-1**

Так как запись в новый файл должна производиться построчно, то второй параметр потока **wf** должен быть равным True, который означает, что при каждом использовании метода **WriteLine()** будет производиться добавление строки в существующий файл.

Далее в цикле считывается очередная строка исходного файла, и из нее с помощью процедуры **DelProb()** удаляются лишние пробелы и создается массив слов **Mas()**. Затем во внутреннем цикле каждое слово из массива Mas() переворачивается наоборот с использованием функции **Perevorot().**

После завершения внутреннего цикла полученный массив перевернутых слов рассматриваемой строки с помощью метода Join() склеивается через один пробел в одну новую строку **line2**, которая и записывается в новый файл.

Программный код приложения представлен на рис. 4.10.3-2.

|  |
| --- |
| *'Дан текстовый файл.*  *'Слова в строках файла разделены одним или несколькими пробелами.*  *'В каждой строке файла перевернуть слова наоборот и*  *'записать в новый текстовый файл через один пробел*  *'Потоковый способ построчно*  **Option Strict On**  **Option Explicit On**  **Imports System.IO**  **Public Class Form1**  *'функция переворачивает строку наоборот*  **Function Perevorot(ByVal s As String) As String**  **Dim temp As String = ""**  **Dim L As Integer = s.Length**  **For i As Integer = L - 1 To 0 Step -1**  **temp = temp + s.Substring(i, 1)**  **Next**  **Return temp**  **End Function**  *'процедура удаления из строки лишних пробелов*  *'чтобы слова разделялись только одним пробелом*  **Sub DelProb(ByRef s As String)**  **s = s.Trim**  **Do While s.IndexOf(Space(2)) >= 0**  **s = s.Replace(Space(2), Space(1))**  **Loop**  **End Sub**      **Sub Resh(ByVal filename1 As String, \_**  **ByVal filename2 As String,\_**  **ByRef Lb1 As ListBox, ByRef Lb2 As ListBox)**  *'создание потока для чтения из исходного текстового файла*  **Dim Rf As New StreamReader(filename1)**  *'создание потока для записи в другой текстовый файл*  *'Значение True 2-го параметра означает, что будет*  *'производиться дозапись в конец файла*  **Dim wf As New StreamWriter(filename2, True)**  **Dim line1, line2, slovo As String**  **Dim Mas() As String**  **Try**  **Do While Not Rf.EndOfStream()**  *'пока не достигнут конец потока*  **line1 = Rf.ReadLine()**  *'чтение строки из 1 файла в line1*  **Lb1.Items.Add(line1)**  *'вывод прочитанной строки line1 в ListBox*  **DelProb(line1)**  *'удаление из строки line1 лишних пробелов*  **Mas = Split(line1, " ")**  *'разбиваем строку line1 на массив слов*  **line2 = ""** *'новая строка для записи в другой файл*  **For i As Integer = 0 To Mas.Length - 1**  **slovo = Mas(i)***'один элемент массива - это слово*  *'перевернули слово и снова записали в массив*  **Mas(i) = Perevorot(slovo)**  **Next**  *'склеили массив перевернутых слов в строку line2,*  *'разделив слова одним пробелом*  **line2 = Join(Mas, " ")**  **wf.WriteLine(line2)** *'запись строки line2 во 2 файл*  **Lb2.Items.Add(line2)***'вывод строки line2 в ListBox*  **Loop**  **Catch ex As Exception**  **MsgBox("Невозможно прочитать данные из файла")**  **Finally**  **wf.Flush()** *'очистка буфера потока*  **Rf.Close()** *'закрытие потока*  **wf.Close()**  **End Try**  **End Sub**  **Private Sub Button1\_Click(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) Handles Button1.Click**  **Dim filename1 As String = "", filename2 As String = ""**  **filename1 = TextBox1.Text**  **filename2 = TextBox2.Text**  **Resh(filename1, filename2, ListBox1, ListBox2)**  **End Sub**  **Private Sub Button2\_Click(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs)**  **End**  **End Sub**  **End Class** |

*Рис. 4.10.3-2. Программный код проекта* **Пример 10**

Пример 4.10.3-2. Задан текстовый файл, каждая строка которого содержит только слова из русских букв, разделенных одним или несколькими пробелами. Написать программный код четырех процедур:

* процедуры-функции Perevorot() с необходимыми параметрами, которая получает строку и переставляет ее символы в обратном порядке;
* процедуры-подпрограммы Reshenie(), с необходимыми параметрами, которая считывает текст из файла с помощью объекта My.Computer.FileSystem, и для каждой строки исходного файла переворачивает каждое слово наоборот с использованием процедуры-функции и записывает в новую строку через один пробел по отдельности перевернутые слова исходной строки;
* процедуры vivodFile() с необходимыми параметрами, которая полученный в п.2, результат с помощью предыдущей процедуры записывает в новый файл;
* событийную процедуру, в которой будет осуществляться ввод имен файлов, вызов составленных процедур и вывод результатов.

Программный код проекта представлен на рис. 4.10.3-3

Первое задание реализует процедура-функция **Perevorot()**, которая, получив через формальный параметр по значению строку **s**, с помощью регулярного цикла с отрицательным шагом записывает в новую строку **temp** все символы исходной строки **s** и возвращает созданную строку **temp** в качестве результата.

Второе задание реализует процедура **Reshenie()**, которой через формальные параметры передается строка **filename**, содержащая имя исходного файла, и по ссылке возвращаются строка **s1** с исходным файлом и строка **s2** с перевернутыми словами, которую надо будет записать в новый файл. В ней сначала с помощью объекта**My.Computer.FileSystem** весь исходный файл считывается целиком в одну строку **s1**, затем полученная строка с помощью метода **Split()** разбивается на массив строк **Stroki()** по признаку конца строки (**vbNewLine**). Далее во внешнем цикле из каждой строки исходного файла с помощью процедуры **DelProb()** удаляются лишние пробелы, и строка разбивается на массив слов **slova()** по признаку одного пробела. Во внутреннем цикле каждое слово переворачивается наоборот с использованием процедуры-функции **Perevorot()** и записывается в новую строку **s2** через один пробел. После завершения внутреннего цикла в новую строку **s2** добавляется признак конца строки (**vbNewLine**) для разделения строк в новом файле, и далее повторение внешнего цикла для обработки очередной строки исходного файла.

Третье задание реализует процедура **vivodFile()**, которой через параметры передается строка **filename**, содержащая имя нового файла, и строка **S** с перевернутыми словами для записи в новый файл.

|  |
| --- |
| **Option Strict On**  **Option Explicit On**  **Public Class Form1**  *'функция переворачивает строку наоборот*  **Function Perevorot(ByVal s As String) As String**  **Dim temp As String = ""**  **Dim L As Integer = s.Length**  **For i As Integer = L - 1 To 0 Step -1**  **temp = temp + s.Substring(i, 1)**  **Next**  **Return temp**  **End Function**  *'процедура удаления из строки лишних пробелов*  *'чтобы слова разделялись только одним пробелом*  **Sub DelProb(ByRef s As String)**  **s = s.Trim**  **Do While s.IndexOf(Space(2)) >= 0**  **s = s.Replace(Space(2), Space(1))**  **Loop**  **End Sub**    *'Процедура чтения данных из текстового файла целиком в строку*  *'s1 и формирования новой строки s2 для записи в новый файл*  **Sub Reshenie(ByVal filename As String, ByRef s1 As String, \_**  **ByRef s2 As String)**  **Dim line, OdnoSlovo As String**  **Dim Stroki() As String** *'массив строк исходного файла*  **Dim slova() As String** *'массив слов одной строки исх.файла*  **s2 = ""** *'новая строка с перевернутым файлом*  *'Открытие существующего файла и загрузка его текстовое*  *'содержимое целиком в одну строку s1*  **Try**  **s1 = My.Computer.FileSystem.ReadAllText(filename)**  **Stroki = s1.Split(CChar(vbNewLine))**  *'массив строк исходного файла*  **For i As Integer = 0 To Stroki.Length - 1**  **line = Stroki(i)** *'одна строка исходного файла*  **DelProb(line)** *'удалили лишние пробелы*  **slova = line.Split(CChar(" "))**  *'разбили ее на массив слов*  **For j As Integer = 0 To slova.Length - 1**  **OdnoSlovo = Perevorot(slova(j))**  **s2 = s2 + OdnoSlovo + " "**  **Next**  **s2 = s2 + vbNewLine**  **Next**  **Catch ex As Exception**  **MsgBox("Файл не может быть открыт из-за " & \_**  **" следующей ошибки:" & vbNewLine & vbCrLf & \_**  **ex.Message)**  **End Try**  **End Sub**  *'Процедура записи строки в текстовый файл*  **Sub vivodFile(ByVal filename As String, ByVal S As String)**  *'Запись строки-результата S в текстовый файл (False)*  **Try**  **My.Computer.FileSystem.WriteAllText(filename, S, False)**  **Catch ex As Exception**  **MsgBox("ошибка при сохранении (создании) файла")**  **MsgBox("Данные в файл не могут быть переписаны из-за" \_**  **& " следующей ошибки:" & vbNewLine & ex.Message)**  **End Try**  **End Sub**  **Private Sub Button1\_Click(...)**  **Dim filename1, filename2 As String**  **Dim OldStr, NewStr As String**  **filename1 = TextBox1.Text**  **filename2 = TextBox2.Text**  **Reshenie(filename1, OldStr, NewStr)**  **vivodFile(filename2, NewStr)**  **TextBox3.Text = OldStr**  **TextBox4.Text = NewStr**  **End Sub**  **End Class** |

*Рис. 4.10.3-3. Программный код проекта* **Пример 10**

### 4.10.6. Тестовые задания

1. Файл – это
2. поименованная совокупность данных, размещенных на ВЗУ и хранимая, пересылаемая и обрабатываемая как единое целое
3. поименованная совокупность данных, размещенных в памяти компьютера и хранимая, пересылаемая и обрабатываемая как единое целое
4. совокупность данных, размещенных на ВЗУ и хранимая, пересылаемая и обрабатываемая как единое целое
5. поименованная совокупность данных, размещенных на ВЗУ
6. Каталог файла – это
7. оглавление файла
8. директория файла
9. папка файла
10. нет правильного ответа
11. Текстовые файлы – это
12. последовательные файлы, состоящие из записей переменной длины и расположенные на внешнем носителе
13. файлы произвольного доступа, состоящие из записей переменной длины и расположенные на внешнем носителе
14. последовательные файлы, состоящие из записей постоянной длины и расположенные на внешнем носителе
15. последовательные файлы, состоящие из записей переменной длины и расположенные в памяти компьютера
16. Работа с файлом состоит из этапов
17. открытия, чтения или записи информации, закрытия
18. открытия, записи информации, закрытия
19. открытия, чтения, закрытия
20. открытия, , закрытия
21. нет правильного ответа
22. В Vb возможны следующие способа доступа к текстовым файлам
23. с использованием методов класса System.IO
24. с использованием объекта My.Comhuter.System
25. с использованием потоков
26. нет правильного ответа
27. Метод ReadLine( )
28. требует предварительного создания потока
29. не требует предварительного создания потока
30. требует предварительного открытия файла с помощью дескриптора
31. требует указания пути к текстовому файлу
32. Метод ReadToEndLine( )
33. требует предварительного создания потока
34. не требует предварительного создания потока
35. требует предварительного открытия файла с помощью дескриптора
36. Чтобы добавить строку в существующий файл, можно использовать оператор
37. Dim t As New StreamWriter("f[le1.txt", True)
38. Dim t As New StreamWriter("f[le1.txt", False)
39. Dim t As StreamWriter("f[le1.txt")
40. все ответы верны
41. Метод ReadAllText( ) применяется
42. только при работе с объектом My.Computer.FileSystem
43. при работе с потоковым вводом-выводом
44. при работе с дескриптором
45. Метод WriteAllText( )
46. может добавить данные в конец существующего файла или полностью перезаписать весь файл
47. можно только добавить данные в конец существующего файла
48. можно вывести все данные из файла на экран монитора
49. Ошибочной записью оператора является
50. StreamWriter.WriteAllText("Hello")
51. My.Computer.FileSystem.WriteAllText("file1.txt","Hello", False)
52. My.Computer.FileSystem.WriteAllText("file1.txt", "Hello", True)
53. нет верного ответа
54. Метод WriteAllText( )
55. имеет 3 аргумента
56. имеет 2 аргумента
57. имеет 1 аргумента
58. может иметь 1, 2 или 3 аргумента
59. нет верного ответа
60. Метод ReadAllText( )
61. имеет 3 аргумента
62. имеет 2 аргумента
63. имеет 1 аргумента
64. может иметь 1, 2 или 3 аргумента
65. нет верного ответа
66. Метод ReadToEnd( )считывает из файла
67. всю информацию (все строки символов файла) \*
68. очередной символ или null, если больше символов нет
69. только одну строку
70. код символа или 0, если больше символов нет
71. Метод ReadLine( )считывает из файла
72. всю информацию (все строки символов файла)
73. очередной символ или null, если больше символов нет
74. только одну строку символов\*
75. код символа или 0, если больше символов нет
76. При достижении конца символьного потока метод ReadLine() вернет значение
77. -1
78. 0
79. null
80. EndOfStream
81. Метод Length при обращении к потоку FileStream определит количество:
82. бит в потоке
83. +: байт в потоке
84. значений некоторого типа в потоке.
85. Если файл с именем filename.txt не существует, то при попытке выполнить следующую процедуру

|  |
| --- |
| **Sub readfile1(ByVal filename As String, ByRef p As String)**  **Dim Rf As StreamReader**  **Rf = New StreamReader(filename)**  **Try**  **p = Rf.ReadToEnd() : Rf.close()**  **Catch ex As Exception**  **MsgBox("Невозможно прочитать данные из файла")**  **End Try**  **End Sub** |

1. открывается файл с именем filename.txt и читаются все данные
2. появляется окно каталогов, в котором пользователь должен найти данный файл
3. генерируется исключение FileNotFoundException
4. генерируется исключение IOException
5. В результате выполнения процедуры

|  |
| --- |
| **Sub writefile1(ByVal filename As String, ByVal p As String)**  **If File.Exists(filename)Then**  **MsgBox("Файл " & filename & " уже существует")**  **Exit Sub**  **End If**  **Dim wf As New StreamWriter(filename)**  **wf.WriteLine(p): wf.Close() 'закрытие потока**  **End Sub** |

1. создается, открывается файл с именем filename.txt и происходит запись всей строки\*
2. появляется окно каталогов, в котором пользователь должен найти данный файл
3. генерируется исключение FileNotFoundException
4. генерируется исключение IOException
5. открывается файл filename.txt и происходит запись всей cтроки
6. В результате выполнения процедуры записи строки, при условии что файл уже существует

|  |
| --- |
| **Sub writefile1(ByVal filename As String, ByVal p As String)**  **If File.Exists(filename)Then**  **MsgBox("Файл " & filename & " уже существует")**  **Exit Sub**  **End If**  **Dim wf As StreamWriter= File.CreateText(filename)**  **wf.WriteLine(p) : wf.Close()**  **End Sub** |

1. создается, открывается файл filename.txt и происходит запись всей строки
2. появляется окно каталогов, в котором пользователь должен найти данный файл
3. генерируется исключение FileNotFoundException
4. генерируется исключение IOException
5. выводится сообщение, что файл уже существует\*

## 4.10.7. Лабораторная работа по теме «Создания приложений, работающие с потоковыми файлами»

**Цель работы:** овладение практическими навыками разработки и программирования вычислительного процесса, связанного с созданием текстовых файлов, обработкой, внесенной в них информации, получение дальнейших навыков по отладке и тестированию программ

### Вопросы, подлежащие изучению

1. Понятие файла. Особенности работы с текстовыми файлами.
2. Основные операторы VB для работы с текстовыми файлами.
3. Средства обработчика ошибок времени исполнения.
4. Классы и объекты VB для работы с текстовыми файлами.
5. **Использование классов** StreamReader, StreamWriter и объекта My.Computer.FileSystem **для работы с текстовыми файлами**.

### Общее задание на разработку проекта

1. ***Изучите вопросы создания приложений, работающие с потоковыми файлами*** *(Тема 10).*
2. ***Создайте приложение*** *с именем* ***Проект-10****.*
3. ***Выберите вариант задания*** *из таблицы по усмотрению преподавателя.*
4. ***Создайте с помощью стандартной программы Microsoft Windows «Блокнот» текстовый файл*** *из 5-7 строк с исходными данными, соответствующими варианту задания.*
5. ***Разработайте интерфейс пользователя****, предусмотрев в нем несколько кнопок: для ввода данных из текстового файла, для обработки данных, для вывода результата в текстовый файл.*
6. ***Напишите программный код процедуры*** *пользователя для обработки текстового файла, которая должна вызывать процедуру обработки заданной строки (см. вариант задания лабораторной работы № 9), а также процедур для чтения и записи данных в/из текстового файла. Обмен данными между пользовательскими процедурами должен осуществляться через параметры, без глобальных переменных.*
7. ***Разработайте проект****, с помощью которого можно осуществить ввод исходных данных из текстового файла и отобразить их на форме, а затем обработать в соответствии с* ***индивидуальным заданием.***
8. ***Допишите программный код проекта****, в котором событийные процедуры должны содержать только операторы вызова пользовательских процедур.*
9. ***Выполните приложение*** *и получите результат.*
10. ***Докажите правильность результата****.*

### Варианты индивидуальных заданий

|  |  |
| --- | --- |
| **1)** | В текстовом файле найти строку, содержащую наибольшее количество цифр, идущих подряд. |
| **2)** | Зашифровать каждую строку текстового файла, содержащего русские буквы, пробелы и знаки препинания: заменить каждую букву, непосредственно следующей за ней по алфавиту ("**я**" на "**а**"), строчную букву преобразовать в прописную, прописную - в строчную. |
| **3)** | В текстовом файле найти строку, содержащую наибольшее количество слов, начинающихся и заканчивающихся на одну и ту же букву |
| **4)** | В текстовом файле найти строки, в которых не было ни одной замены каждого третьего слова «мама» на слово «мамочка». |
| **5)** | Из каждой строки текстового файла удалить все символы, входящие в нее более одного раза и найти самую короткую строку - результат. |
| **6)** | Преобразовать каждую строку текстового файла, повторив в ней каждый символ, совпадающий с заданным и найти самую длинную строку - результат. |
| **7)** | Записать в новый файл те строки исходного текстового файла, из символов которых можно составить вашу фамилию. |
| **8)** | В каждой строке текстового файла определить максимальную длину подстроки, состоящей из заданного символа и найти их сумму. |
| **9)** | Исключить из каждой строки текстового файла группы символов, расположенных между круглыми скобками, включая сами скобки. Предполагается, что внутри каждой пары скобок нет других скобок. Найти самую длинную строку – результат. |
| **10)** | В текстовом файле найти строку, содержащую наименьшее количество слов четной длины, в которых нет буквы «а». |
| **11)** | Записать в новый файл те строки исходного текстового файла, в которых нет ни одного слова, содержащего ровно три буквы «а». |
| **12)** | В каждой строке текстового файла определить самое длинное слово нечетной длины и записать их в новый файл. |
| **13)** | В текстовом файле найти строку, содержащую самое короткое слово четной длины. |
| **14)** | Каждая строка текстового файла содержит дату в формате "**mm-dd-yyyy**", где **mm** - месяц, **dd** - день, **yyyy** - год. Записать в новый файл только те строки исходного текстового файла, которые содержат корректно заданные даты, изменив их формат на "**dd.mm.yy**", где **yy** - две последние цифры года. |
| **15)** | Из каждой строке текстового файла, не содержащей цифр, получить "сжатую" строку, заменив самую длинную подстроку вида "**XXX…XX**" на подстроку вида "**nX**", где **X** - повторяющийся символ, **n** - число его повторений и записать в новый файл только измененные строки. |
| **16)** | Записать в новый файл те строки исходного текстового файла, которые нельзя преобразовать в число. |
| **17)** | Из каждой строки текстового файла получить новую строку, записывая каждый символ исходной строки только один раз. |
| **18)** | В каждой строке текстового файла подсчитать сумму цифр и найти строку с наименьшей суммой. |
| **19)** | В текстовом файле, не содержащем букв русского алфавита, найти строку с максимальным процентом гласных букв (**a, e, i, o, u, y**). |
| **20)** | Записать в новый файл те строки исходного текстового файла, в которых самое длинное слово встречается раньше, чем самое короткое слово. |
| **21)** | Преобразовать каждую строку текстового файла, удалив из каждого слова нечетной длины средний символ, и найти строку с минимальным количеством удалений. |
| **22)** | В текстовом файле найти строку, содержащую наибольшее количество симметричных слов, т.е. читающихся одинаково слева направо и справа налево. |
| **23)** | Записать в новый файл те строки исходного текстового файла, которые являются записью вещественного числа с нулевой дробной частью. |
| **24)** | Для каждой строки текстового файла, содержащей корректно заданную дату в формате "**dd.mm.yy**", где **dd** - день, **mm** - месяц, **yy** - год, получить строку с датой, отстоящей от заданной на **n** дней вперед, в том же формате и записать в новый файл. |
| **25)** | Записать в новый файл те строки исходного текстового файла, символы которых расположены в алфавитном порядке. |
| **26)** | Записать в новый файл те строки исходного текстового файла, которые являются записью целого числа. |
| **27)** | Записать в новый файл те строки исходного текстового файла, в которых нет **n**-ого с конца вхождения заданного символа. |
| **28)** | Преобразовать каждую строку текстового файла, состоящую из "слов" длиной менее **10** символов и разделенных запятыми, заменив запятые группами пробелов так, чтобы каждое слово начиналось с позиции **10\*(k - 1) + 1**, где **k** - номер "слова". |
| **29** | Преобразовать каждую строку текстового файла, вставив в середину каждого слова четной длины букву «К» и найти строку с максимальным количеством вставок. |
| **30)** | Для каждой строки текстового файла, содержащей корректно заданную дату в формате "**dd.mm.yy**", где **dd** - день, **mm** - месяц, **yy** - год, получить строку с датой, отстоящей от заданной на **n** дней назад, в том же формате и записать в новый файл. |

### Содержание отчёта

1. Тема и название лабораторной работы;
2. Фамилия, имя студента, номер группы, номер варианта.
3. Задание на разработку проекта.
4. Формализация и уточнение задания.
5. Элементы, разрабатываемого проекта:
6. графический интерфейс пользователя;
7. таблица свойств объектов;
8. схема алгоритма решаемой задачи;
9. программный код проекта.
10. Результаты выполнения проектов.
11. Доказательство правильности работы программы.

### Первый пример выполнения задания

1. **Тема и название лабораторной работы:**

Работа с текстовыми файлами;

Чтение и запись информации из файла/в файл с помощью объекта **My.Computer.FileSystem**.

1. **Фамилия, имя студента, номер группы, номер варианта:**

Иванов И., БИН1405, вариант 13.

1. **Задание на разработку проекта:**

Создайте проект, который реализует ввод из файла строк **a** и **b**, отображает их на форме, а затем обрабатывает, удаляя из строки **a** все символы, входящие в строку **b** и расположенные на нечетных позициях строки **b**. Используйте разработанную для лабораторной работы 9 процедуру **Обработка()**.

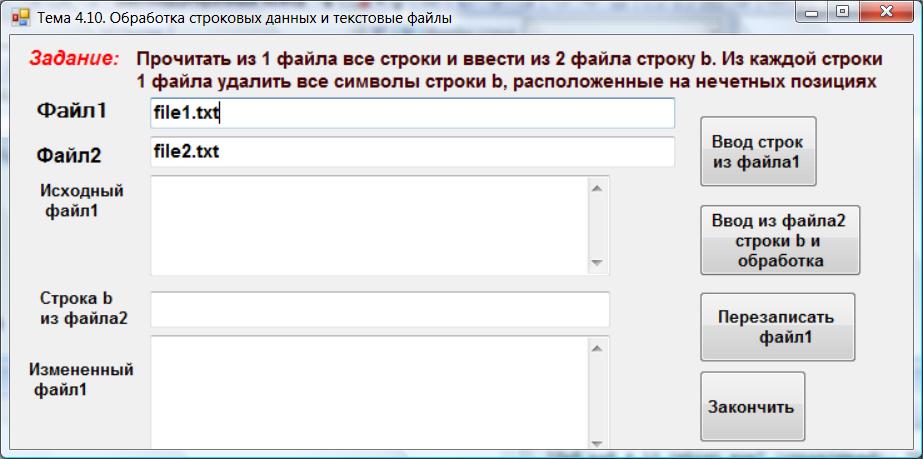
Результат обработки вышеприведенной задачи необходимо отобразить на форме и перезаписать в первый текстовый файл. Работу с текстовыми файлами осуществить с помощью объекта My.Computer.FileSystem.

1. **Формализация и уточнение задания:**

Формализация не требуется.

1. **Элементы, разрабатываемого проекта:**
2. Графический интерфейс пользователя:

Форма приложения представлена на рис. 4.10.7-1.



*Рис. 4.10.7-1. Форма проекта* **Проект 10-1**

1. Таблица свойств объектов:

Для вывода на форму текстового файла, состоящего из нескольких строк, используются текстовые поля **TextBox3** и **TextBox5**, для которых свойство Multiline должно быть установлено в состояние *True*. Кроме того, требуется установить для этих полей свойство ScrollBars в состояние *Vertical* для обеспечения возможности вертикальной прокрутки содержимого текстовых полей. Текстовое поле **TextBox4**, предназначенное для вывода на форму только одной строки из второго файла может оставаться со значением свойства Multiline, равным *False*, так же, как и два других текстовых поля **TextBox1**и **TextBox2**, служащие для ввода путей к текстовым файлам.

Определите, установите и сведите в табл. свойства объектов

самостоятельно.

1. Схема алгоритма решаемой задачи:

Схемы отдельных алгоритмов представлены в Теме 4.10.

1. Программный код проекта:

Программный код проекта представлен на рис. 4.10.7-2.

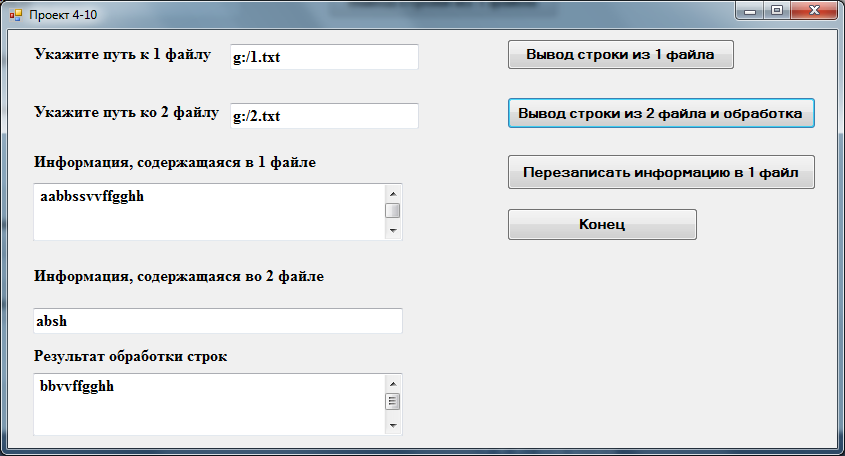
|  |
| --- |
| **Public Class Form1**  *'Глобальные строки*  **Dim Str1 As String** *'строка с исходным первым файлом*  **Dim Rez As String** *'строка с измененным первым файлом*  *' Процедура записи строки в файл.*  **Sub Запись(ByVal ПутьФ As String, ByVal str1 As String)**  **If (My.Computer.FileSystem.FileExists(ПутьФ)=True) Then**  **If (MsgBox("Файл существует. Переписать?", \_**  **MsgBoxStyle.YesNo Or MsgBoxStyle.Question, "Overwrite")\_**  **<> MsgBoxResult.Yes) Then Exit Sub**  **End If**  *' Сохраняем данные*  **Try**  **My.Computer.FileSystem.WriteAllText(ПутьФ, str1,False)**  **Catch ex As Exception**  **MsgBox("Файл не может быть записан на диск " & \_**  **"из-за ошибки:" & vbCrLf & vbCrLf & ex.Message)**  **End Try**  **End Sub**  *'Процедура чтения данных из файла целиком в одну строку*  **Sub Чтение(ByRef Str3 As String, ByVal filename As String)**  *' Открываем существующий файл и загружаем его в одну строку*  **Try**  **Str3 = My.Computer.FileSystem.ReadAllText(filename)**  **Catch ex As Exception**  **MsgBox("Файл не может быть открыт " & \_**  **" из-за ошибки:" & vbCrLf & vbCrLf & ex.Message)**  **End Try**  **End Sub**      *'Процедура обработки одной строки (удаления символов из строки)*  **Sub Обработка(ByVal b As String, ByRef a As String)**  **Dim d, m, i, j As Short**  **m = CShort(a.Length) 'длина строки a**  **d = CShort(b.Length) 'длина строки b**  **For i = 0 To CShort(d - 1) Step 2**  **'перебор нечетных симв. строки b**  **j = 0**  **Do While j < m** *'перебор всех символов строки a*  **If a.Substring(j, 1) = b.Substring(i, 1) Then**  **a = a.Remove(j, 1)**  *'пока символы совпадают - удаляем из a*  **m = CShort(m - 1) 'уменьшаем длину строки a**  **Else**  **j = CShort(j + 1)**  *'переход к следующемусимволу строки a*  **End If**  **Loop**  **Next i**  **End Sub**  *'Процедура решения задачи удаления символов из строк файла*  **Sub Resh(ByVal b As String, ByVal s1 As String, \_**  **ByRef s2 As String)**  **Dim a As String**  **Dim Stroki() As String** *'массив строк первого файла*  **Stroki = Split(s1, vbNewLine)**  *'Разбиваем строку с первым файлом*  *' на массив строк по признаку конца строки*  **s2 = ""** *' новая строка для нового файла целиком*  **For i As Integer = 0 To Stroki.Length - 1**  **a = Stroki(i)** *' одна строка первого файла*  **Обработка(b, a)** *' удаление символов из строки*  *'собираем измененные строки первого файла в*  *'одну строку s2*  **s2 = s2 + a + vbNewLine**  **Next**  **End Sub**    **Function vvod(ByVal T As TextBox) As String**  **Return T.Text**  **End Function**  **Sub vivod(ByVal z As String, ByRef t As TextBox)**  **If z.Length <> 0 Then**  **t.Text = z**  **Else**  **t.Text = "Все удалено"**  **End If**  **End Sub**  **Private Sub Button1\_Click(sender As Object, \_**  **e As EventArgs) Handles Button1.Click**  **Dim ПутьФ1 As String = ""**  **ПутьФ1 = vvod(TextBox1)**  **Чтение(Str1, ПутьФ1) : vivod(Str1, TextBox3)**  **End Sub**  **Private Sub Button2\_Click(sender As Object, \_**  **e As EventArgs) Handles Button2.Click**  **Dim Str2 As String = ""**  **Dim ПутьФ2 As String = ""**  **ПутьФ2 = vvod(TextBox2) : Чтение(Str2, ПутьФ2)**  **vivod(Str2, TextBox4)**  **Resh(Str2, Str1, Rez) : vivod(Rez, TextBox5)**  **End Sub**    **Private Sub Button3\_Click(sender As Object, \_**  **e As EventArgs) Handles Button3.Click**  **Dim ПутьФ As String = ""**  **ПутьФ = vvod(TextBox1) : Запись(ПутьФ, Rez)**  **End Sub**  **Private Sub Button4\_Click(sender As Object, \_**  **e As EventArgs) Handles Button4.Click**  **End**  **End Sub**  **End Class** |

*Рис. 4.10.7-2. Программный код проекта* **Проект 10-1**

Программный код, решаемой задачи содержит процедуры **vvod(),   
vivod(), Обработка(), Resh(), Чтение()** и **Запись().** Так как разработанный интерфейс имеет не одну, а несколько кнопок, которые работают со строками, содержащими первый файл (**Str1** – для исходного файла и **Rez** – для измененного файла), то эти строки должны быть доступны нескольким событийным процедурам, поэтому в программном коде их необходимо сделать глобальными (видимыми) для всех процедур, и объявить в начале программы, до работы всех процедур.

1. **Результаты выполнения проектов.**

Результаты выполнения проекта приведены на рис. 4.10.7-3.



*Рис. 4.10.7-3. Результаты выполнения проекта* **Проект 10-1**

1. **Доказательство правильности работы программы.**

Дана исходная строка **a**=**"aabbssvvffgghh"**. В строке **b=** **"absh"** на нечетных местах стоят символы **a** и **s**. После их удаления строка, **а**=**"bbvvffgghh"**. Полученный вид строки полностью совпадает с результатом, полученным с использованием программы. В первый файл перезаписана информация, полученная в результате выполнения программы, т.е. строка **"bbvvffgghh"**, следовательно, результаты достоверны.

### Второй пример выполнения задания

1. **Тема и название лабораторной работы:**

Работа с текстовыми файлами.

Чтение и запись информации из файла/в файл с помощью классов

StreamReader, StreamWriter.

1. **Фамилия, имя студента, номер группы, номер варианта:**

Иванов И., БИН1405, вариант 13.

1. **Задание на разработку проекта:**

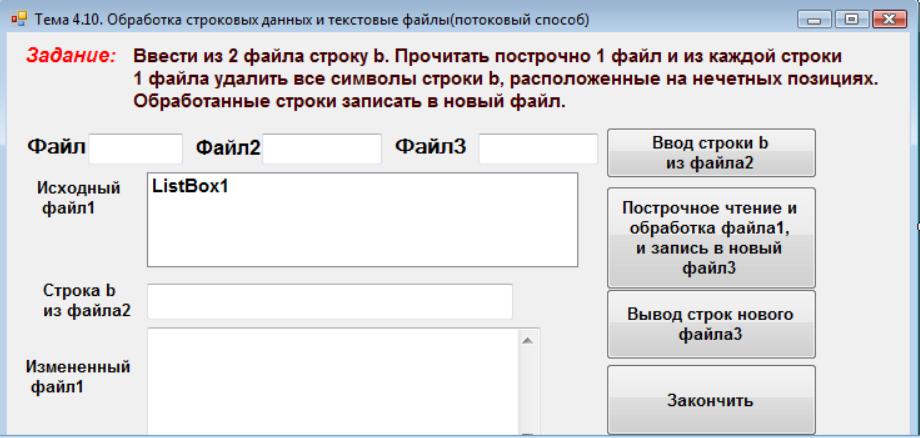
Создайте проект, который вводит из второго файла строку **b**, построчно вводит все строки первого файла и отображает их на форме, а затем обрабатывает, удаляя из каждой строки первого файла все символы, расположенные на нечетных позициях строки **b**, используя ранее разработанную для лабораторной работы 9 процедуру **Обработка()**. Результат обработки необходимо отобразить на форме и сохранить в новом (третьем) текстовом файле. Работу с текстовыми файлами осуществить с помощью классов **StreamReader, StreamWriter**.

1. **Формализация и уточнение задания:**

Формализация не требуется.

1. **Элементы, разрабатываемого проекта:**
   1. **Форма интерфейса приведена на рис. 4.10.7-4.**

Для построчного вывода на форму первого текстового файла, удобно использовать элемент управления **ListBox**. Для вывода на форму нового третьего текстового файла, состоящего из нескольких строк, которые будут прочитаны целиком в одну строку, используется текстовое поле **TextBox4**, для которого свойство Multiline обязательно должно быть установлено в состояние *True* и свойство ScrollBars в состояние *Vertical* для обеспечения возможности вертикальной прокрутки содержимого этого текстового поля. Текстовое поле **TextBox3** для вывода на форму только одной строки из второго файла может оставаться со значением свойства Multiline, равным *False*, также, как и три других текстовых поля **TextBox1** и **TextBox5** и **TextBox6** для ввода путей к текстовым файлам.



*Рис. 4.10.7-4. Форма проекта* **Проект 10-2**

* 1. **Таблица свойства объектов формы:**

Свойства объектов разработанной формы установите

самостоятельно.

* 1. **Схема алгоритма решаемой задачи:**

Схемы отдельных алгоритмов представлены в Теме 4.10.

* 1. **Программный код проекта:**

Программный код проекта представлен на рис. 4.10.7-5.

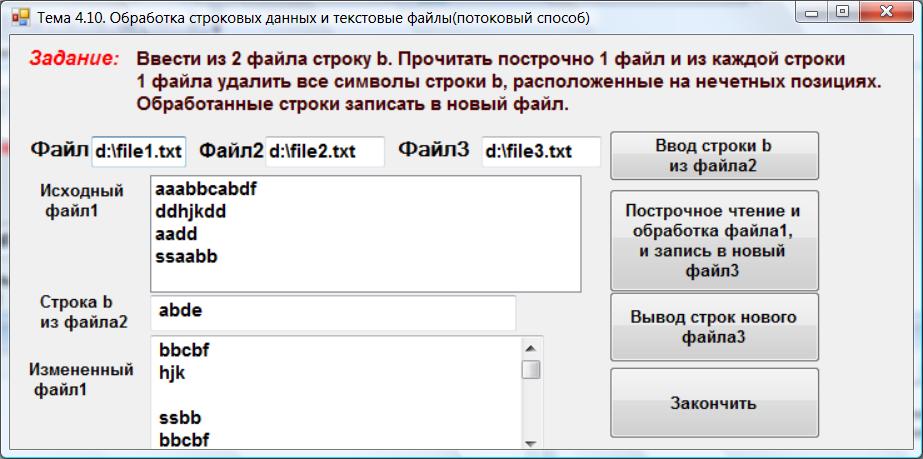
Процедура **Resh()** с помощью класса **StreamReader** построчно считывает содержимое 1 файла, удаляет из строки нужные символы процедурой **Обработка()** и дописывает обработанную строку в новый 3 файл с помощью класса **StreamWriter**. Процедура **readfile()** с помощью класса **StreamReader** считывает содержимое текстового файла целиком в одну строку. Так как разработанный интерфейс имеет не одну, а несколько кнопок, которые работают со строкой **b**, содержащей 2 файл, то эта строка должны быть доступна нескольким событийным процедурам, и поэтому в программном коде ее необходимо сделать глобальной (видимой для всех процедур), и объявить в начале программы, до описания процедур.

|  |
| --- |
| **Option Strict On**  **Option Explicit On**  **Imports System.IO**  **Public C lass Form1**  **Dim b As String** *'Глобальная строка b из второго файла*  *'Процедура построчного чтения из 1 файла и записи в 3 файл*  **Sub Resh(ByVal filename1 As String, \_**  **ByVal filename3 As String, \_**  **ByVal b As String, ByRef L As ListBox)**  *'создание потока для чтения из исходного файла1*  **Dim Rf As New StreamReader(filename1)**  *'создание потока для записи в другой текстовый файл*  *'Значение True 2-го параметра означает, что будет*  *'производиться дозапись в конец файла*  **Dim wf As New StreamWriter(filename3, True)**  **Dim a As String**  **Try**  **Do While Not Rf.EndOfStream()**  *'пока не достигнут конец потока*  **a = Rf.ReadLine()*'чтение строки а из 1 файла***  **L.Items.Add(a)** *'вывод прочитанной строки в ListBox*  **Обработка(b, a)***'удаление символов из строки a*  **wf.WriteLine(a)***'запись обработ. строки в новый файл*  **Loop**  **Catch ex As Exception**  **MsgBox("Невозможно прочитать данные из файла")**  **Finally**  **wf.Flush()** :**Rf.Close()** : **wf.Close()**  **End Try**  **End Sub**  *'Процедура чтения из текстового файла целиком в одну строку*  **Sub readfile(ByVal filename As String, \_**  **ByRef b As String)**  *'создание потока для чтения из текстового файла*  **Dim Rf As StreamReader**  **Rf = New StreamReader(filename)**  **Try**  **b = Rf.ReadToEnd()** *'чтение текста из файла в b*  **Rf.Close()** *'закрытие потока*  **Catch ex As Exception**  **MsgBox("Невозможно прочитать данные из файла")**  **End Try**  **End Sub**  *'Процедура обработки одной строки (удал. символов из строки)*  **Sub Обработка(ByVal b As String, ByRef a As String)**  **Dim d, m, i, j As Short**  **m = CShort(a.Length)** *'длина строки a*  **d = CShort(b.Length*) 'длина строки b***  **For i = 0 To CShort(d-1) Step 2*'перебор нечетных симв.стр. b***  **j = 0**  **Do While j < m** *'перебор всех символов строки a*  **If a.Substring(j, 1) = b.Substring(i, 1) Then**  **a = a.Remove(j, 1)'пока симв.совпад.-удаляем из a**  **m = CShort(m - 1) *'уменьшаем длину строки a***  **Else**  **j = CShort(j + 1)*'переход к след.символу строки a***  **End If**  **Loop**  **Next i**  **End Sub**  **Function vvod(ByVal T As TextBox) As String**  **Return T.Text**  **End Function**    **Sub vivod(ByVal z As String, ByRef t As TextBox)**  **If z.Length <> 0 Then**  **t.Text = z**  **Else**  **t.Text = "Все удалено"**  **End If**  **End Sub**  **'Кнопка чтения второй строки из второго файла**  **Private Sub Button1\_Click(ByVal sender As System.Object,\_**  **ByVal e As System.EventArgs) Handles Button1.Click**  **Dim ПутьФ2 As String = ""**  **ПутьФ2 = vvod(TextBox5)**  **readfile(ПутьФ2, b)**  **vivod(b, TextBox3)**  **End Sub**  *'Кнопка чтения 1 файла и запись обработанных строк*  *'в третий файл*  **Private Sub Button3\_Click(ByVal sender As System.Object,\_**  **ByVal e As System.EventArgs) Handles Button3.Click**  **Dim ПутьФ3 As String = ""**  **Dim ПутьФ1 As String = ""**  **ПутьФ1 = vvod(TextBox1)**  **ПутьФ3 = vvod(TextBox6)**  **Resh(ПутьФ1, ПутьФ3, b, ListBox1)**  **End Sub**  *'Кнопка чтения нового(третьего) файла*  *'и вывод его на форму*  **Private Sub Button2\_Click(ByVal sender As System.Object,\_**  **ByVal e As System.EventArgs) Handles Button2.Click**  **Dim ПутьФ3 As String = ""**  **ПутьФ3 = vvod(TextBox6)**  **Dim Rez As String = ""**  **readfile(ПутьФ3, Rez)**  **vivod(Rez, TextBox4)**  **End Sub**  **Private Sub Button6\_Click(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) Handles Button6.Click**  **End**  **End Sub**  **End Class** |

*Рис. П.1.10-5. Программный код проекта* **Проект 10-2**

1. **Результаты выполнения проектов:**

Результаты выполнения проекта приведены на рис. 4.10.7-6.



*Рис. 4.10.7-6. Результаты выполнения проекта* **Проект 10-2**

1. **Доказательство правильности работы программы:**

Из всех строк исходного первого файла удалены символы, стоящие на 1 и 3 позициях строки b из второго файла (т.е. удалились буквы a и d), причем из третьей строки удалены все символы и в измененный файл записана пустая строка.

### Контрольные вопросы

1. Что такое файл?
2. Какие классы используются для работы с файлами?
3. Каким образом можно обработать ошибки времени выполнения?
4. Что такое потоковый доступ к файлам?
5. Какая функция открывает, и, какая функция закрывает текстовый файл?
6. Какие функции осуществляют операции чтения и записи?