№ 3 Регулярные выражения, атрибуты валидации, меню, панели инструментов, строки состояния

Задание

1) Измените функциональность предыдущей лабораторной работы. Добьте на форму меню с пунктами:
   a. «Поиск» (подменю указанными в вариантах) В поиске, кроме поиска на полное соответствие, реализовать поиск по на основе регулярных выражений (диапазон, наличие букв на определенных позициях, число повторений символов и т.п.). Результаты поисковых запросов можно выводить в отельное окно. Сделайте отдельное окно для конструирования поисковых запросов (в том числе и по нескольким критериям).
   b. «Сортировка по» (году, фамилии, специальности и т.п.) Для поиска, сортировки и модификаций используйте LINQ.
   c. «Сохранить» результаты поиска и сортировок в отдельных xml или json-файл. Используйте сериализацию.
   d. Добавьте пункт «О программе». При выборе пункта меню «О программе» должно выводиться окно сообщений с версией и ФИО разработчика.
2) Добавьте валидацию данных на основе атрибутов. При валидации вводимых данных используйте функционал в виде атрибутов из пространства имен System.ComponentModel.DataAnnotations и классов ValidationResult, Validator и ValidationContext. Используйте атрибуты RegularExpression, Range, свойство ErrorMessage и т.д
3) Создайте свой атрибут валидации (см. таблицу с вариантами).
4) Добавить панель инструментов с кнопками дублирующими команды меню «поиск», «сортировки», «очистить», «удалить», «вперед», «назад». Добавить возможность скрывать и закреплять панель инструментов.
5) Добавить строку состояния с тестовыми сообщениями о текущем количестве объектов и последнем выполненном действии, текущей датой и временем.

Вариант	Задание
1, 9	Поиск по: ФИО (по шаблону), специальности, курсу, среднему баллу (>n , диапазон). Сортировка по стажу группе и курсу. Пользовательский атрибут для валидации индекса.
2, 10	Поиск по: номеру, ФИО, балансу, типу вклада. Сортировка по типу вклада и дате открытия счета. Пользовательский атрибут для валидации паспортных данных владельца (паспорт РБ).

3, 11	Поиск по: лектору, семестру и курсу. Сортировка по количеству лекций и виду контроля. Пользовательский атрибут для валидации семестра.
4, 12	Поиск по: издательству, году издания, диапазону страниц. Сортировка по названию, дате загрузки. Пользовательский атрибут для валидации УДК.
5, 13	Поиск по: типу (количество комнат), году, району и городу. Сортировка по площади, количеству комнат, цене. Пользовательский атрибут для валидации индекса.
6, 14	Поиск по производителю и модели процессора. Сортировка по частоте работы процессора, размеру ОЗУ. Пользовательский атрибут для валидации процессора.
7, 15	Поиск по: авиакомпании, типу, количеству мест, грузоподъемности. Сортировка по году выпуска, дате последнего тех. обслуживания. Пользовательский атрибут для валидации ID самолета.
8, 16	Поиск по: названию, типу, диапазону цены. Сортировка по дате производства, стране производителя, затем по названию. Пользовательский атрибут для валидации инвентарный номер.

Краткие теоретические сведения

Регулярные выражения

Регулярные выражения – это язык для описания текста и внесения в него изменений. Регулярное выражение применяется к строке. Результатом применения является фрагмент строки, либо новая строка, либо группы подстрок, либо логический результат – в зависимости от того, какая операция выполняется.

Регулярные выражения очень мощный и в то же время простой механизм обработки текстовой информации. На данный момент наиболее полно они реализованы в язык Perl, хотя возникли гораздо раньше.

Для работы с регулярными выражениями в C# существует класс System.Text.RegularExpressions.Regex. Многие методы этого класса также существуют в двух версиях – статической и экземпляра.

У класса два конструктора с одним параметром строкового типа, определяющего правило обработки и с двумя – второй в этом случае задает параметры регулярного выражения (аналог опций в Perl).

В регулярных выражениях существует понятие «метосимвол», это аналог управляющей последовательности в строке. Если нам необходим символ как он есть, то перед ним ставится обратная наклонная черта. Некоторые символы наоборот

начинаются с обратной наклонной черты. Именно по этому для РВ с C# лучше использовать дословные строки.

() – определение группы

| - задание перечисления

{n, m} – предназначены для обозначения кратности. В общем виде количество больше или равно n, но меньше или равное m. {2, 7}

Существуют частные случаи: {n,} – не менее, {,m} – не более, ровно – {n}, а также метасимволы

* аналогичен {0,}

+ аналогичен {1,}

? аналогичен {0,1}

^ - начало строки

$ - конец строки

[] – обозначение класса символов

[abcxyz] – обозначачет любой из символов класса,

[a-cx-y], [a-z], – можно использовать интервалы

\d –числовой символ аналог[0-9]

\D – нечисловой символ (Регулярные выражения регистрозависимы)

\w – алфавитно-цифровой символ или знак подчеркивания аналог [a-zA-Z_]

\W – не алфавитно-цифровой символ или знак подчеркивания

\s – пробел

\S – не пробел

```
Regex r1 = new Regex(@" |, |,");
string []a3 = r1.Split(s5);
foreach(string s in a3)
    Console.WriteLine(s);
```

Метод Split РВ аналогичен строковому, но можно задать разделители состоящие более чем из одного символа (строки).

Для поиска в строке существуют два метода Match – ищет первое вхождение и Matches – ищет все вхождения. Первый возвращает объект класса Match, второй - коллекцию MatchCollection. Для выделения групп, как уже говорилось, используются круглые скобки

То что в круглых скобках и будет заносится в коллекцию MatchCollection.

```
Regex r2 = new Regex(@"(\w+)");
MatchCollection mc = r2.Matches(s5);
foreach(Match m in mc)
    Console.WriteLine(m.Value);
```

С помощью регулярных выражений можно проверять строку на соответствие какому-либо формату.

Вопросы:

1. Назовите классы, которые используются для создания меню. Перечислите свойства и методы.
2. Что может содержать строка состояния? Какие есть методы управления строкой состояния?
3. Что такое регулярные выражения? Где и как их можно использовать?
4. Что такое привязки (якоря) в RegEx? Приведите примеры.
5. Зачем используют конструкции группирования? Приведите примеры.
6. Что такое квантор или множители? Приведите примеры.
7. Напишите регулярное выражение для проверки номера МТС (Velcom).
8. Напишите регулярное выражение для проверки паспорта.
9. Напишите регулярное выражение для проверки даты.
10. Напишите регулярное выражение для проверки УДК
11. Напишите регулярное выражение для проверки Фамилии.
12. Напишите регулярное выражение для проверки пароля, логина.
13. Разберите

?<=<img .*?src\s*=\s*"")[^""]+(?="".*?>)"

"[a-zA-Za-яA-Я,-;:]{5,50}"

/^[a-z0-9_-]{3,16}$/

/^#?([a-f0-9]{6}|[a-f0-9]{3})$/

/^(https?:\/\/)?([\da-z\.-]+)\.([a-z\.]{2,6})([\/\w \.-]*)*\/?$/

/^(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$/

/^<([a-z]+)([^<]+)*(?:>(.*)<\/\1>|\s+\/>)$/