Практикум 4. Приложение «Библиотечный фонд»

Постановка задачи

В текстовом файле в единообразном виде хранятся данные о книгах, хранящихся в библиотеке: Автор (фамилия и инициалы), жанр, название, год издания, количество экземпляров.

Обеспечить:

1. Чтение данных из файла и вывод на экран в виде таблицы.
2. Поиск данных по полю автор. Найденные элементы записываются в новый файл. В первой строке должно быть записана дата, затем критерий поиска, затем список найденных элементов, ниже выводится общее количество найденных записей. Если таковых нет, то должно быть выдано сообщение вида: «Элементов, удовлетворяющих критериям поиска не обнаружено».

Пример выводимого сообщения:

Дата обращения: 21.09.2013

Поиск по автору: Пушкин А. С.

Пушкин А. С. Поэзия Евгений Онегин 2004 3

Пушкин А. С. Проза Капитанская дочка 2011 5

Итого найдено: 2 наименования

3. Возможность добавления данных в текстовый файл.
4. Возможность изменения данных в файле (перезаписывать измененный файл под другим именем).

Структура -- пользовательский тип данных

В данной задаче данные для хранения следует представить в виде структуры, т.е. созданного программистом собственного типа данных, состоящем из набора стандартных типов. В этом случае проще будет организовать работу с данными, собрав их в единый массив или коллекцию.

Структура - это набор зависимых друг от друга переменных. Зависимость здесь исключительно логическая и определяется условиями задачи.

Описание структуры:

```
struct имя_структуры
{
   public тип поле1;
   public тип поле2;
}
```

Для решения задачи целесообразно описать структуру **Book,** содержащую текстовые и числовые поля.

Описание типа данных было выполнено в глобальной области, чтобы этот тип можно было использовать в нескольких обработчиках событий.

Данные о книгах хранятся в текстовом файле. Файл имеет определенную структуру, которой следует строго придерживаться. Так, например, данные (т.е. значения полей одного экземпляра структуры) разделяются точкой с запятой. Такие файлы называют «файлами с разделителем».

Некоторые подробности о файлах и возможности работы с ними описаны в теоретических отступлениях.

Подробнее: ☞8. Чтение и запись в текстовый файл.

Для отбора данных и для сортировки коллекции использованы Linq-запросы. Их конструкция проста и понятна из рассмотренных в коде примеров.

Элементы управления

Вид работающего меню: используйте элемент управления ***MenuStrip***, (для вставки разделителя щелкните правой клавишей мыши в пункте меню и выберите ***Insert – Separator***

Для создания клавиш быстрого доступа (горячие клавиши), необходимо перед символом имени (пункта) меню ввести символ ***&***. Например, для пункта меню ***File*** надо ввести ***&File*** (Рисунок 28). Для установления клавиш быстрого набора следует установить свойство ***ShortcutKeys***. Например, ***Ctrl+F9***

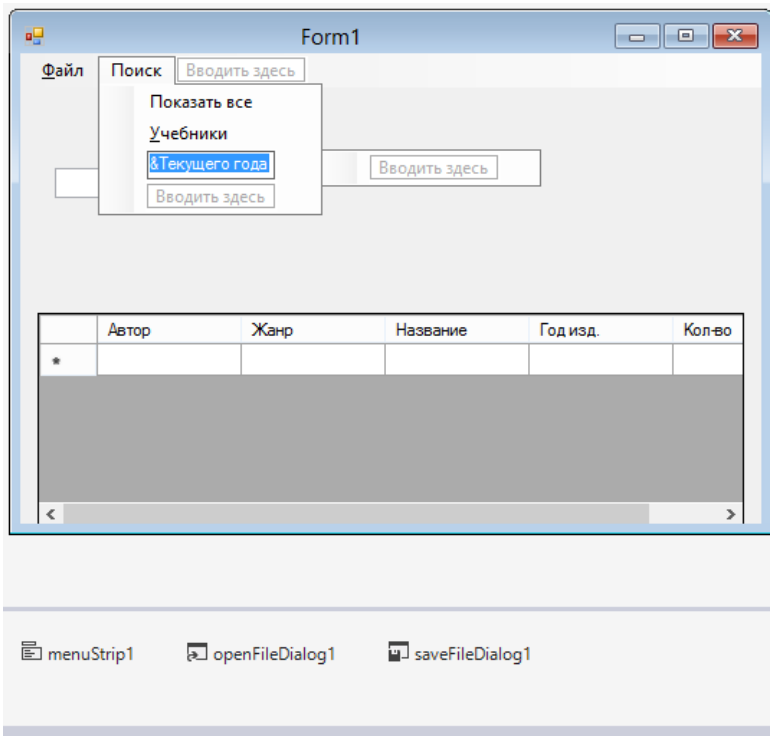

Рисунок 28. – Размещение на форме элемента MenuStrip

Для организации чтения из выбранного файла и записи в файл необходимо разместить на форме невизуальные компоненты из группы Диалоговые окна ***OpenFileDialog***, ***SaveFileDialog***, которые отвечают за создание стандартных диалоговых окон.

У элемента управления ***OpenFileDialog*** были установлены свойства:

DefaulExt=txt, Filter=TextFiles|.txt*

У элемента управления ***OpenFileDialog*** свойство: *DefaulExt=txt*

Внешний вид приложения представлен на Рисунке 29.

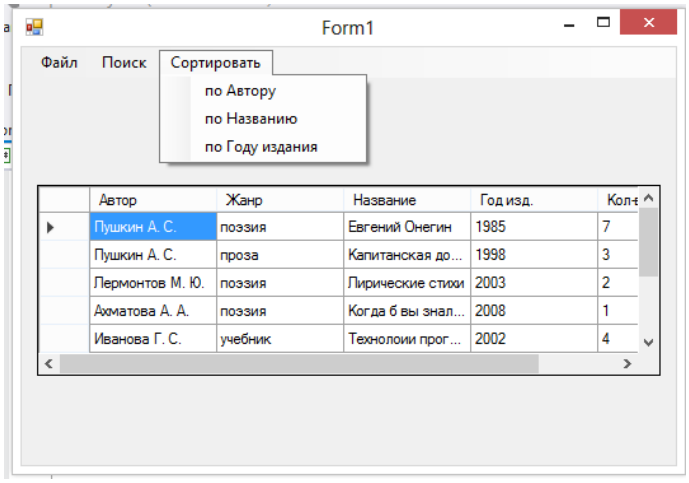

Рисунок 29. – Внешний вид приложения

Полностью код проекта представлен в листинге 9.

Листинг 9

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;


namespace BiblioFond
{
    public partial class Form1 : Form
    {

        struct book
        {
            public string autor;
            public string name;
            public string kind;
            public int year;
            public int numb;

        }
        List<book> spisok = new List<book>();
        StreamReader readfl;
        StreamWriter writefl;

        public Form1()
        {
            InitializeComponent();
        }


        private void print()
        {
            dataGridView1.Rows.Clear();
            foreach (var t in spisok)
                dataGridView1.Rows.Add(t.autor, t.kind, t.name, t.year.ToString(),
t.numb.ToString());

        }
        //Открытие файла и вывод в таблицу
        private void открытьФайлToolStripMenuItem_Click(object sender, EventArgs e)
        {
            string str;
            if (openFileDialog1.ShowDialog() == DialogResult.OK)
            {
                openFileDialog1.InitialDirectory = Environment.CurrentDirectory;
                string path = openFileDialog1.FileName;

                try
                {

                    readfl = new StreamReader(path, Encoding.Default);
                    //readfl = new StreamReader(path);
                    while (readfl.EndOfStream == false)
                    {
                        book t;
                        str = readfl.ReadLine();
                        string[] s = str.Split(';');
```

```
                        t.autor = s[0];
                        t.kind = s[1];
                        t.name = s[2];
                        t.year = Convert.ToInt32(s[3]);
                        t.numb = Convert.ToInt32(s[4]);
                        spisok.Add(t);
                    }
                    readfl.Close();
                    print();
                }


                catch
                {
                    MessageBox.Show("Ошибка чтения файла");
                }
            }
        }

        private void поНазваниюToolStripMenuItem_Click(object sender, EventArgs e)
        {
            var queryResult =
                from n in spisok
                    //сортировка по полю Название
                orderby n.name
                select n;
            dataGridView1.Rows.Clear();
            foreach (book n in queryResult)
            {
                dataGridView1.Rows.Add(n.autor, n.kind, n.name, n.year.ToString(),
n.numb.ToString());
            }
        }

        private void поАвторуToolStripMenuItem_Click(object sender, EventArgs e)
        {
            var queryResult =
                from s in spisok
                .OrderBy(s => s.autor)
                .ThenBy(s => s.name)
                select s;
            dataGridView1.Rows.Clear();
            foreach (book n in queryResult)
            {
                dataGridView1.Rows.Add(n.autor, n.kind, n.name, n.year.ToString(),
n.numb.ToString());

            }
        }

        private void поГодуИзданияToolStripMenuItem_Click(object sender, EventArgs e)
        {
            var queryResult =
                 from s in spisok
                 .OrderBy(s => s.year)
                 select s;
            dataGridView1.Rows.Clear();
            foreach (book n in queryResult)
            {
                dataGridView1.Rows.Add(n.autor, n.kind, n.name, n.year.ToString(),
n.numb.ToString());

            }
        }
```

```
        private void показатьВсеToolStripMenuItem_Click(object sender, EventArgs e)
        {
            dataGridView1.Rows.Clear();
            print();
        }

        private void учебникиToolStripMenuItem_Click(object sender, EventArgs e)
        {
            var queryResult =
                        from n in spisok
                            //отбор по условию
                        where n.kind == "учебник"
                        select n;
            dataGridView1.Rows.Clear();
            foreach (book n in queryResult)
            {
                dataGridView1.Rows.Add(n.autor, n.kind, n.name, n.year.ToString(),
n.numb.ToString());
            }
        }

        private void текущегоГодаToolStripMenuItem_Click(object sender, EventArgs e)
        {
            var queryResult =
                                    from n in spisok
                                        //отбор по условию
                                    where n.year == DateTime.Now.Year
                                    select n;
            dataGridView1.Rows.Clear();
            foreach (book n in queryResult)
            {
                dataGridView1.Rows.Add(n.autor, n.kind, n.name, n.year.ToString(),
n.numb.ToString());
            }
        }

        private void сохранитьДаныеToolStripMenuItem_Click(object sender, EventArgs e)
        {
            if (saveFileDialog1.ShowDialog() == DialogResult.OK)
            {
                saveFileDialog1.InitialDirectory = Environment.CurrentDirectory;
             string path1 = saveFileDialog1.FileName;

                try
                {
                    string result;

                    writefl = new StreamWriter(path1,false, Encoding.Default);
                    //readfl = new StreamReader(path);
                    //while (writefl.EndOfStream == false)
                        result = string.Format("{0,12}{1,10}{2,35}{3,5}{4,3}", "Автор",
"Жанр", "Название", "Год издания", "Кол-во экз.");
                    writefl.WriteLine(result);
                    for (int k = 0; k < dataGridView1.Rows.Count - 1; k++)
                    {
                        result = string.Format("{0,12}{1,10}{2,35}{3,5}{4,3}",
                                        dataGridView1.Rows[k].Cells[0].Value+" ",
                                        dataGridView1.Rows[k].Cells[1].Value + " ",
                                        dataGridView1.Rows[k].Cells[2].Value + " ",
                                        dataGridView1.Rows[k].Cells[3].Value + " ",
                                        dataGridView1.Rows[k].Cells[4].Value);
                        writefl.WriteLine(result);
                    }
                    writefl.Close();
                }
```

```
                catch (Exception exc)
                {

                    MessageBox.Show("Ошибка доступа к файлу" + exc.ToString(),
"BiblioFond", MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
        }
    }
}
```

Задание 4.1.

Создайте новый проект, обеспечивающий требуемый функционал. Изучите приведенный выше код. Протестируйте работу приложения BiblioFond.

Итоговое задание

Создайте свой вариант решения задачи. Расширьте функционал проекта возможностью добавлять данные в файл со списком литературы.

РЕКОМЕНДАЦИЯ: добавьте проверку вводимых данных для текстовых полей. Кнопку (пункт меню) «Добавить книгу» активируйте только после того, как все необходимые поля будут заполнены.

Выполненное задание пришлите на проверку тьютору стажировки-практикума.

8. ЧТЕНИЕ И ЗАПИСЬ В ТЕКСТОВЫЙ ФАЙЛ.

Для работы с потоками в первую очередь необходимо подключить библиотеку

```
 System.IO;
Using System.IO;
```

Для записи в файл нужно сначала создать файл и открыть поток для работы с ним

```
StreamWriter sw = File.CreateText("test.txt");
```

Здесь создан поток sw, а при помощи метода `File.CreateText("test.txt")` создан файл «`test.txt`»(по умолчанию сохраняется в папку Debug сохраненного проекта).

После того как открыт поток, можно записывать в него текстовые строки используя методы `Write` и `WriteLine`

```
sw.WriteLine("Мой первый файл созданный в C#");
```

Но если проверить после запуска программы файл, то можно увидеть что он пуст. Это нормально. Данные не появятся в файле до тех пор пока поток не будет закрыт.

Делается это с помощью метода `Close`:

```
sw.Close();
```

Теперь после запуска вы увидите, что в файле появилась строка введенного текста.

Рассмотрим ситуацию, когда файл существует и в него необходимо дописать какой-либо текст.

Для этого понадобится метод `AppendText`.

```
StreamWriter sw = File.AppendText("test.txt");
sw.WriteLine("Добавили вторую строчку в файл");
sw.Close();
```

Чтение из файла.

Для чтения в первую очередь нужно открыть поток класса StreamReader, привязав его к файлу.

Делается это при помощи метода File.OpenText('имя файла');

```
StreamReader sr = File.OpenText("test.txt");
```

Если нужно считать только первую строку из файла, то для этого воспользуемся следующей конструкцией sr.ReadLine().

Допустим вам нужно вывести первую строчку из файла на консоль. Делается следующим образом:

```
System.Console.WriteLine(sr.ReadLine());
```

Немного сложнее если мы не знаем сколько строк в файле, а нужно считать содержимое всего файла. Чтение из файла проводится построчно. Для вывода всего файла нам нужно построчно его считывать до тех пор пока в нем есть строчки.

```
while (true){
string st = sr.ReadLine();
if (st==null)
break;
System.Console.WriteLine(st);
}
```

В этом примере мы считали файл построчно и вывели его на консоль.

Как и при записи в файл, так и при чтении из него не стоит забывать о закрытии потока методом Close.

```
Sr.Close();
```

Пример (создадим файл запишем в него 2 строчки текста и после этого выведем их на консоль)

```
StreamWriter sw = File.CreateText("test.txt");
sw.WriteLine("Первая строка");
sw.WriteLine("Вторая строка");
sw.Close();
StreamReader sr = File.OpenText("test.txt");
while (true)
{
string st = sr.ReadLine();
if (st==null)
break;
System.Console.WriteLine(st);
}
sw.Close();
System.Console.ReadLine();
```

Файлы с разделителями.

Файлы с разделителями – это распространенная форма хранения данных, используемая во многих системах. Например, файл хранит данные о студентах. В каждой строке файла хранится информация об одном студенте, в качестве разделителей могут служить, например, запятые: «Иванов, Иван, 1992, Прикладная информатика».

Данные из такого файла будут читаться построчно, а затем они должны быть разделены на отдельные части, соответствующие разделителям. В нашем примере из каждой строки следует выделить «Фамилию», «Имя», «Год рождения», «Специальность».

Для этого понадобится строковая команда Split.

<строка>.Split () – преобразует строку в массив строк, разбивая ее на части по указанным символам-разделителям. Эти символы указываются в массиве char, который может содержать, например, запятую, пробел и др. символы. Символы-разделители указываются в апострофах.

```
String [] MyWords; //описание массива для хранения слов строки
MyWords=MyString.Split(‘,’);//формирование массива слов
```

Далее все слова в этом массиве выводятся на консоль с помощью цикла foreach:

```
Foreach(string word in MyWords)
{
Console.WriteLine(“{0}”,word);
}
```

При работе команды Split, все лишние пробелы удаляются, остаются только сами слова.

☞ Назад к основному тексту