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)
10	3 / 108	-	-	26	82	ЗАЧЕТ С ОЦЕНКОЙ
Итого	3 / 108	-	-	26	82	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:

- Познакомить учащихся с рядом новых технологий в программировании, в частности лямбда-выражениями, LINQ, основами распараллеливания под управлением платформы .NET Framework 4.5.
- Сформировать и закрепить опыт применения новых технологий на основе практических задач.
- Обучить студентов основам оптимизации и отладки кода с применением IDE Visual Studio 2012.

2. МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ОПОП ВО

Дисциплина «Практикум по решению задач на ЭВМ» относится к вариативной части учебного плана по направлению «Педагогическое образование» (Б1.В.ОД.14).

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

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:

- ряд новых технологий в программировании, в частности лямбда-выражения, LINQ, PLINQ под управлением платформы .NET Framework 4.5 (ОК-6 / ПК-1).

Уметь:

- применять новые технологии на основе практических задач (ПК-1 / ПК-12).
- осуществлять согласованную работу в коллективе из нескольких человек в целях достижения поставленной учебной задачи (ПК-12).

Владеть:

- основами оптимизации и отладки кода с применением IDE Visual Studio 2012 (ПК-1).

4. СТРУКТУРА И СОДЕРЖАНИЕ ДИСЦИПЛИНЫ (МОДУЛЯ)

Общая трудоёмкость дисциплины составляет 3 зачетные единицы, 108 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	Интерпретатор командной строки. Компиляция программ на языке C#. Пакетные файлы	10	1-2			4		8		2/50%	
2	Лямбда-выражения	10	3			2		8		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		10		1/25%	

4	LINQ. Операторы group, into, join, let. Группирование результатов с помощью оператора group.	10	6-7			4		14		1/25%	Рейтинг-контроль №2
5	LINQ. Анонимные типы. Методы в LINQ.	10	8-9			4		10		2/50%	
6	Многопоточное программирование. Класс Thread Многопоточное программирование. Классы Task и Parallel	10	10-13			8		32		4/100%	Рейтинг-контроль №3
Всего						**26**		**82**		**11/42,3%**	**ЗАЧЕТ С ОЦЕНКОЙ**

Темы и содержание лабораторных занятий

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 запросов.
- 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

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.
9. 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 с оценкой

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

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

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.
5. 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

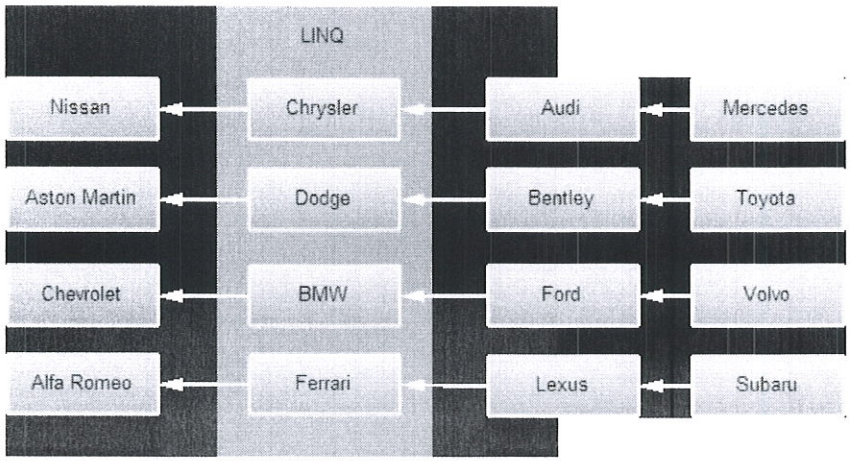
В своей основе Parallel LINQ, он же PLINQ — это версия LINQ to Objects, в которой объекты исходного перечисления обрабатываются параллельно.

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.

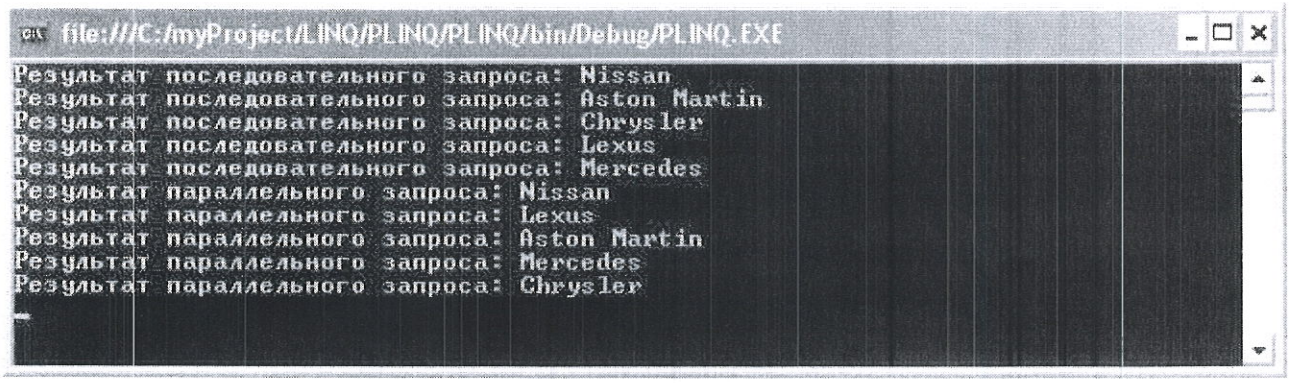
Запросы Parallel LINQ

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.

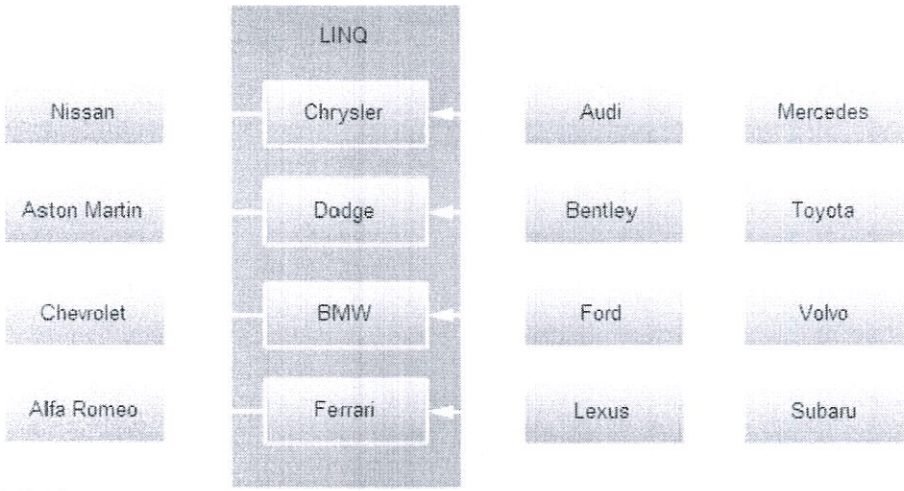
Второй запрос делает то же самое, но вдобавок вызывается метод AsParallel. С его помощью источник данных преобразуется в ParallelQuery, что автоматически подразумевает применение Parallel LINQ. И как явно следует из кода, никаких других изменений не требуется. Просто вызвав AsParallel, мы получаем PLINQ:



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.

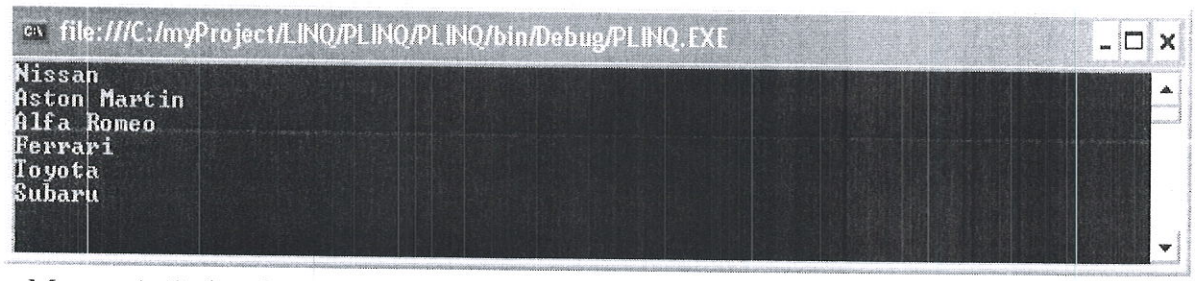
Найденное соответствие добавляется в результирующий набор. На рисунке наглядно видно, что если элементы обрабатывались примерно в одно и то же время каждым из ядер, то первым результатом будет Nissan, за ним последуют Aston Martin, Lexus, и т.д.

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

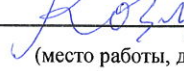
8. МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ (МОДУЛЯ)

- 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.*** 

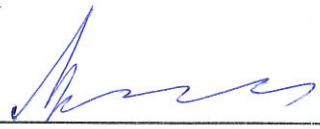
(ФИО, подпись)

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

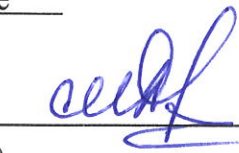
Протокол № 7а от 10.03.16 года

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_____