**Лабораторная работа №6**

**Списки. Поиск по спискам. LINQ.**

*Списки*

Список – это абстрактный тип данных, представляющий собой упорядоченный набор значений, в котором некоторое значение может встречаться более одного раза. В C#, списки входят в пространство имен System.Collections, в том числе дженерики. При этом, для дженерики находятся в подпространстве System.Collections.Generics.

Дженерик – обобщающая конструкция, позволяющая абстрагировать множество типов. Использование дженериков позволяет «зарезервировать» коллекцию определенного типа без приведения типа, что снижает вероятность возникновения ошибки “Runtime Error”.

*List<T>*

List<T> представляет строго типизированный список объектов, доступных по индексу. Поддерживает методы для поиска по списку, выполнения сортировки и других операций со списками.

|  |
| --- |
| using System;  using System.Collections.Generic;  // Simple business object. A PartId is used to identify the type of part  // but the part name can change.  public class Part : IEquatable<Part>  {  public string PartName { get; set; }    public int PartId { get; set; }  public override string ToString()  {  return "ID: " + PartId + " Name: " + PartName;  }  public override bool Equals(object obj)  {  if (obj == null) return false;  Part objAsPart = obj as Part;  if (objAsPart == null) return false;  else return Equals(objAsPart);  }  public override int GetHashCode()  {  return PartId;  }  public bool Equals(Part other)  {  if (other == null) return false;  return (this.PartId.Equals(other.PartId));  }  // Should also override == and != operators.  }  public class Example  {  public static void Main()  {  // Create a list of parts.  List<Part> parts = new List<Part>();  // Add parts to the list.  parts.Add(new Part() {PartName="crank arm", PartId=1234});  parts.Add(new Part() { PartName = "chain ring", PartId = 1334 });  parts.Add(new Part() { PartName = "regular seat", PartId = 1434 });  parts.Add(new Part() { PartName = "banana seat", PartId = 1444 });  parts.Add(new Part() { PartName = "cassette", PartId = 1534 });  parts.Add(new Part() { PartName = "shift lever", PartId = 1634 });  // Write out the parts in the list. This will call the overridden ToString method  // in the Part class.  Console.WriteLine();  foreach (Part aPart in parts)  {  Console.WriteLine(aPart);  }    // Check the list for part #1734. This calls the IEquatable.Equals method  // of the Part class, which checks the PartId for equality.  Console.WriteLine("\nContains(\"1734\"): {0}",  parts.Contains(new Part {PartId=1734, PartName="" }));  // Insert a new item at position 2.  Console.WriteLine("\nInsert(2, \"1834\")");  parts.Insert(2, new Part() { PartName = "brake lever", PartId = 1834 });    //Console.WriteLine();  foreach (Part aPart in parts)  {  Console.WriteLine(aPart);  }  Console.WriteLine("\nParts[3]: {0}", parts[3]);  Console.WriteLine("\nRemove(\"1534\")");  // This will remove part 1534 even though the PartName is different,  // because the Equals method only checks PartId for equality.  parts.Remove(new Part(){PartId=1534, PartName="cogs"});  Console.WriteLine();  foreach (Part aPart in parts)  {  Console.WriteLine(aPart);  }  Console.WriteLine("\nRemoveAt(3)");  // This will remove the part at index 3.  parts.RemoveAt(3);  Console.WriteLine();  foreach (Part aPart in parts)  {  Console.WriteLine(aPart);  }  /\*    ID: 1234 Name: crank arm  ID: 1334 Name: chain ring  ID: 1434 Name: regular seat  ID: 1444 Name: banana seat  ID: 1534 Name: cassette  ID: 1634 Name: shift lever  Contains("1734"): False  Insert(2, "1834")  ID: 1234 Name: crank arm  ID: 1334 Name: chain ring  ID: 1834 Name: brake lever  ID: 1434 Name: regular seat  ID: 1444 Name: banana seat  ID: 1534 Name: cassette  ID: 1634 Name: shift lever  Parts[3]: ID: 1434 Name: regular seat  Remove("1534")  ID: 1234 Name: crank arm  ID: 1334 Name: chain ring  ID: 1834 Name: brake lever  ID: 1434 Name: regular seat  ID: 1444 Name: banana seat  ID: 1634 Name: shift lever  RemoveAt(3)  ID: 1234 Name: crank arm  ID: 1334 Name: chain ring  ID: 1834 Name: brake lever  ID: 1444 Name: banana seat  ID: 1634 Name: shift lever  \*/  }  } |

*ArrayList*

ArrayList – реализует интерфейс IList с помощью массива с динамическим изменением размера по требованию. ArrayList спроектирован для хранения гетерогенный коллекций объектов, однако предпочтительней использовать List<Object> для гетерогенных данных и List<T> – для однородных, поскольку ArrayList не гарантирует высокую производительность.

|  |
| --- |
| using System;  using System.Collections;  public class SamplesArrayList {  public static void Main() {  // Creates and initializes a new ArrayList.  ArrayList myAL = new ArrayList();  myAL.Add("Hello");  myAL.Add("World");  myAL.Add("!");  // Displays the properties and values of the ArrayList.  Console.WriteLine( "myAL" );  Console.WriteLine( " Count: {0}", myAL.Count );  Console.WriteLine( " Capacity: {0}", myAL.Capacity );  Console.Write( " Values:" );  PrintValues( myAL );  }  public static void PrintValues( IEnumerable myList ) {  foreach ( Object obj in myList )  Console.Write( " {0}", obj );  Console.WriteLine();  }  } |

*LINQ*

Аббревиатура LINQ обозначает целый набор технологий, создающих и использующих возможности интеграции запросов непосредственно в язык C#. Традиционно запросы к данным выражаются в виде простых строк без проверки типов при компиляции или поддержки IntelliSense. Кроме того, разработчику приходится изучать различные языки запросов для каждого типа источников данных: баз данных SQL, XML-документов, различных веб-служб и т. д. Технологии LINQ превращают запросы в удобную языковую конструкцию, которая применяется аналогично классам, методам и событиям. Запросы создаются к строго типизированным коллекциям объектов с помощью ключевых слов языка и знакомых операторов. Семейство технологий LINQ обеспечивает согласованное функционирование запросов для объектов (LINQ to Objects), реляционных баз данных (LINQ to SQL) и XML (LINQ to XML).

Выражения запроса используют декларативный синтаксис запроса. С помощью синтаксиса запроса можно выполнять фильтрацию, упорядочение и группирование данных из источника данных, обходясь минимальным объемом программного кода. Одни и те же базовые выражения запроса позволяют одинаково легко получать и преобразовывать данные из баз данных SQL, наборов данных ADO .NET, XML-документов, XML-потоков и коллекций .NET.

LINQ в C# позволяет формировать запросы для обращения к базам данных SQL Server, XML-документам, наборам данных ADO.NET и любым коллекциям объектов, поддерживающим интерфейс IEnumerable или универсальный интерфейс IEnumerable<T>. Кроме того, сторонние разработчики обеспечивают поддержку LINQ для множества веб-служб и других реализаций баз данных.

В следующем примере показан полный пример использования запроса. Полная операция сначала создает источник данных, затем определяет выражение запроса и выполняет этот запрос в инструкции foreach.

|  |
| --- |
| class LINQQueryExpressions  {  static void Main()  {    // Specify the data source.  int[] scores = new int[] { 97, 92, 81, 60 };  // Define the query expression.  IEnumerable<int> scoreQuery =  from score in scores  where score > 80  select score;  // Execute the query.  foreach (int i in scoreQuery)  {  Console.Write(i + " ");  }  }  }  // Output: 97 92 81 |

**Варианты заданий**

## Задание

1. Создать консольное приложение на смоделированной структуре из ЛР 5, осуществляющее поиск по экземплярам (с использованием коллекций List<T> или ArrayList, без LINQ).

Приложение должно содержать:

* Экземпляры классов, по которым будет осуществляться поиск (см.пример).
* Интерфейс командной строки: вопрос о намерении пользователя осуществить поиск (y/n), уточнение класса экземпляра (точное название), поиск по любому из атрибутов, содержащихся в классах (число, текст, дата).
* После выбора экземпляра пользователь вводит любые данные для соответствующей структуры (например, название жанра литературы или ФИО автора), поиск должен осуществляться по всем свойствам.
* Логичный вывод релевантной информации, связанной с искомым объектом. Студент должен самостоятельно определить, что будет выводить приложение.

Например, структура «Аэропорт» содержит классы: аэропорт, летательный аппарат, самолет, вертолет, пассажир. Минимальное количество экземпляров класса «аэропорт»: 2; минимальное количество экземпляров класса самолет: 2 (в каждом аэропорту); минимальное количество экземпляров класса «вертолет»: 1 (в каждом аэропорту); минимальное количество

1. Реализовать поиск аналогично п.1, с использованием LINQ.

Студент должен самостоятельно предусмотреть возможные ошибки, которые могут возникнуть при взаимодействии пользователя с приложением, и реализовать соответствующую обработку ошибок.

Вариант № 1

**Смоделировать структуру предприятия:**

|  |  |
| --- | --- |
| **Классы** | **Свойства и методы** |
| Фирма | название (get, set)  список отделов (get, set) |
| Отдел | название (get, set)  количество сотрудников (get, set)  список сотрудников (get, set) |
| Сотрудник | фио (get, set)  должность (get, set)  оклад (get, set)  рассчитать зарплату() |
| Штатный сотрудник | премия (get, set)  рассчитать зарплату() |
| Сотрудник по контракту | рассчитать зарплату() |

Вариант № 2

**Смоделировать структуру банка:**

|  |  |
| --- | --- |
| **Классы** | **Свойства** |
| Банк | название (get, set)  список вкладов (get, set) |
| Филиал | название (get, set)  общая сумма вкладов (get, set)  список вкладов (get, set) |
| Вклад | фио вкладчика (get, set)  сумма вклада (get, set)  рассчитать сумму вклада (количество месяцев) |
| Долгосрочный вклад | рассчитать сумму вклада (количество месяцев) |
| Вклад до востребования | рассчитать сумму вклада (количество месяцев) |

Вариант № 3

**Смоделировать структуру аэропорта:**

|  |  |
| --- | --- |
| **Классы** | **Свойства** |
| Аэропорт | название (get, set)  список летательных аппаратов (get, set) |
| Летательный аппарат | название (get, set)  макс. количество пассажиров (get, set)  взлет()  список пассажиров (get, set) |
| Самолет | длина полосы разгона (get, set)  взлет()  список пассажиров (get, set) |
| Вертолет | взлет()  список пассажиров (get, set) |
| Пассажир | фио (get, set)  №посадочного места (get, set) |

Вариант № 4

**Смоделировать структуру библиотеки:**

|  |  |
| --- | --- |
| **Классы** | **Свойства** |
| Библиотека | название (get, set)  список отделов (get, set) |
| Отдел (по жанрам) | название жанра (get, set)  количество изданий (get, set)  список изданий (get, set) |
| Издание | название (get, set)  автор (get, set)  год издания (get, set)  сформировать описание() |
| Книга | резюме (get, set)  сформировать описание() |
| Журнал | статьи (get, add)  сформировать описание() |

Вариант № 5

**Смоделировать структуру компании сотовой связи:**

|  |  |
| --- | --- |
| **Классы** | **Свойства** |
| Компания | Название (get, set)  список тарифов (get, set) |
| Тариф | название (get, set)  количество абонентов (get, set)  рассчитать стоимость разговора (количество секунд)  список абонентов (get, set) |
| Посекундный тариф | стоимость секунды разговора» (get, set)  рассчитать стоимость разговора (количество секунд)  список тарифов (get, set) |
| Поминутный тариф | стоимость минуты разговора» (get, set)  рассчитать стоимость разговора (количество секунд)  список тарифов (get, set) |
| Абонент | фио (get, set)  номер телефона (get, set)  остаток на счете (get, set) |

Вариант № 6

**Смоделировать структуру автосалона:**

|  |  |
| --- | --- |
| **Классы** | **Свойства** |
| Автосалон | название (get, set)  список автомобилей (get, set) |
| Автомобиль | марка (get, set)  макс. количество пассажиров (get, set)  стоимость (get, set)  количество на складе(get, set)  boolean наличие (get, set)  список заявок (get, set) |
| Заявка на покупку | фио покупателя (get, set)  номер телефона (get, set)  автомобили (add, remove, get)  рассчитать стоимость заказа() |
| Заявка на приобретение со стенда | рассчитать стоимость заказа() |
| Заявка на отложенную поставку | процент скидки (get, set)  рассчитать стоимость заказа() |

Вариант № 7

**Смоделировать структуру музыкальной коллекции:**

|  |  |
| --- | --- |
| **Классы** | **Свойства** |
| Коллекция | название (get, set)  фио владельца (get, set)  список альбомов (get, set) |
| Музыкальный носитель (альбом) | автор/группа (get, set)  жанр (get, set)  год выпуска (get, set)  общая продолжительность звучания (get, set)  список музыкальных произведений (get, set) |
| Музыкальное произведение | название (get, set)  продолжительность (get, set)  сформировать описание() |
| Песня | текст (get, set)  автор текста (get, set)  сформировать описание() |
| Инструментальное произведение | инструменты (get, set)  сформировать описание() |

Вариант № 8

**Смоделировать структуру реестра городского жилья:**

|  |  |
| --- | --- |
| **Классы** | **Свойства** |
| Город | название (get, set)  список зданий (get, set) |
| Здание | название улицы (get, set)  номер дома (get, set)  общая площадь (get)  базовая ежемесячная оплата за кв.м площади (get, set)  список помещений (get, set) |
| Помещение | номер (get, set)  площадь (get, set)  рассчитать ежемес. оплату() |
| Квартира | ФИО жильцов (get, add, remove)  рассчитать ежемес. оплату() |
| Офис | название фирмы-владельца (get, set)  вид деятельности (get, set)  рассчитать ежемес. оплату() |

Вариант № 9

**Смоделировать структуру зоопарка:**

|  |  |
| --- | --- |
| **Классы** | **Свойства** |
| Зоопарк | название (get, set)  список клеток (get, set) |
| Вольер/клетка | номер (get, set)  размер (get, set)  макс. количество животных (get, set)  текущее количество животных (get, set)  список животных (get, set) |
| Животное | название (get, set)  boolean хищник (get, set)  сформировать описание() |
| Рыба | глубоководная (get, set)  сформировать описание() |
| Птица | скорость полета (get, set)  сформировать описание() |
| Зверь | среда обитания (get, set)  сформировать описание() |

Вариант № 10

**Смоделировать структуру автоматизированного банкомата:**

|  |  |
| --- | --- |
| **Классы** | **Свойства** |
| Банк | название (get, set)  список счетов (get, set)  список банкоматов (get, set) |
| Счет | номер (get, set)  PIN-код (get, set)  остаток (get, set)  снять со счета (сумма) |
| Обычный счет | снять со счета (сумма) |
| Льготный счет | снять со счета (сумма) |
| Банкомат | идентификационный номер (get, set)  адрес (get, set) |

Приложение 1

Справочные материалы

Structure and Interpretation of Computer Programs, 2nd ed.

<https://web.mit.edu/alexmv/6.037/sicp.pdf>

List<T>

<https://docs.microsoft.com/ru-ru/dotnet/api/system.collections.generic.list-1?view=netframework-4.8>

ArrayList

<https://docs.microsoft.com/ru-ru/dotnet/api/system.collections.arraylist?view=netframework-4.8>

LINQ

<https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/concepts/linq/>