Лабораторная работа №7.

Тема: *Работа с компонентами ListBox, ComboBox, RadioButton, CheckBox, GroupBox, Panel.*

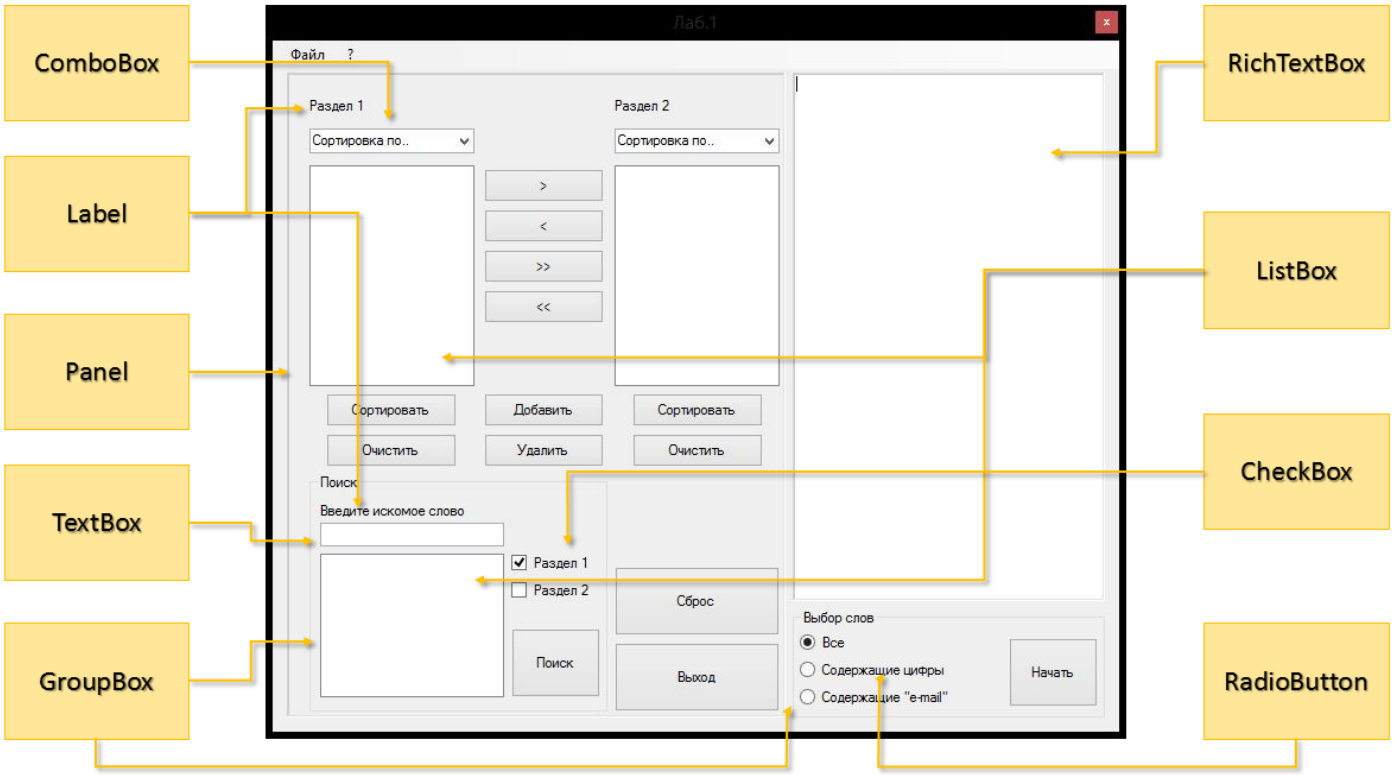


Рисунок 1.

Требования к работе:

1) Создать визуальную часть, используя необходимые компоненты. Все нужные компоненты указаны на рисунке 1. Их расположение и общий вид формы может выбираться самостоятельно.

2) Приложение должно позволить пользователю открыть текстовый файл, который считается в RichTextBox. Далее, пользователь может выбрать критерий по которому он хочет отобрать слова: «Все», «Содержащие цифры», «Содержащие ‘e-mail’. После нажатия на кнопку «Начать», текст в RichTextBox’е разбивается на слова, которые в свою очередь, заносятся в ListBox (Раздел 1), по заданному критерию. Между двумя разделами имеется панель, в которой находятся 4 кнопки, посредством которых можно переносить отдельные выбранные слова, либо всю коллекцию из одного ListBox’а в другой, а также кнопки «Добавить» и «Удалить», которые соответственно реализуют добавление/удаление элементов из разделов.

Также каждый раздел можно очистить, либо отсортировать четырьмя способами: по длине (возр.), по длине (убыв.), по алфавиту (возр.) и соответственно по алфавиту (убыв.). В нижнем правом углу находится блок, отвечающий за поиск строк в разделах. Должна быть также реализована возможность сохранения содержимого из Раздела 2 в текстовый файл.

3) Реализовать Сортировку разделов, любым известным алгоритмом сортировки самостоятельно.

Задание 1. Создание визуальной части приложения.

1) Перенесите на форму все необходимые элементы из Панели элементов, чтобы сделать форму, показанную на рисунке 1.

2) Создайте меню.

Порядок действий:

1. Расположите все компоненты, как показано на рисунке 1, либо в произвольном порядке.

2. У компонента Panel установите значение свойства BorderStyle (Fixed3D – выпуклая, утопленная). У компонентов RadioButton, CheckBox и ComboBox установите начальные значения свойств Checked, Checked и Text, как показано на рисунке 1, соответственно. Измените свойство SelectionMode у ListBox’ов на MultiExtended. Также добавьте в свойство Items обоих компонентов ComboBox, четыре строки:

Алфавиту (по возрастанию)

Алфавиту (по убыванию)

Длине слова (по возрастанию)

Длине слова (по убыванию)

3. Создание меню.

а) Перенесите на форму компонент MenuStrip.

б) Создайте меню по типу, показанному на рисунке 2. Для добавления пунктов/подпунктов просто, нажимайте на квадратные области в месте, где установлено меню, и вводите необходимый текст. Установите «горячие клавиши»: нажмите на необходимый элемент меню, например «Открыть»,

далее перейдите в свойства компонентов и в свойстве «ShortcutKeys» установите необходимые сочетания.

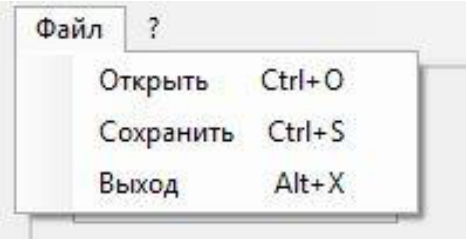


Рисунок 2.

Задание 2. Программирование элементов.

1) Меню.

2) Обработчики нажатий на кнопки.

Порядок действий:

1. Меню.

а) Реализуйте открытие текстового файла. Для того, чтобы открыть и записать текстовый файл в RichTextBox, необходимо изначально в обработчике события «Click» элемента «Открыть» создать объект класса OpenFileDialog.

```
OpenFileDialog OpenDlg = new OpenFileDialog();
```

Далее, если в диалоговом окне, пользователь нажмёт на кнопку «ОК», то нужно считать выбранный файл в RichTextBox. Для этого мы создаём объект класса StreamReader, параметрами которого будут являться Имя выбранного файла и стандартная кодировка. Считывание производится с помощью метода ReadToEnd(), который считывает текстовый файл от начала до конца в необходимое местоположение.

```
if (OpenDlg.ShowDialog() == DialogResult.OK)
{
       StreamReader Reader = new StreamReader(OpenDlg.FileName, Encoding.Default);
       richTextBox1.Text = Reader.ReadToEnd();
       Reader.Close();
}

OpenDlg.Dispose();
```

б) Сохранение файла производится аналогично, только теперь мы создаём объект класса StreamWriter, и из ListBox’а построчно заносим в файл наши слова:

```
if (SaveDlg.ShowDialog() == DialogResult.OK
{
	StreamWriter Writer = new StreamWriter(SaveDlg.FileName);

	for (int i = 0; i < listBox2.Items.Count; i++)
	{
		Writer.WriteLine((string)listBox2.Items[i]);
	}

	Writer.Close();
}

SaveDlg.Dispose();
```

в) Выход из приложения: `Application.Exit();`

г) Информационное сообщение: `MessageBox.Show("Информация о приложении и разработчике");`

2. Обработчики нажатий на кнопки.

а) Кнопка «Начать». Как было уже сказано выше, при нажатии на кнопку начать, нам необходимо разбить считанный текст на слова, т.е. убрать пробелы, знаки табуляции и переходы на новую строку. Для удобства мы заносим текст в массив строк, из которого, в свою очередь, отбираем необходимые строки по 3-ём критериям. По первому критерию мы заносим абсолютно все строки в список, делается это с помощью метода `listBox1.Items.Add`, параметром для которого соответственно является строка, которую мы хотим добавить. Для того, чтобы реализовать следующие два критерия нам необходимо добавить ещё одно пространство имён `using System.Text.RegularExpressions`. В нём имеется класс Regex, с помощью одного из методов которого мы и будем искать вложенные подстроки: `(Regex.IsMatch(Str, @"\d"))`. Т.е., как очевидно, метод IsMatch, ищет в строке Str некоторые подстроки, и если в какой либо строке находится подстрока, удовлетворяющая условиям поиска, то он возвращает true. Пример работы кнопки «Начать» показан ниже.

```
listBox1.Items.Clear();
listBox2.Items.Clear();

listBox1.BeginUpdate();

string[] Strings = richTextBox1.Text.Split(new char[] { '\n', '\t', ' ' },
StringSplitOptions.RemoveEmptyEntries);

foreach (string s in Strings)
{
	string Str = s.Trim();

	if (Str == String.Empty) continue;
	if (radioButton1.Checked) listBox1.Items.Add(Str);
	if (radioButton2.Checked)
```

```
        {
            if (Regex.IsMatch(Str, @"\d")) listBox1.Items.Add(Str);
        }
        if (radioButton3.Checked)
        {
            if (Regex.IsMatch(Str, @"\w+@\w+\.\w+")) listBox1.Items.Add(Str);
        }
}

listBox1.EndUpdate();
```

б) Кнопки «Выход», «Сброс», «Очистить». Запрограммируйте кнопку «Выход» соответственно для закрытия приложения. «Сброс» означает, что наша форма должна вернуться к первоначальному виду, т.е. нужно очистить ListBox’ы (метод ListBox.Items.Clear()), поле Text у RichTextBox’а и TextBox’а, установить изначальное состояние свойства Checked у RadioButton и CheckBox.

в) «Поиск». Для поиска подстрок в строках, можно воспользоваться методом Contains класса string. Пример работы функции поиска:

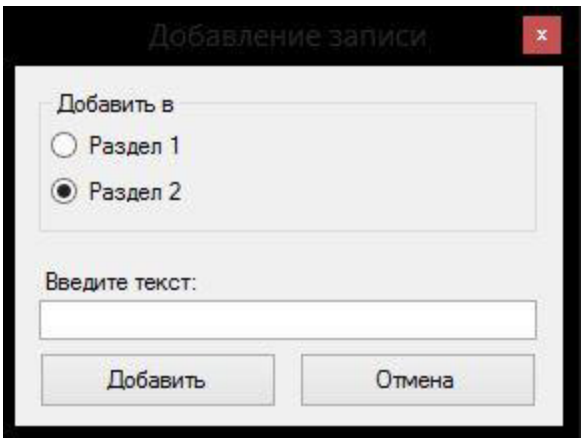
```
            listBox3.Items.Clear();

            string Find = textBox1.Text;

            if (checkBox1.Checked)
            {
                foreach (string String in listBox1.Items)
                {
                    if (String.Contains(Find)) listBox3.Items.Add(String);
                }
            }

            if (checkBox2.Checked)
            {
                foreach (string String in listBox2.Items)
                {
                    if (String.Contains(Find)) listBox3.Items.Add(String);
                }
            }
```

г) Кнопки «Добавить» и «Удалить». При нажатии на кнопку добавить у нас должна открыться новая модульная форма (модульная, значит, что пока она открыта, мы не можем взаимодействовать с основной формой). Чтобы создать новую форму, нужно в Обозревателе решений нажать правой кнопкой на наш проект и: Добавить –> Создать элемент.., где в открывшемся окне выбираем «Форма Windows Forms». Данная форма должна выглядеть примерно так:



Для того, чтобы связать две формы нужно в обработчике для кнопки «Добавить» (которая находится в ГЛАВНОЙ ФОРМЕ) вписать такой код:

```
Form2 AddRec = new Form2();

AddRec.Owner = this;
AddRec.ShowDialog();
```

В первой строке, мы создали объект AddRec, т.е. создали форму. Далее, нам необходимо указать, что «родителем» этой новой формы является наша главная форма (свойство Owner). Затем мы соответственно открываем созданную форму, используя метод ShowDialog, который собственно и делает эту форму модульной.

Теперь, в форме 2, создайте обработчик нажатия на кнопку «Добавить» и пропишите такой код:

```
Form1 Main = this.Owner as Form1;

if (textBox1.Text != "")
{
    if (this.radioButton1.Checked == true)
    Main.listBox1.Items.Add(this.textBox1.Text);
    else Main.listBox2.Items.Add(this.textBox1.Text);

    this.Close();
}
```

В первой строке, мы создаём объект Main класса Form1(наша основная форма) и указываем, что Main, и есть «родитель» формы 2. Наконец, мы связали наши формы, но подобная связка позволит им взаимодействовать только при условии, что мы «откроем» необходимые поля. Для этого, опять же, в Обозревателе решений откройте файл Form1.Designer.cs, найдите блок кода, где были созданы ListBox’ы, и вместо private, присвойте им уровень доступа public. Теперь мы можем из второй формы обращаться к спискам первой. Создайте обработчик для нажатия кнопки Отмена, которая закрывает Форму 2.

Кнопка «Удалить» позволит нам удалить выбранные строки из разделов. Целесообразно создать отдельную функцию, например DeleteSelectedStrings, в которую и передавать нужный ListBox, в зависимости от выбранного для удаления раздела:

```
for (int i = ListBox.Items.Count - 1; i >= 0; i--)
{
	if (ListBox.GetSelected(i)) ListBox.Items.RemoveAt(i);
}
```

д) Кнопки переноса строк.

Для переноса строк из одного раздела в другой необходимо воспользоваться методом ListBox.Items.Add(строка из другого лист бокса), для обращения к отдельным строкам: ListBox.Items.[..]. Не забудьте после переноса строки удалять строки из исходного раздела, методы Remove() и RemoveAt(), в первый посылается в параметры объект, а во второй индекс нужной строки соответственно. Для обращения к выделенным строкам: ListBox.SelectedItems. Пример кода для переноса всей коллекции из первого списка во второй:

```
ListBoxTo.Items.AddRange(ListBoxFrom.Items);
ListBoxFrom.Items.Clear();
```

Перенос выделенных строк:

```
ListBoxTo.BeginUpdate();

foreach (object Item in ListBoxFrom.SelectedItems)
{
	ListBoxTo.Items.Add(Item);
}

ListBoxTo.EndUpdate();
```

е) «Сортировать». Напишите самостоятельно функции для сортировки разделов с помощью любых вам известных алгоритмов сортировки. *Подсказка: у ListBox'а есть свойство Sort, которое располагает элементы списка в алфавитном порядке. Для реализации сортировок «по убыванию» можно перенести все элементы списка в некий массив, у которого есть метод Reverse(). Для сортировки по длине, можно написать отдельную функцию, которая будет сортировать элементы списка в неком массиве, затем очищать список и заносить в него уже отсортированные элементы.*

Пример готовой программы: