**Лабораторная работа № 13**

**«Разработка динамической библиотеки по работе с текстовыми файлами»**

**Цель работы:** получить практические навыки разработки динамической библиотеки по обработке текстовых файлов, обобщить и расширить знания по использованию стандартных интерфейсов .NET, разработки тест-кейсов.

**Ход работы:**

1. Создайте **динамическую библиотеку**, в которую поместите методы обработки текстового файла:

* открытие файла;
* сохранение файла;
* вывод файла на принтер;
* подсчёт количества слов в тексте (слова разделены одним или несколькими пробелами и знаками препинания);
* перегрузку операций сравнения двух файлов на равенство, на неравенство, причём равными считать файлы, у которых *одинаковое количество символов*, без учёта содержания;
* поиска файлов в указанном каталоге;
* сортировку списка найденных файлов разными способами: по именам файлов   
  (в алфавитном порядке), по количеству символов в файле, по количеству слов в файле;
* метод обработки файла по своему варианту.

1. Подготовьте **тест-кейсы** для тестирования метода подсчёта количества слов в файле, двух операций сравнения на ваш выбор и метода по вашему варианту. Протестируйте библиотеку.

*Пример тест-кейса:*

[TestMethod]

public void TestMethod\_OperationEqual()

{

MyFile f1 = new MyFile("1.txt", "Привет, мир!");

MyFile f2 = new MyFile("2.txt", "Здравствуй, мир!");

bool expected = false; // ожидаемое значение

bool actual = (f1 == f2);

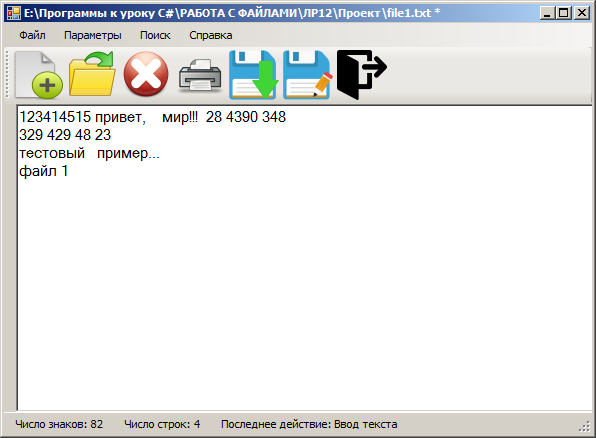
Assert.AreEqual(expected, actual, "Операция сравнения файлов на равенство выполнена не верно! ");

}

1. Разработайте **основной модуль** (проект, включающий форму). Пользовательский интерфейс должен включать: систему меню, панель инструментов, строку состояния.

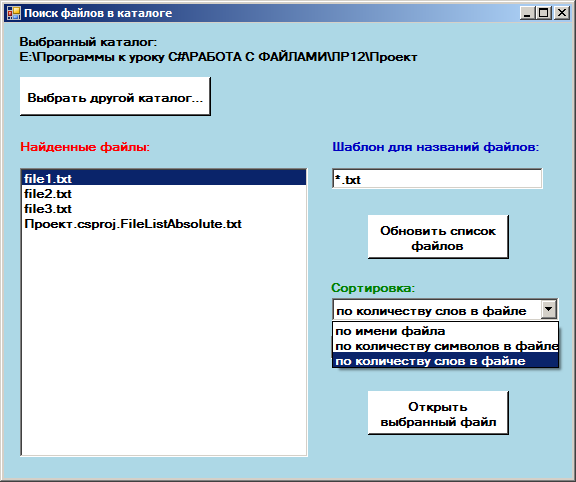
*Рекомендации по настройке главного меню:*

* Пункт меню **«Файл»** содержит следующие подпункты: Создать, Открыть, Сохранить, Сохранить как, Печать и Выход.
* Пункт меню **«Параметры»** содержит подпункты: Шрифт, Показать/Скрыть панель инструментов.
* Пункт меню **«Поиск»** содержит возможность вызова метода для подсчета количества слов в тексте и метода по вашему варианту, а также пункт «Найти файлы в указанном каталоге».
* Пункт меню **«Справка»** содержит подпункты «Об авторе» и «О программе». Кнопки на панели инструментов дублируют некоторые пункты главного меню.



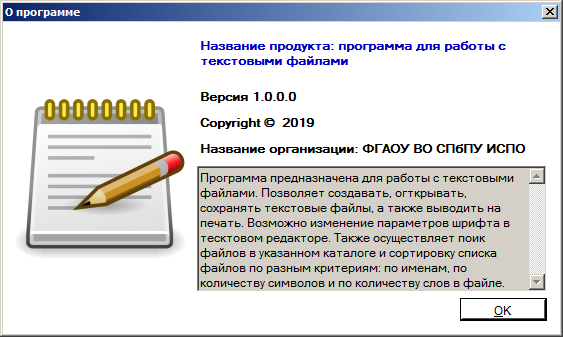
**Рисунок 1** Примерный вид главной формы проекта

1. Подключите динамическую библиотеку к своему проекту. Для подключения библиотеки добавьте в проект ссылку на библиотеку (файл с расширением .dll) и укажите имя библиотеки в списке используемых модулей.
2. Создайте метод обработки события выбора пункта меню «Открыть файл». При выборе этого пункта пользователю предлагается выбрать файл для открытия при помощи диалогового окна, а затем вызывается ранее созданный метод открытия файла из динамической библиотеки. Сохраните и откомпилируйте проект. Исправьте синтаксические ошибки. Запустите проект на выполнение. Протестируйте метод выбора пункта меню «Открыть файл». Произведите отладку проекта.
3. Создайте метод выбора пункта меню «Сохранить файл» и «Сохранить как». Эти функции будут похожи. При обычном сохранении файла используем тоже имя, которое было задано в переменной при открытии файла. При выборе пункта меню «Сохранить как» происходит вызов диалогового окна сохранения файла. Если пользователь вводил текст вручную, не открывая файл, или редактировал открытый ранее файл, то при закрытии формы нужно предложить ему сохранить его. Произведите отладку проекта.
4. Создайте метод выбора пункта меню «Печать». При нажатии на кнопку вызывается соответствующий метод дополнительного класса. Произведите отладку проекта.
5. Назначьте соответствующим кнопкам на панели инструментов методы обработки событий выбора нужных пунктов меню. Дважды писать один и тот же код в программе не нужно.
6. Создайте метод обработки события выбора пункта меню: «Поиск файлов». При этом должно вызываться окно второй формы для поиска файлов в указанном каталоге.



**Рисунок 2** Дополнительная форма – окно поиска и сортировки файлов

1. Создайте методы обработки события выбора остальных пунктов меню: «Параметры», «Поиск», «Справка». Пункт меню «Параметры» должен предоставлять возможности скрыть/показать панель инструментов, настроить параметры шрифта с помощью диалогового окна «Шрифт», настроить цвет шрифта с помощью диалогового окна выбора цвета. Пункт меню «Справка» должен открывать диалоговое окно «О программе» или стандартное окно сообщения «Об авторе».



**Рисунок 3** Вызов окна «О программе»

1. Внесите подробные XML-комментарии в текст библиотеки, обычные комментарии в код основной программы. Скопируйте в отчёт тексты всех модулей (включая тестовый проект).
2. Составьте тестовый сценарий (план тестирования). Занесите его в отчёт.
3. Протестируйте приложение в соответствие с тестовым сценарием. Вставьте в отчёт скриншоты проекта на этапе выполнения (результаты тестирования).
4. Продемонстрируйте работу проекта и отчёт преподавателю и получите оценку.

*Указания:*

Приведём пример метода открытия файла (выбора пункта меню «Открыть»). Тот же код должен быть назначен кнопке «Открыть» на панели инструментов.

private void открытьToolStripMenuItem\_Click(object sender, EventArgs e)

{

toolStripStatusLabel3.Text = "Состояние: Открытие файла";

try

{

openFileDialog1.FileName = String.Empty;

openFileDialog1.ShowDialog();

F = new TextFile(openFileDialog1.FileName);

F.OpenFile(); //вызов метода класса TextFile

richTextBox1.Text = F.Text;

}

catch

{

MessageBox.Show("Ошибка доступа к файлу!", "Error",

MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

В классе TextFile будет присутствовать метод, который непосредственно выполняет открытие файла:

public void OpenFile()

{

System.IO.StreamReader sr = new System.IO.StreamReader(this.path,

System.Text.Encoding.GetEncoding(1251));

// создание нового входного потока, установка кодировки

this.text = sr.ReadToEnd();

sr.Close(); // закрытие входного потока

}

В классе TextFile необходимо разработать методы для вывода текста на принтер. Приведем примерный код таких методов:

StreamReader streamToPrint; //поток для принтера

Font printFont;

// Метод печати файла

// Входные параметры: передаются параметры шрифта

// Результат: переданный текст выводится на печать,

// если нет ошибки, передается результат true

public bool PrintResult(Font pF)

{

try

{

streamToPrint = new System.IO.StreamReader(filename,

System.Text.Encoding.GetEncoding(1251));

try

{

printFont = pF;

PrintDocument pd = new PrintDocument();

pd.PrintPage += new PrintPageEventHandler

(this.pd\_PrintPage);

pd.Print();

return true;

}

finally

{

streamToPrint.Close();

}

}

catch

{

return false;

}

}

// Событие PrintPage вызывается для каждой страницы, которая будет напечатана

private void pd\_PrintPage(object sender, PrintPageEventArgs ev)

{

float linesPerPage = 0;

float yPos = 0;

int count = 0;

float leftMargin = ev.MarginBounds.Left;

float topMargin = ev.MarginBounds.Top;

string line = null;

// Чтобы вычислить количество строк на странице

linesPerPage = ev.MarginBounds.Height / printFont.GetHeight(ev.Graphics);

// Печатаем каждую строку файла

while (count < linesPerPage && ((line = streamToPrint.ReadLine()) != null))

{

yPos = topMargin + (count \* printFont.GetHeight(ev.Graphics));

ev.Graphics.DrawString(line, printFont, Brushes.Black,

leftMargin, yPos, new StringFormat());

count++;

}

// если строки не закончились, распечатаем еще одну страницу

if (line != null)

ev.HasMorePages = true;

else

ev.HasMorePages = false;

}

**Подведение итогов лабораторной работы:**

Результатом выполнения лабораторной работы должен быть проект, сохранённый в отдельной папке, и отчёт. Отчёт должен содержать:

* название лабораторной работы, цель работы;
* текст вашего задания с указанием номера варианта;
* код собственной библиотеки с XML-комментариями;
* тест-кейсы и скриншот их успешного выполнения;
* скриншоты форм с надписанными компонентами;
* тексты программных модулей с комментариями;
* тестовый сценарий в виде таблицы;
* скриншоты проекта на этапе выполнения;
* колонтитулы с вашей фамилией и номером группы.

**Контрольные вопросы:**

1. Какие диалоговые компоненты можно использовать для организации работы пользователя с файлом? Перечислите свойства диалоговых компонентов для работы с файлами.
2. Как подключить динамическую библиотеку к проекту?
3. Как вызвать методы динамической библиотеки?
4. Как организовать тестирование методов библиотеки?
5. Как организовать сортировку списка найденных файлов по разным критериям?

# Варианты:

1. Разработать метод, который выполняет поиск в файле самого длинного слова (таких может быть и несколько). Результатами поиска должны быть все эти слова и максимальное количество букв.
2. Разработать метод, который выполняет поиск в текстовом файле слов, встречающихся 2 или более раз, и возвращает список таких слов в алфавитном порядке.
3. Разработать метод, который выполняет поиск в текстовом файле слов, которые не включают заданную букву или сочетание букв, и возвращает список таких слов в порядке следования в тексте или в алфавитном порядке по желанию пользователя.
4. Разработать метод, который выполняет поиск в текстовом файле слов, заканчивающихся на заданную букву или слог, стоящих в конце предложений, и возвращает список слов в порядке следования их в тексте.
5. Разработать метод, который выполняет поиск в текстовом файле слов, включающих заданное сочетание букв в начале или в конце слова, и возвращает список слов в порядке следования в тексте.
6. Разработать метод, который выполняет поиск в текстовом файле слов, включающих все буквы заданного слова в том же порядке плюс любые другие, и возвращает список слов в порядке следования в тексте.
7. Разработать метод, выполняет поиск в текстовом файле слов, включающих все буквы заданного слова в любом порядке, и возвращает список слов в порядке следования в тексте (для каждого файла отдельно).
8. Разработать метод, который производит сравнение двух файлов по количеству символов, и в том файле, который оказался больше, удаляет лишние строки и выравнивает длину каждой строки (укорачивает строки).
9. Разработать метод, который производит сравнение двух файлов по количеству символов, и в том файле, который оказался меньше, добавляет к более коротким строкам указанный символ и дописывает недостающие строки, заполняя их тем же символом.
10. В текстовом файле хранятся двоичные числа (наборы нулей и единиц, каждое число на отдельной строке). Разработать метод, который выполняет перевод этих чисел в восьмеричную систему счисления. Если какая-либо строка в файле содержит не двоичное число (то есть в ней встречаются символы, отличные от 0 и 1), то в качестве результата вернуть прочерк.
11. В текстовом файле хранятся шестнадцатеричные числа (каждое число на отдельной строке). Разработать метод, который выполняет перевод этих чисел в десятичную систему счисления. Если какая-либо строка в файле содержит не шестнадцатеричное число (то есть в ней встречаются символы, отличные от нужного набора), то в качестве результата вернуть прочерк.
12. Разработать метод, который проверяет правильность оформления арифметического выражения вида Y = a\*x + b, где x, Y – переменные, a, b – целые или вещественные числа (каждое выражение хранится в файле на отдельной строке). Метод сохраняет в результирующий файл те выражения, которые составлены верно.
13. Разработать метод, который проверяет правильность расстановки круглых скобок в арифметическом выражении, а также метод, который в заданном файле проверяет арифметические выражения (каждое выражение хранится на отдельной строке) и сохраняет в результирующий файл те из них, где скобки расставлены неверно.
14. Разработать метод, который выполняет поиск в текстовом файле самой длинной последовательности цифр (таких может быть несколько).
15. Разработать метод, выполняет частотный анализ текста (посимвольный), то есть подсчитывает количество и относительную частоту встречаемости различных символов в тексте. Текст вводится с клавиатуры или из текстового файла, результат выводится на экран.
16. Разработать метод, который вычисляет расстояние между точками на плоскости, заданными своими координатами, а также метод, который определяет минимальное расстояние между N точками, заданными координатами на плоскости. Координаты точек заранее внести в текстовый файл. Возможно дополнение файла новыми точками или перезапись файла по желанию пользователя. Расчет производится по данным, занесенным в файл.
17. Разработать метод, который вычисляет расстояние между точками на плоскости, заданными своими координатами, а также метод, который определяет периметр фигуры, заданной N точками, (координатами на плоскости). Координаты точек заранее внести в текстовый файл. Возможно дополнение файла новыми точками или перезапись файла по желанию пользователя.
18. Разработать метод, который выполняет поиск в текстовом файле слов, заканчивающихся на любую согласную (кириллицы или латиницы по выбору) и возвращает список таких слов в порядке следования в тексте.
19. Разработать метод, который выполняет поиск в текстовом файле сочетаний из двух слов на русском языке, разделённых пробелом, у которых в конце первого слова и в начале второго в сумме более 4-х согласных, например: текст программы (5 согласных), и возвращает список таких сочетаний слов в порядке следования их в тексте.
20. Разработать метод, который выполняет анализ текста: сколько в нем цифровых символов, разделителей, букв латиницы, кириллицы и специальных знаков, а также вычисляет эти данные в процентном соотношении и возвращает результат в виде структуры.