2014 г.

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«Владимирский государственный университет имени
Александра Григорьевича и Николая Григорьевича Столетовых»
(ВлГУ)**

УТВЕРЖДАЮ
Проректор по учебно-методической работе
_____ А.А. Панфилов
«17» 03 2016 г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ
Практикум по решению задач на ЭВМ

Направление подготовки 44.03.05 Педагогическое образование

Профиль подготовки «Информатика. Математика»

Уровень высшего образования бакалавриат

Форма обучения очная

Семестр	Трудоем-кость зач. ед,час.	Лекций, час.	Практ. зан., час.	Лаборат. работ, час.	СРС, час.	Форма промежуточного контроля (экз./зачет)
8	1 / 36	-	-	20	16	ЗАЧЕТ
9	4 / 144	-	-	24	120	ЗАЧЕТ
10	5 / 180	-	-	26	154	ЗАЧЕТ С ОЦЕНКОЙ
Итого	**10 / 360**	**-**	**-**	**70**	**290**	**2 ЗАЧЕТА, ЗАЧЕТ С ОЦЕНКОЙ**

Владимир, 2016

1. ЦЕЛИ И ЗАДАЧИ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Цели:

1. Формирование у студентов навыков работы с современными технологиями в программировании для решения прикладных задач.
2. Развитие операционного мышления направленного на выбор оптимальных действий, на умение планировать свою деятельность и предвидеть ее результаты.
3. Формирование опыта работы в коллективе, в частности рефлексии.

Задачи дисциплины:

- Сформировать навыки работы с программной платформой .NET Framework и реализации ООП парадигмы.
- Развитие принципов разработки алгоритмов и программ, их оптимизации.
- Изучение и использование различных методов программирования.
- Формирование опыта разработки алгоритмов и решения задач.
- Освоение понятий и принципов ООП в рамках платформы .NET Framework.

2. МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ОПОП ВО

Дисциплина «Практикум по решению задач на ЭВМ» относится к вариативной части учебного плана по направлению «Педагогическое образование».

Для освоения дисциплины студенты используют знания и умения, сформированные в процессе изучения таких дисциплин, как «Современные ИТ», «Программирование», «Функциональное программирование», «Теория алгоритмов», «Теоретическая информатика».

Освоение данной дисциплины способствует подготовке студентов к итоговой государственной аттестации.

3. КОМПЕТЕНЦИИ ОБУЧАЮЩЕГОСЯ, ФОРМИРУЕМЫЕ В РЕЗУЛЬТАТЕ ОСВОЕНИЯ ДИСЦИПЛИНЫ (МОДУЛЯ)

В результате освоения дисциплины формируются следующие компетенции:

Шифр компетенции	Расшифровка компетенции
ОК-6	способностью к самоорганизации и самообразованию;
ПК-1	готовностью реализовывать образовательные программы по учебным предметам в соответствии с требованиями общеобразовательных стандартов;
ПК-12	способностью руководить учебно-исследовательской деятельностью обучающихся.

В результате освоения дисциплины обучающийся должен демонстрировать следующие результаты образования:

Знать:

- теоретическую основу, важнейшие особенности и возможности программной платформы .NET Framework 4.5 (ОК-6 / ПК-1 / ПК-12);

- синтаксис языка программирования C# и реализация принципов ООП средствами .NET Framework (ПК-1);
- методы и приемы реализации алгоритмов на базе объектной и компонентной модели проекта (ОК-6 / ПК-1).

Уметь:

- моделировать научные и практические задачи средствами .NET Framework (ОК-6 / ПК-1);
- применять новые технологии на основе практических задач (ПК-1 / ПК-12);
- проецировать полученные знания для реализации педагогических задач в процессе обучения основам алгоритмизации и программирования (ОК-6 / ПК-1);
- осуществлять согласованную работу в коллективе из нескольких человек в целях достижения поставленной учебной задачи (ПК-12).

Владеть:

- приемами исследования математических задач средствами .NET Framework (ПК-1);
- навыками работы со справочными системами по технологии NET Framework и языку программирования C# (ПК-1).

4. СТРУКТУРА И СОДЕРЖАНИЕ ДИСЦИПЛИНЫ (МОДУЛЯ)

Общая трудоёмкость дисциплины составляет 10 зачетных единиц, 360 часов.

№ п/п	Раздел (тема) дисциплины	Семестр	Неделя семестра	Виды учебной работы, включая самостоятельную работу студентов и трудоемкость (в часах)						Объем учебной работы, с применением интерактивных методов (в часах / %)	Формы текущего контроля успеваемости (по неделям семестра), форма промежуточной аттестации (по семестрам)
				Лекции	Практические	Лабораторные	Контрольные работы, коллоквиум	СРС	КП/КР		
1	Статические компоненты.	8	1			2		2			
2	Структуры и перечисления	8	2			2		2		1/50%	
3	Строки и файлы	8	3-4			4		4		1/25%	Рейтинг-контроль №1
4	Обработка исключений	8	5			2		2		1/50%	

5	Наследование и полиморфизм	8	6			2		2			Рейтинг-контроль №2
6	Интерфейсы. Делегаты. События	8	7-8			4		2		1/25%	
7	Обобщения	8	9-10			4		2		1/25%	Рейтинг-контроль №3
Всего (8 семестр)						**20**		**16**		**5/20%**	**ЗАЧЕТ**
1	Перегрузка операторов.	9	1-2			4		20		1/25%	
2	Методы расширения.	9	3-4			4		20		1/25%	Рейтинг-контроль №1
3	Динамически подключаемые библиотеки.	9	5-6			4		20		1/25%	
4	Инкапсуляция на уровне класса.	9	7			2		10		2/100%	Рейтинг-контроль №2
5	Разработка проектов WindowsForms.	9	8-12			10		50		4/40%	Рейтинг-контроль №3
Всего (9 семестр)						**24**		**120**		**9/37.5%**	**ЗАЧЕТ**
1	Интерпретатор командной строки. Компиляция программ на языке C#. Пакетные файлы	10	1-2			4		8		2/50%	
2	Лямбда-выражения.	10	3			2		12		1/50%	Рейтинг-контроль №1
3	Технология LINQ. Основы. Операторы select, from, orderby, where Отбор запрашиваемых значений с помощью оператора where.	10	4-5			4		30		1/25%	
4	LINQ. Операторы group, into, join, let. Группирование результатов с помощью оператора group.	10	6-7			4		32		1/25%	Рейтинг-контроль №2
5	LINQ. Анонимные типы. Методы в LINQ.	10	8-9			8		24		2/50%	
6	Многопоточное программирование. Класс Thread. Классы Task и Parallel	10	10-13			8		48		4/100%	Рейтинг-контроль №3
Всего (10 семестр)						**26**		**154**		**11/42,3%**	**ЗАЧЕТ С ОЦЕНКОЙ**
ВСЕГО						**70**		**290**		**25/35.7%**	**ЗАЧЕТ С ОЦЕНКОЙ**

Темы и содержание лабораторных занятий

8 семестр

Тема 1. Статические компоненты

- Задачи, приводящие к необходимости введение статических компонентов.
- Модификатор static.
- Статичные поля класса.
- Статичные методы класса.
- Статичные классы.
- Примеры использования.

Тема 2. Структуры и перечисления

- Использование структур в качестве параметров методов.
- Использование структур в иерархии типов данных.
- Перечислимый тип как механизм повышения качества логики построения приложения.
- Обработка данных перечислимого типа.
- Решение задач.

Тема 3. Строки и файлы

- Методы класса System.String. Примеры обработки строк.
- Класс StringBuilder.
- Регулярные выражения.
- Пространство имен System.IO.
- Запись и чтение текстовых файлов.
- Работа с жестким диском.
- Решение задач.

Тема 4. Обработка исключений

- Перехват исключений.
- Класс Exception.
- Обработка многочисленных исключений.
- Решение задач.

Тема 5. Наследование и полиморфизм

- Основы наследования.
- Конструкторы класса в наследовании.
- Виртуальные методы.
- Абстрактные классы.
- Решение задач.

Тема 6. Интерфейсы. Делегаты. События

- Интерфейсы и их роль в иерархии типов.
- Интерфейсные свойства.
- Делегаты.
- Групповой вызов и адресация делегируемых методов.

- Делегаты Action<T> и Func<T>.
- События как методы обратного вызова.
- Связь событий и делегатов.
- Решение практических задач.

Тема 7. Обобщения

- Назначение обобщений и их применение.
- Обобщённые классы.
- Стек.
- Решение практических задач.

9 семестр

Тема 1. Перегрузка операторов

- Понятие перегрузки, ее применение в реальных задачах.
- Синтаксис перегрузок.
- Изучение возможности перегрузки операторов на базе реализации модуля класса комплексных чисел.

Тема 2. Методы расширения

- Роль методов расширения, дополнительный функционал.
- Методика использования методов расширения.
- Примеры создания и использования методов расширения.
- Решение задач.

Тема 3. Динамически подключаемые библиотеки

- Технология динамически подключаемых библиотек.
- Механизмы создания и подключения динамических библиотек.
- Вопросы производительности и безопасности технологии DLL.
- Решение практических задач.

Тема 4. Инкапсуляция на уровне класса

- Уровни доступа к компонентам класс и классам.
- Запечатанные классы.
- Методика работы с закрытыми полями, принцип «черного ящика»/
- Свойства.
- Решение задач.

Тема 5. Разработка проектов WindowsForms

- GUI приложения.
- Разработка GUI с помощью Visual Studio, SharpDevelop и сборка в ручном режиме.
- Структура шаблона GUI приложения.
- Визуальный конструктор формы.
- Компоненты класса Control.
- Разработка приложений.

10 семестр

Тема 1. Интерпретатор командной строки. Компиляция программ на языке C#. Пакетные файлы.

- Интерпретатор командной строки. Пакетные файлы.
- Компиляция программ без использования IDE.
- Решение задач.

Тема 2. Лямбда-выражения.

- Лямбда-выражения и лямбда-вычисления.
- Расширение синтаксиса C#. Практическое применение.
- Решение задач.

Тема 3. Технология LINQ. Основы. Операторы select, from, orderby, where Отбор запрашиваемых значений с помощью оператора where.

- Технология интегрированных запросов LINQ. Основы. Механизмы работы за-просов.
- Операторы select, from, orderby, where.
- Отбор запрашиваемых значений с помощью оператора where.
- Связь с SQL.
- Решение задач.

Тема 4. LINQ. Операторы group, into, join, let. Группирование результатов с помощью оператора group.

- Операторы group, into, join, let.
- Группирование результатов с помощью оператора group.
- Связывание нескольких источников данных.
- Решение задач.

Тема 5. LINQ. Анонимные типы. Методы в LINQ.

- Операторы group, into, join, let.
- Группирование результатов с помощью оператора group.
- Связывание нескольких источников данных.
- Решение задач.

Тема 6. Многопоточное программирование. Класс Thread.

- Многопоточное программирование.
- Потоки. Класс Thread. Запуск и остановка потоков.
- Решение задач.

Тема 7. Многопоточное программирование. Классы Task и Parallel

- Многопоточное программирование.
- Задачи. Классы Task и Parallel.
- Решение задач.

5. ОБРАЗОВАТЕЛЬНЫЕ ТЕХНОЛОГИИ

Изучение курса «ПРЗ на ЭВМ» предполагает сочетание лабораторных занятий и самостоятельной работы студентов.

На лабораторных занятиях, общий объем которых указан в тематическом плане, студенты изучают теоретический минимум, выполняют задания (индивидуально / попарно или в группах из нескольких человек), консультируются по самостоятельной работе с преподавателем.

Самостоятельная работа предполагает более детальное знакомство с теоретическим материалом и предварительную подготовку к новым лабораторным работам.

При изучении учебного материала данной дисциплины следующие технологии обучения: учебные групповые дискуссии: обсуждения задач (методы, приемы решения, выбор оптимального способа решения, количество возможных случаев для рассмотрения и т.п.), мозговой штурм, презентация микроисследований и их обсуждение, технология проблемного обучения.

6. ОЦЕНОЧНЫЕ СРЕДСТВА ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ УСПЕВАЕМОСТИ, ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ПО ИТОГАМ ОСВОЕНИЯ ДИСЦИПЛИНЫ И УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ СТУДЕНТОВ

6.1 ТЕКУЩИЙ КОНТРОЛЬ УСПЕВАЕМОСТИ

Примеры заданий для проведения рейтинг-контроля

8 семестр

Рейтинг-контроль №1

1. Разработайте статический класс MyMath с методами, позволяющими вычислять модуль числа, находить целую и дробную часть указанного числа, вычислять факториал.
2. Разработать класс Figure, описывающий геометрические фигуры на плоскости. Класс обладает свойствами «тип» и «площадь». Типы описать перечислениями и ограничить треугольником, прямоугольником и окружностью. Продемонстрировать применение экземпляров этого класса.

Рейтинг-контроль №2

1. Создайте интерфейс Supercar, наследующий интерфейс Luxury (роскошь) с дополненным методом moreInfo(), выводящим небольшое сообщение о суперкарах в целом.
2. Цель: Создать проект, реализующий абстрактные классы и методы для некоторых элементарных геометрических фигур в двумерном и трехмерном пространстве.
   Ход работы: Проект содержит два абстрактных класса Figure2 и Figure3, задающих абстрактные методы:

```
abstract class Figure2
{
```

```
    public abstract double Square();
    public abstract double Perimetr();
    public abstract void Type();
}

abstract class Figure3
{
    public abstract double Volume();  // объем
    public abstract double Surface(); // площадь поверхности
    public abstract void Type();
}
```

Первые два метода определяют числовые характеристики, последний – тип фигуры в своем классе.

Описать несколько классов, наследующих абстрактные. Каждый класс должен содержать конструктор, получающий в качестве параметров длины сторон (ребер), радиусов и т.п. в зависимости от типа фигуры. Например:

```
class Triangle: Figure2
{
    // реализация
}
// ...  ...  ...
Triangle Tr1 = new Triangle(2,1,2);
Console.WriteLine("Тип треугольника: ");
Tr1.Type();
Console.WriteLine("P={0}   S={1}",Tr1.Perimetr(),Tr1.Square());
// При выводе на консоль получим сообщение:
// Тип треугольника:
// равнобедренный
// P=5  S=0,96...
```

В качестве классов-фигур рассмотреть треугольник, трапецию (обобщает в т.ч. параллелограмм, ромб, прямоугольник, квадрат), круг; прямоугольный параллелепипед, цилиндр, шар.

Рейтинг-контроль №3

1. Создайте приложение WindowsForms. Поместите текстовое поле на форму. Требуется, чтобы во время работы приложения в заголовке окошка отображался текущий размер формы, а в текстовом поле – ее положение. Любое изменение размера и положения должно автоматически отслеживать соответствующими событиями.
2. Создайте приложение WindowsForms. Поместите на форму кнопку. При наведении курсора на область кнопки она должна изменить положение случайным образом. Если пользователю удается нажать на нее, то выводится сообщение о победе в игре.

9 семестр

Рейтинг-контроль №1

1. Разработать класс, реализующий операции над векторами в трехмерном евклидовом пространстве. С помощью перегрузки операторов определить следующие операции над векторами:
   - сложение / разность;
   - умножение на скаляр;
   - скалярное умножение векторов;
   - метод, вычисляющий модуль вектора.
2. Объясните, какую роль играет ссылка this в контексте создания следующего метода расширения:

```
public static bool Contains_Sim(this String str, char sim)
```

Рейтинг-контроль №2

1. Создайте библиотеку динамической компоновки, реализующую класс SortArray с методами, осуществляющими пузырьковую сортировку и сортировку простым выбором. Во внешней программе продемонстрировать подключение и использование возможностей библиотеки.
2. Разработать библиотеку динамической компоновки, реализующую рациональные числа, а также операции сложения и разности (с помощью перегрузки операторов). Во внешней программе продемонстрировать подключение и использование возможностей библиотеки.

Рейтинг-контроль №3

1. Разработать GUI приложение, решающее квадратное уравнение. Заявленный визуальный интерфейс:

2. Дополнительно, обработать ситуацию, при которой в случае ввода в текстовые поля нечисловых значений они сбрасываются, выводится сообщении о недопустимости значения, и выставляются некоторые значения по умолчанию.

10 семестр

Рейтинг-контроль №1

Описать нижеприведенные делегаты, создать совместные к ним лямбда-выражения и продемонстрировать работу на 2-3х экземплярах этих делегатов:

- delegate byte RNDNumber(); - лямбда-выражение возвращает случайное целое число в промежутке [10,20];
- delegate bool Matrix(byte n,int[,] M); - возвращает true, если матрица M порядка n единичная.

Что такое лямбда-выражения и для чего их применяют? Указать основные приемы создания лямбда-выражений, особенности конструкций.

Рейтинг-контроль №2

1. Создать запросы и вывести результат:
   - Из целочисленного массива отобрать элементы, кратные 10.
   - Отобрать из целочисленного массива значения, лежащие в промежутке (0;5] или [8;10].
   - В строковом массиве отобрать слова, начинающиеся и оканчивающиеся одинаковым символом.
   - Произвести сортировку массива вещественных чисел по убыванию.
   - Из массива вещественных чисел отобрать только положительные и возвратить их квадраты.
   - Из массива целых чисел отобрать числа n и возвратить n*5, если n кратно 5, n*7, если n кратно 7, либо n*100 для других значений n.
2. Изучить и воспроизвести лекционный пример многокритериальной сортировки. Каковы особенности описания дополнительного класса Account? Что можно сказать о его полях (свойствах)? Как происходит инициализация экземпляров этого класса.
3. Основываясь на пункте 2), придумать и разработать свой пример использования известных вам операторов LINQ при обработке экземпляров класса (либо структур). Проект должен включать несколько запросов на условную выборку данных (по усмотрению пользователя), запрос на сортировку глубиной не менее 2-х критериев.
4. Дана таблица:

Уникальный номер авто	Название марки
2345	BMW
1235	AUDI
432	BMW
. . .	. . .

   - Требуется составить запрос, выводящий строки таблицы, предварительно сгруппированные по названию марки автомобиля.
   - Преобразовать запрос таким образом, чтобы внутри каждой группы уникальные номера отображались в порядке возрастания; при этом алфавитный порядок вывода самих групп должен сохраниться.
   - Доработать предыдущий запрос так, чтобы на экран выводились группы, содержащие не менее 2 записей в каждой.

1) Источником данных является массив предложений (строк). Требуется составить запрос, выбирающий из каждого слова только те символы, чей ASCII код лежит в промежутке [97; 100]. Эти символы поместить в результат вывода.

2) Объединение нескольких источников данных. Преобразуйте лекционный пример с оператором join: новая запись (экземпляр класса Rezult) создается лишь в том случае, когда стоимость автомобиля не превышает 750000 р. При этом марки авто на выводе следуют в алфавитном порядке и по мере роста цены.

3) Используя возможности анонимных типов и методов LINQ, перепишите задачу на объединение нескольких источников данных (см. лекцию 4, оператор join; лекция 5, теория), демонстрирующую эти возможности. А именно – реализацию анонимного типа через стандартный синтаксис, с использованием инициализатора проекции, замену оператора join на соответствующий LINQ метод.

4) Строительной фирме требуется приложение, способное работать с несколькими источниками данных. В качестве источников выступают две таблицы. Первая – Dispatcher, содержащая логин, фамилию, имя, отчество рабочего, а также поля с личной характеристикой и контактной информацией. Вторая – Orders, хранящая поступающие заказы; ее поля: номер заказа, логин рабочего (уникален для каждого), описание заказа (услуги), стоимость услуги. Требуется создать запрос, объединяющий эти две таблицы и выводящий номер заказа, фамилию и имя рабочего, описание услуги, ее стоимость.

Рейтинг-контроль №3

1. Создать приложение с двумя дополнительными потоками. Все потоки (в т.ч. основной) производят возведение матрицы в квадрат, матрица задается случайным образом, размер матрицы $n \sim 10^3$.
2. Привести пример создания потоков через экземпляры класса и с использованием статических методов и полей.
3. Создать и запустить массив потоков. Каждому из потоков отвести время ожидания, имитирующее работу с данными. Организовать работу потоков таким образом, чтобы основной поток дождался завершения выполнения всех дополнительных.
4. Разработать приложение, обрабатывающее несколько потоков. Каждому из потоков установить приоритет выполнения. «Прогнать» программу на выполнение потоками одной и той же длительной операции, например вложенных циклов. Определить, насколько полученные результаты соответствуют теоретически возможным.
5. Доработать лекционный пример обработки ошибок, добавив другие возможные ошибки.
6. Создать базу данных, содержащую информацию о некотором предприятии. К этой базе практически одновременно могут обращаться различные пользователи (потоки). Требует синхронизировать доступ по чтению и записи данных, избежав нарушения их корректности.

7. Разработать приложение, применяющее мьютекс и семафор для синхронизации потоков.
8. Разобрать и реализовать примеры аварийного завершения потоков. Продемонстрировать аварийную ситуацию и существующие возможности завершения приложения без потери основных результатов, полученных в других процессах.

Перевести ряд заданий из LINQ на PLINQ. Произвести сравнение скорости обработки запросов. В каких случаях распараллеливание оказывается действительно эффективным? Ответ обосновать.

6.2 ПРОМЕЖУТОЧНАЯ АТТЕСТАЦИЯ ПО ИТОГАМ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Вопросы к зачету 8-го семестра

1. Статические компоненты
2. Задачи, приводящие к необходимости введение статических компонентов.
3. Модификатор static. Статичные поля класса.
4. Статичные методы класса.
5. Статичные классы. Примеры использования.
6. Структуры и перечисления.
7. Использование структур в качестве параметров методов.
8. Использование структур в иерархии типов данных.
9. Перечислимый тип как механизм повышения качества логики построения приложения.
10. Обработка данных перечислимого типа.
11. Строки и файлы.
12. Методы класса System.String. Примеры обработки строк.
13. Класс StringBuilder.
14. Регулярные выражения.
15. Пространство имен System.IO.
16. Запись и чтение текстовых файлов.
17. Работа с жестким диском.
18. Обработка исключений и перехват исключений.
19. Класс Exception.
20. Обработка многочисленных исключений.
21. Наследование и полиморфизм
22. Конструкторы класса в наследовании.
23. Виртуальные методы.
24. Абстрактные классы.
25. Интерфейсы и их роль в иерархии типов.
26. Интерфейсные свойства.
27. Делегаты.
28. Групповой вызов и адресация делегируемых методов.

29. Делегаты Action<T> и Func<T>.
30. События как методы обратного вызова.
31. Связь событий и делегатов.
32. Обобщения.
33. Назначение обобщений и их применение.
34. Обобщённые классы.
35. Стек на обобщенном типе.

Вопросы к зачету 9-го семестра

1. Понятие перегрузки операторов. Математические задачи, реализуемые с помо-щью перегрузки.
2. Синтаксис перегрузок.
3. Перегрузка операторов на базе реализации модуля класса комплексных чисел.
4. Перегрузка операторов на базе реализации модуля класса матриц.
5. Методы расширения. Роль методов расширения, дополнительный функционал.
6. Методика использования методов расширения.
7. Примеры создания и использования методов расширения.
8. Динамически подключаемые библиотеки. Технология DLL.
9. Механизмы создания и подключения динамических библиотек.
10. Вопросы производительности и безопасности технологии DLL.
11. Инкапсуляция на уровне класса.Уровни доступа к компонентам класс и классам.
12. Запечатанные классы.
13. Методика работы с закрытыми полями, принцип «черного ящика».
14. Свойства.
15. Разработка проектов WindowsForms.
16. GUI приложения.
17. Разработка GUI с помощью Visual Studio, SharpDevelop и сборка в ручном ре-жиме.
18. Структура шаблона GUI приложения.
19. Визуальный конструктор формы.Компоненты класса Control.
20. Реализация меню.

Вопросы к зачету с оценкой для 10-го семестра

1. Лямбда-выражения. Примеры использования.
2. Лямбда-выражения и анонимные методы. Достоинства и недостатки.
3. Понятие LINQ. Методика конструирования запросов. Примеры.
4. LINQ. Операторы select, from, orderby, where Отбор запрашиваемых значений с помощью оператора where.
5. LINQ. Операторы group, into, join, let. Группирование результатов с помощью оператора group.C#.
6. LINQ. Анонимные типы. Методы в LINQ.
7. Многопоточное программирование. Основные понятия и концепции. Класс Thread.

8. Многопоточное программирование. Примеры приложений с применением класса Thread.
9. Аварийное завершение потоков.
10. Потоки и задачи.
11. Класс Task.
12. Класс Parallel.
13. PLINQ. Примеры.

6.3 МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ СТУДЕНТОВ

Вопросы для самостоятельной работы

8 семестр

1. Статичные классы в пространстве имен System.
2. Строковые операции. Исследование задач, требующих работы со строками.
3. Регулярны выражения. Преимущества и недостатки.
4. Абстрактные классы. Реализация математических моделей.
5. Наследование и полиморфизм. Целесообразность наследования при реализации собственных классов и сборок.
6. Делегаты. Принципы построения. Связь с указателями на методы в языке C++. Вопросы безопасности.
7. Интерфейсы и абстрактные классы.
8. Обобщения как средство расширения возможностей работы с компонентами.

9 семестр

1. Реализация абстрактной модели векторных пространств и основных операций над векторами средствами C#.
2. Модель рационального числа. Реализация операций над рациональными числами
3. Проблемы использования DLL.
4. Использование библиотеки .NET Framework для получения информации о компьютере.
5. Основные классы для работы с сетью Интернет. Примеры подключения и обработки запросов.
6. Работа с графическими примитивами.
7. Технология GDI+.
8. Программирование GUI в рамках школьного курса. Проблемы и трудности в реализации подхода.

10 семестр

1. История лямбда-выражений.

2. LINQ как инструмент работы с базами данных.
3. Работа с потоками. Вопросы блокировки.
4. Проблемы распараллеливания. Оптимизация вычислений. Результаты тестов.
5. Мьютексы и семафоры.
6. PLINQ. Основные возможности.

Примеры заданий для проектной деятельности

8 семестр

1. Реализуйте класс Complex, позволяющий работать с комплексными числами в удобной форме. Класс позволяет:
    - создавать комплексные числа (с помощью параметризированного конструктора);
    - складывать, вычитать, умножать и делить числа (с помощью статических методов);
    - вычислять модуль и аргумент числа;
    - выводить результат в удобной форме.
2. Реализуйте класс Matrix, позволяющий работать с квадратными матрицами. Класс позволяет:
    - создавать матрицы по указанной размерности или массиву;
    - складывать, вычитать, умножать, матрицы, домножать матрицу на число.
    - выводить результат в удобной форме.

9 семестр

Разработать класс, реализующий основные операции над квадратными матрицами. С помощью перегрузки операторов определить следующие операции:

- сложения;
- разности;
- умножения матрицы на матрицу;
- умножение матрицы на число;
- изменение знака на противоположный.

Перегрузить оператор равенства, проверяющий матрицы на равенство.

Предполагается, что операции совершаются над матрицами одинаковой размерности. В противном случае возвращается ложный результат.

В классе определить два конструктора: первому передается двумерный массив, отвечающий за матрицу, а второму передается число (размерность матрицы) и создается матрица из нулей.

10 семестр

После изучения основ LINQ учащимся предлагается разработать проект, позволяющий подключаться к MS SQL Server / MySQL для выполнения запросов, добавления, удаления и редактирования данных выбранной БД посредством LINQ.

Проекты выносятся на обсуждение. Наиболее удачные рекомендуются для участия в научной конференции ВлГУ.

Задания

1. Смоделируйте базу данных для небольшой фирмы и реализуйте с помощью LINQ запросы на выборку данных. В запросах требуется продемонстрировать возможности изученных вами операторов и методов LINQ: выборка данных, условная выборка, фильтрация данных, сортировка, группировка; вариативность синтаксиса.
2. Составить примеры программ, позволяющие наглядно проследить за работой основных и фоновых потоков. Показать, какие изменения вносит установка приоритета потока. Какие механизмы позволяют отслеживать окончание потока? Продемонстрировать ситуации, в которых возможно получение неверных результатов из-за неправильного завершения потока.
3. Разработать приложение, производящее синхронизацию запущенных в нем потоков. Среди инструментов должен присутствовать оператор lock. Дополнительно продемонстрировать работу мьютекса и семафора.
4. Какие возможности предоставляет класс Parallel? Приведите примеры программ, обрабатывающие параллельно несколько методов. Источники данных следует брать достаточно большими (по количеству элементов, объему данных). Сравнить полученные результату с однопоточным выполнением, сделать соответствующие выводы.

Провести сравнительную характеристику работы средств LINQ и PLINQ с одинаковыми источниками данных и по возможности на различных процессорах. При необходимости воспользоваться возможностями таймера. Результаты работы программы могут быть представлены в форме небольшого отчета.

Пример дополнительного материала для организации самостоятельной работы

8 семестр

Обработка поведения элементов перечисления

Перечисления задают логику построения, но не определяют поведение элементов объекта в каждом конкретном значении. Эта работа должна быть проделана отдельно.

Обычно все элементы перечисления равнозначны в плане реализации. Т.е. алгоритм их обработки одинаков, а разница лишь в итоговом значении.

Так, в нашей задаче определено два перечисления, отвечающие за тип фигуры и ее цвет, то однотипная обработка очевидна. Если мы хотим на выводе получать названия цветов и типов фигур на родном языке, то необходимо добавить функцию, которая для каждого значения задаст корректное соотвествие.

```
using System;

enum FigureType { Triangle, Rectangle, Circle };
enum FigureColor { Red, Green, Blue, Black };

class Figure
{
    public FigureType Type { get; set; }
    public FigureColor Color { get; set; }

    public Figure(FigureType type, FigureColor color)
    {
        Type = type;
        Color = color;
    }

    // метод выводит общие данные по объекту
    public void Info()
    {
        Console.WriteLine("Тип: " + GetFigureType(Type));
    }

    // метод возвращает название указанного цвета на русском языке
    private string GetFigureType(FigureType type)
    {
        switch (type)
        {
            case FigureType.Triangle:
                return "Треугольник";
            case FigureType.Rectangle:
                return "Квадрат";
            default:
                return "Круг";
        }
    }
}

class Program
{
    static void Main()
    {
        // создание объекта и инициализация его свойств
        Figure fig = new Figure(
            FigureType.Rectangle,
            FigureColor.Black
        );

        fig.Info();

        Console.ReadKey();
    }
}
```

Обратите внимание, что методу GetFigureType определен закрытый доступ. Таким образом мы скрываем его работу внутри класса и запрещаем внешний вызов метода (что и нежелательно).

Разумеется, аналогичный механизм необходимо проделать и для элементов перечисления FigureColor.

Инкапсуляция по смыслу

Последнее, что необходимо отметить в нашей задаче – достаточно ограниченное распространение типов FigureType и ColorType. Класс Figure весьма узкопрофилирован, и нет смысла распространять перечисления на другие классы.

Разумная идея – внедрить перечисления FigureType и ColorType внутри класса Figure.

Перечисления

```
using System;

class Figure
{
    public enum FigureType { Triangle, Rectangle, Circle };
    public enum FigureColor { Red, Green, Blue, Black };

    // свойства, конструктор, вывод и т.д.
    // . . .
}

class Program
{
    static void Main()
    {
        // создание объекта и инициализация его свойств
        Figure fig = new Figure(
            Figure.FigureType.Rectangle,
            Figure.FigureColor.Black
        );

        fig.Color = Figure.FigureColor.Green;

        fig.Info();

        Console.ReadKey();
    }
}
```

Перечисления в примере являются уже элементами класса, поэтому его необходимо будет указывать при каждом обращении к значению перечисления.

9 семестр

Класс ImageBox

Элемент управления PictureBox применяется для отображения графических изображений. Изображение может быть в формате BMP, JPEG, GIF, PNG, метафайла или пиктограммы.

Свойство Image задает ссылку на изображение:

```
public Image Image { get; set; }
```

Из одноименного класса Image отдельно выделим статический метод FromFile, загружающий изображение по указанному пути.

Свойство SizeMode определяет способ отображения внутри рамки:

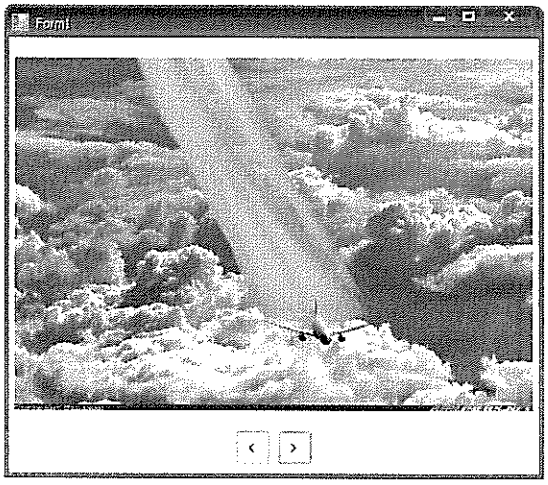
```
public PictureBoxSizeMode SizeMode { get; set; }
```

где перечисление PictureBoxSizeMode задает следующие варианты:

```
public enum PictureBoxSizeMode = {
      AutoSize, CenterImage, Normal, StretchImage, Zoom
};
```

Form1.cs

Следующее приложение позволяет просматривать изображения в текущей директории.




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

namespace WF_2
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private string[] files;      // массив путей к файлам изображений
        private int index = 0;       // индекс текущего изображения

        private void Form1_Load(object sender, EventArgs e)
        {
            // директория с изображениями
```

```
            string path =
                @"D:\Мои документы\Visual Studio 2010\Projects\WF_2\WF_2\Pictures";
            // получить массив из имен файлов с расширением "jpg"
            files = System.IO.Directory.GetFiles(path, "*.jpg");
            // загрузить первое изображение
            this.pictureBox1.Image = Image.FromFile(files[0]);
        }

        private void button1_Click(object sender, EventArgs e)
        {
            if (index > 0) index--; else index = files.Length - 1;
            this.pictureBox1.Image = Image.FromFile(files[index]);
        }

        private void button2_Click(object sender, EventArgs e)
        {
            if (index < files.Length - 1) index++; else index = 0;
            this.pictureBox1.Image = Image.FromFile(files[index]);
        }
    }
}
```

При настроке свойств формы объекту PictureBox рекомендуется задать якорь со всех сторон, чтобы при изменеии размера формы изображение масшабировалось автоматически.

Класс необходимо расширить двумя полями. Первое хранит пути к изображениям: их можно получить с помощью метода GetFiles() класса Directory. Второе поле – индекс текущего изображения; он будет увеличиваться либо уменьшаться при прокрутке. На этом уровне переменные будут общедоступны остальным управляющим элементам формы, в частности кнопкам.

10 семестр

В своей основе Parallel LINQ, он же PLINQ — это версия LINQ to Objects, в которой объекты исходного перечисления обрабатываются параллельно.

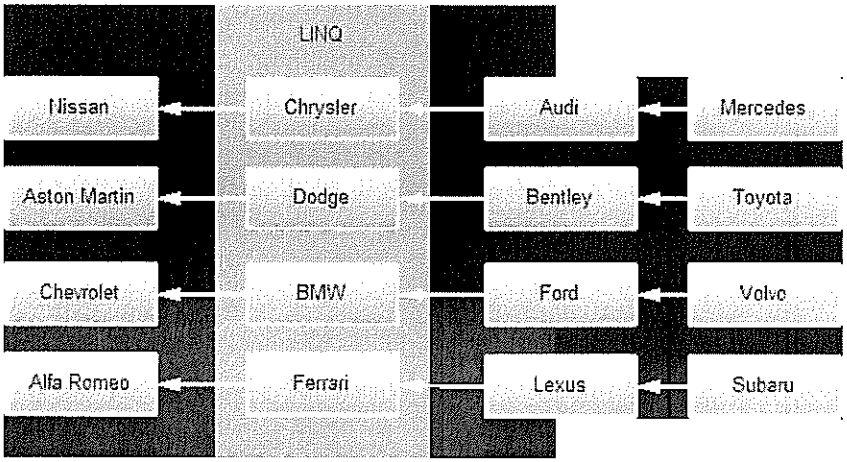
В версии .NET 4 появился целый набор расширенных средств для упрощения параллельного программирования. Средства параллельного программирования существуют уже в течение длительного времени, но они были сложны в применении и многие программисты испытывали затруднения в их эффективном использовании. Средства .NET 4 были разработаны в расчете на более широкую аудиторию, и опираются на преимущества широкого распространения многопроцессорных и многоядерных машин.

Если посмотреть на исходный запрос в примере ниже можно заметить, что в нем имя каждого названия машины обрабатывается по очереди:

```
string[] cars = { "Nissan", "Aston Martin", "Chevrolet",
                  "Alfa Romeo","Chrysler", "Dodge", "BMW",
                  "Ferrari", "Audi", "Bentley", "Ford",
                  "Lexus", "Mercedes", "Toyota", "Volvo",
                  "Subaru", "Жигули"};
```

```
string auto = cars.Where(p => p.StartsWith("S")).First();
Console.WriteLine(auto);
```

На рисунке проиллюстрировано, как это работает:



Механизм LINQ начинает с того, что проверяет, не начинается ли "Nissan" с "S". Затем переходит к "Aston Martin" и осуществляет проверку снова, после чего проверяет "Chevrolet", "Alfa Romeo", "Chrysler" и т.д. LINQ проходит последовательно по всем именам в массиве. Разумеется, это называется последовательным выполнением. Проблема последовательного выполнения состоит в том, что в каждый момент времени используется только одно ядро или один центральный процессор (далее будем говорить только о ядрах, но подразумеваться и то, и другое). На четырехядерных машинах три из четырех ядер не делают ничего при выполнении LINQ подобным образом.

Parallel LINQ изменяет правила игры, разбивая исходные данные на части и обрабатывая их параллельно, как показано ниже:

```
string[] cars = { "Nissan", "Aston Martin", "Chevrolet",
                  "Alfa Romeo","Chrysler", "Dodge", "BMW",
                  "Ferrari", "Audi", "Bentley", "Ford",
                  "Lexus", "Mercedes", "Toyota", "Volvo",
                  "Subaru", "Жигули"};

string auto = cars.AsParallel().Where(p => p.StartsWith("S")).First();
Console.WriteLine(auto);
```

Названия "Nissan", "Aston Martin", "Chevrolet" и "Alfa Romeo" обрабатываются одновременно — каждый в отдельном ядре машины. После того, как каждое ядро завершит обработку имени, оно переходит к следующему, независимо от других ядер. Parallel LINQ заботится о разбиении данных, определяя, сколько элементов может быть обработано одновременно (хотя обычно принимается решение, что одного элемента данных на ядро достаточно), и координируя работу ядер таким образом, что результат получается так, как от любого другого запроса LINQ.

Так зачем вообще связываться с Parallel LINQ? Ответ прост — производительность. Взгляните на пример ниже. В нем определены два запроса LINQ, которые делают одно и то же. Один запрос последовательный, а другой использует Parallel LINQ. Оба запроса select выбирают четные числа между 0 и Int32.MaxValue и подсчитывают количество совпадений. Для генерации последовательности целочисленных значений применяются методы

Enumerable.Range и ParallelEnumerable.Range. Диапазоны будут рассматриваться в одной из следующих статей, а пока просто имейте в виду, что оба метода создают IEnumerable<int>, содержащую все нужные целочисленные значения.

```
using System.Diagnostics;
...

      // Создать последовательный диапазон чисел
      IEnumerable<int> nums1 = Enumerable.Range(0, Int32.MaxValue);

      // Запустить секундомер
      Stopwatch sw = Stopwatch.StartNew();

      // Выполнить запрос LINQ
      int sum1 = (from n in nums1
                  where n % 2 == 0
                  select n).Count();

      Console.WriteLine("Результат последовательного выполнения: " + sum1 +
"\nВремя: " + sw.ElapsedMilliseconds + " мс\n");

     // Создаем параллельный диапазон чисел
     IEnumerable<int> nums2 = ParallelEnumerable.Range(0, Int32.MaxValue);

     // Перезапускаем секундомер
     sw.Restart();

     // Выполняем параллельный запрос LINQ
     int sum2 = (from n in nums2.AsParallel()
                 where n % 2 == 0
                 select n).Count();

     Console.WriteLine("Результат параллельного выполнения: " + sum2 +
          "\nВремя: " + sw.ElapsedMilliseconds + " мс");
```

Parallel LINQ предназначен для объектов

Как уже говорилось, Parallel LINQ — это параллельная реализация API-интерфейса LINQ to Objects. Поэтому он выполняет запросы LINQ to Objects в параллельном режиме. Он не реализует параллельных средств для других видов LINQ.

Это не значит, что с помощью Parallel LINQ не удастся обработать результаты других разновидностей запросов LINQ (например, выбрать объекты Order из базы данных Northwind с использованием LINQ to Entities или LINQ to SQL, а затем применить Parallel LINQ для дальнейшей обработки результатов), но Parallel LINQ не работает ни на чем, кроме объектов.

Далеко не все запросы LINQ to Objects являются хорошими кандидатами для запросов Parallel LINQ. Есть еще накладные расходы, связанные с разбиением данных на фрагменты, установкой и управлением классами, выполняющими параллельные задачи. Если запрос не слишком долго выполняется последовательно, возможно, не имеет смысла выполнять его параллельно, т.к. накладные расходы могут свести на нет весь выигрыш в производительности.

Для использования Parallel LINQ никаких специальных шагов предпринимать не понадобится. Все ключевые классы содержатся в пространстве имен System.Linq, где также находятся и все обычные классы LINQ to Objects. Наиболее важные методы и ключевые операции PLINQ будут описаны далее.

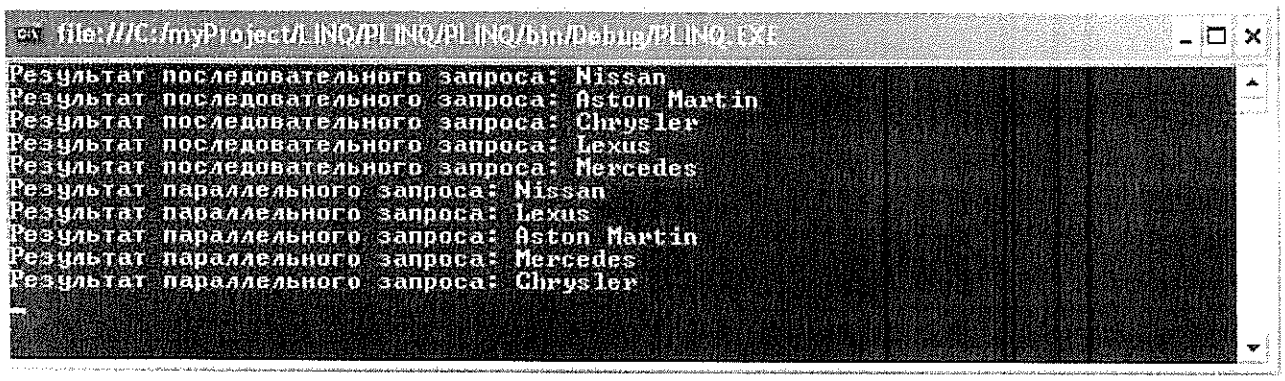
Запросы Parallel LINQ

По большей части использование Parallel LINQ, обычно называемого PLINQ, очень похоже на применение LINQ to Object. Фактически это одна из привлекательных сторон PLINQ. В обычном запросе LINQ to Querey источником данных является IEnumerable<T>, где T — обрабатываемый тип данных. Механизм LINQ автоматически переключается на использование PLINQ, когда источником данных является экземпляр типа ParallelQuery<T>. И здесь есть один трюк: любой IEnumerable<T> может быть преобразован в ParallelQuery<T> просто с использованием метода AsParallel. Давайте рассмотрим код. Ниже показаны запросы LINQ to Object и PLINQ, которые делают одно и то же:

```
string[] cars = { "Nissan", "Aston Martin", "Chevrolet", "Alfa Romeo",
                  "Chrysler", "Dodge", "BMW", "Ferrari", "Audi",
                  "Bentley", "Ford", "Lexus", "Mercedes",
                  "Toyota", "Volvo", "Subaru", "Жигули"};

    // Последовательный запрос LINQ
IEnumerable<string> auto = cars.Where(p => p.Contains("s"));

foreach (string s in auto)
    Console.WriteLine("Результат последовательного запроса: " + s);

    // Запрос Parallel LINQ
auto = cars.AsParallel().Where(p => p.Contains("s"));

foreach (string s in auto)
    Console.WriteLine("Результат параллельного запроса: " + s);
```

Первый запрос использует обычный LINQ to Objects для обработки каждой машины с целью нахождения названий, содержащих букву "s". В качестве результата получается IEnumerable<string> и все подходящие имена выводятся на консоль.

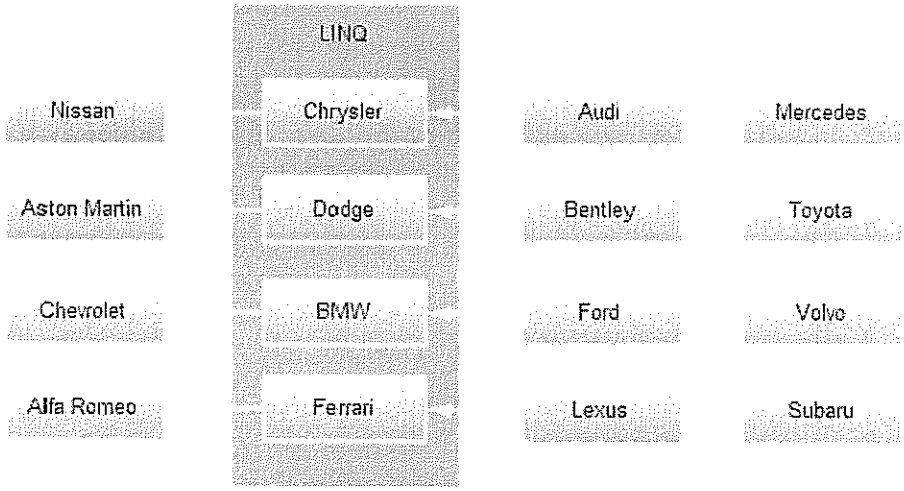
Второй запрос делает то же самое, но вдобавок вызывается метод AsParallel. С его помощью источник данных преобразуется в ParallelQuery, что автоматически подразумевает применение Parallel LINQ. И как явно следует из кода, никаких других изменений не требуется. Просто вызвав AsParallel, мы получаем PLINQ:



Известно, что это не идеальный запрос для применения с PLINQ. В предыдущей статье объяснялось, что накладные расходы в небольших простых запросах могут привести к ухудшению производительности по сравнению с последовательным выполнением. Однако нужно было продемонстрировать средства, и чем меньше времени будет потрачено на избыточно сложные примеры, тем проще понять то, что планировалось донести.

Предохранение порядка результатов

Данные, представленные в качестве источника для запроса PLINQ, разбиваются на части и разделяются для параллельной обработки. Несколько разделов могут обрабатываться одновременно. Однако каждый из этих разделов обрабатывается последовательно. Задумайтесь об этом - параллельное выполнение происходит из последовательной обработки нескольких разделов данных одновременно. Это показано на рисунке:



Когда PLINQ разделяет данные, получается нечто вроде показанного на рисунке, но в этом нельзя быть уверенным, потому что механизм PLINQ анализирует запрос и данные, выполняя разбиение "за кулисами". Но давайте предположим, что есть то, что показано на рисунке, т.е. множество разделов, каждый из которых содержит имена пяти машин. PLINQ назначает один раздел для обработки одному из ядер машины, и каждое ядро обрабатывает назначенный ему раздел последовательно.

Таким образом, продолжая пример, первое ядро проверит Nissan на содержание буквы s. Затем будет выполнена проверка Chrysler, Audi, Mercedes. Пока это происходит, второе ядро проверяет AstonMartin, Dodge, и т.д. В то же время третье и четвертое ядра обрабатывают свои разделы.

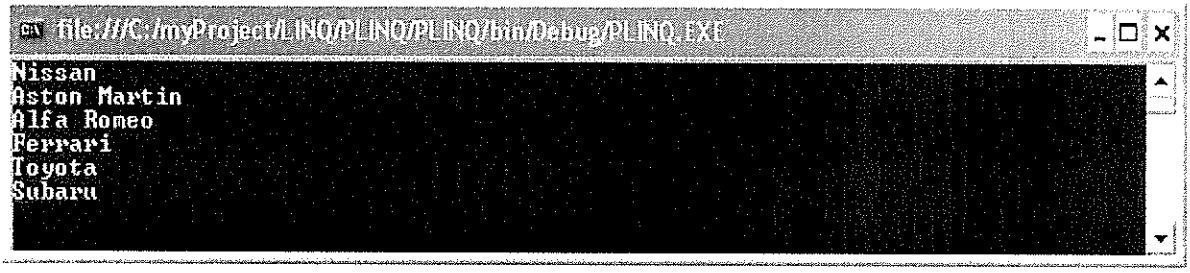
Найденное соответствие добавляется в результирующий набор. На рисунке наглядно видно, что если элементы обрабатывались примерно в одно и то же время каждым из ядер, то первым результатом будет Nissan, за ним последуют Aston Martin, Lexus, и т.д.

Иногда порядок результата не важен. Например, если необходимо только узнать, сколько названий машин содержат букву "s", порядок формирования результата значения не имеет. Однако бывают случаи, когда приходится заботиться о порядке результата. Это особенно верно при преобразовании существующих запросов LINQ в PLINQ. Например, где-то могут существовать предположения о порядке результатов. Чтобы сохранить порядок, необходимо воспользоваться расширяющим методом **AsOrdered** на объекте ParallelQuery, который создан посредством метода AsParallel.

Давайте рассмотрим пример:

```
string[] cars = { "Nissan", "Aston Martin", "Chevrolet", "Alfa Romeo",
                  "Chrysler", "Dodge", "BMW", "Ferrari", "Audi",
                  "Bentley", "Ford", "Lexus", "Mercedes", "Toyota",
                  "Volvo", "Subaru", "Жигули"};

// Запрос Parallel LINQ с сохранением порядка
   IEnumerable<string> auto = cars.AsParallel().AsOrdered()
               .Where(p => p.Contains("a"));
   foreach (string s in auto)
       Console.WriteLine(s);
```

Для расширяющих методов AsParallel и AsOrdered ключевых слов выражений запросов не предусмотрено. Эти методы должны вызываться непосредственно. В приведенном примере ключевые слова запросов смешаны с вызовами расширяющих методов. Если скомпилировать и запустить код, получится следующий результат:



Метод AsOrdered очень полезен, но не стоит привыкать вызывать его автоматически, поскольку он требует от PLINQ выполнения дополнительной работы по упорядочиванию результатов. Учитывая то, что главной целью PLINQ является повышение производительности, следует избегать лишней работы, где это возможно.

7. УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ (МОДУЛЯ)

Основная литература:

1. Златопольский Д.М. Программирование. Типовые задачи, алгоритмы, методы [Электронный ресурс]/ Златопольский Д.М.— Электрон. текстовые данные.— М.: БИНОМ. Лаборатория знаний, 2015.— 224 с. Режим доступа: http://www.iprbookshop.ru/12264
2. Объектно-ориентированное программирование с примерами на C#: Учебное пособие / Хорев П.Б. - М.: Форум, НИЦ ИНФРА-М, 2016. - 200 с.: 70x100 1/16. - (Высшее образование: Бакалавриат) (Обложка) ISBN 978-5-00091-144-0. Режим доступа: http://znanium.com/bookread2.php?book=529350
3. Базовые средства программирования на Visual Basic в среде VisualStudio Net. Практикум: Учебное пособие / Шакин В.Н. - М.: Форум, НИЦ ИНФРА-М, 2015. - 288 с.: 70x100 1/16. - (Высшее образование: Бакалавриат) (Переплёт 7БЦ) ISBN 978-5-00091-054-2. Режим доступа: http://znanium.com/bookread2.php?book=502047

Дополнительная литература

1. Практикум на ЭВМ. Часть 1 [Электронный ресурс]: учебное пособие/ — Электрон. текстовые данные.— М.: Евразийский открытый институт, 2012.— 263 c. Режим доступа: http://www.iprbookshop.ru/14644
2. Алгоритмизация и программирование : Учебное пособие / С.А. Канцедал. - М.: ИД ФОРУМ: НИЦ ИНФРА-М, 2014. - 352 с.: ил.; 60x90 1/16. - (Профессиональное образование). (переплет) ISBN 978-5-8199-0355-1, 500 экз. Режим доступа: http://znanium.com/bookread2.php?book=429576
3. Туркин О.В. VBA. Практическое программирование [Электронный ресурс]/ Туркин О.В.— Электрон. текстовые данные.— М.: СОЛОН-ПРЕСС, 2010.— 128 c. Режим доступа: http://www.iprbookshop.ru/8701

Программное обеспечение и Интернет-ресурсы

1. Портал: Компьютерные технологии, http://ru.wikipedia.org/wiki, 2016.
2. Официальный сайт поддержки компании Microsoft: https://msdn.microsoft.com, 2016.
3. http://professorweb.ru/, электронные материалы по технологии .NET, 2016.

Периодические издания

1. Журнал «Информатика и образование»: http://infojournal.ru/
2. Журнал «Информационные технологии»: http://novtex.ru/IT/
3. Журнал «Информационное общество»: http://www.infosoc.iis.ru/index.html

8. МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ (МОДУЛЯ)

- Компьютерный класс на основе ЭВМ ПК IntelCore с доступом в сеть Интернет, маркерная и интерактивная доски, переносной ноутбук, наушники, колонки.
- Мультимедийный комплекс в составе: Ноутбук с выходом в сеть Интернет, мультимедиа проектор, экран белый матовый, доска маркерная.

Рабочая программа дисциплины составлена в соответствии с требованиями ФГОС ВО по направлению 44.03.05 «Педагогическое образование», профиль «Информатика. Математика»

Рабочую программу составил асс. ***Якубович Д.А.***

(ФИО, подпись)

Рецензент (представитель работодателя) учитель высшей категории МБОУ СОШ №15 г.Владимир ***Козлова С.А.***

(место работы, должность, ФИО, подпись)

Программа рассмотрена и одобрена на заседании кафедры

Протокол № 7а от 10.03.16 года

Заведующий кафедрой ИИТО, проф. ***Медведев Ю.А.***

(ФИО, подпись)

Рабочая программа рассмотрена и одобрена на заседании учебно-методической комиссии направления 44.03.05 Педагогическое образование

Протокол № 3 от 17.03.16 года

Председатель комиссии ***Артамонова М.В.***

(ФИО, подпись)

ЛИСТ ПЕРЕУТВЕРЖДЕНИЯ
РАБОЧЕЙ ПРОГРАММЫ ДИСЦИПЛИНЫ (МОДУЛЯ)

Рабочая программа одобрена на _____ учебный год
Протокол заседания кафедры № _____ от _____ года
Заведующий кафедрой_____

Рабочая программа одобрена на _____ учебный год
Протокол заседания кафедры № _____ от _____ года
Заведующий кафедрой_____

Рабочая программа одобрена на _____ учебный год
Протокол заседания кафедры № _____ от _____ года
Заведующий кафедрой_____