**Рубежный контроль №2**

**5 вариант**

**Как создать LINQ-запрос с сортировкой по убыванию?**

Операции упорядочивания позволяют выстраивать входные последовательности в определенном порядке. Важно отметить, что и OrderBy, и OrderByDescending требуют входной последовательности типа IEnumerable<T> и возвращают последовательность типа IOrderedEnumerable<T>. Передавать операциям OrderBy и OrderByDescending в качестве входной последовательности IOrderedEnumerable<T> нельзя. Причина в том, что последующие вызовы операций OrderBy и OrderByDescending не принимают во внимание порядок, созданный предыдущими вызовами OrderBy и OrderByDescending. Это значит, что передавать последовательность, возвращенную из OrderBy либо OrderByDescending, в последующий вызов операции OrderBy или OrderByDescending не имеет смысла.

Операция OrderByDescending позволяет упорядочить входную последовательность на основе метода keySelector, который возвращает значение ключа для каждого входного элемента. Упорядоченная выходная последовательность IOrderedEnumerable<T> выдается в порядке убывания на основе значений возвращенных ключей.

Сортировка, выполненная операцией OrderByDescending, определена как неустойчивая. Это значит, что она не сохраняет входной порядок элементов. Если два входных элемента поступают в операцию OrderByDescending в определенном порядке, и значения ключей этих двух элементов совпадают, их расположение в выходной последовательности может остаться прежним или поменяться, причем ни то, ни другое не гарантируется. Даже если все выглядит нормально, поскольку порядок определен как неустойчивый, всегда следует исходить из этого. Это значит, что никогда нельзя полагаться на порядок элементов, поступающих из операций OrderBy или OrderByDescending, для любого поля кроме указанного в вызове метода. Сохранение любого порядка, который существует в последовательности, передаваемой любой из этих операций, не может гарантироваться.

string[] cars = { "Alfa Romeo", "Aston Martin", "Audi", "Nissan", "Chevrolet",

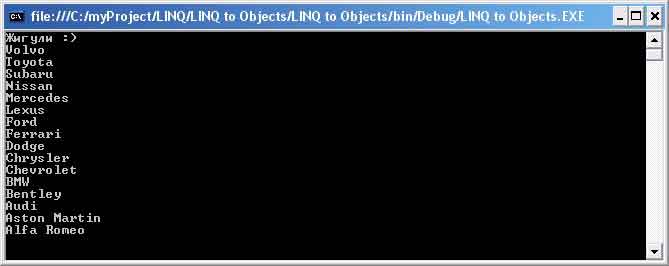
"Chrysler", "Dodge", "BMW", "Ferrari", "Bentley", "Ford", "Lexus",

"Mercedes", "Toyota", "Volvo", "Subaru", "Жигули :)" };

IEnumerable<string> auto = cars.OrderByDescending(s => s);

foreach (string str in auto)

Console.WriteLine(str);



Если требуется большая степень упорядочивания, чем возможно достичь с помощью одиночного вызова операции OrderBy или OrderByDescending, необходимо последовательно вызывать операции ThenBy или ThenByDescending. Вызовы ThenBy и ThenByDescending могут соединяться в цепочку, т.к. они принимают в качестве входной последовательности IOrderedEnumerable<T> и возвращают в качестве выходной последовательности тоже IOrderedEnumerable<T>. Это значит, что сначала должна быть вызвана операция OrderBy или OrderByDescending для создания последовательности IOrderedEnumerable, на которой можно затем вызывать операции ThenBy и ThenByDescending.

Операция ThenByDescending позволяет упорядочивать по убыванию входную последовательность типа IOrderedEnumerable<T> на основе метода keySelector, который возвращает значение ключа. В результате выдается упорядоченная последовательность типа IOrderedEnumerable<T>.

Сортировка, выполняемая операцией ThenByDescending, является устойчивой. Другими словами, она сохраняет входной порядок элементов с эквивалентными ключами. Если два входных элемента поступили в операцию ThenByDescending в определенном порядке, и ключевое значение обоих элементов одинаково, то порядок тех же выходных элементов гарантированно сохранится. В отличие от OrderBy и OrderByDescending, операции ThenBy и ThenByDescending выполняют устойчивую сортировку.

string[] cars = { "Alfa Romeo", "Aston Martin", "Audi", "Nissan", "Chevrolet",

"Chrysler", "Dodge", "BMW", "Ferrari", "Bentley", "Ford", "Lexus",

"Mercedes", "Toyota", "Volvo", "Subaru", "Жигули :)" };

IEnumerable<string> auto = cars.OrderBy(s => s.Length).ThenByDescending(s => s);

foreach (string str in auto)

Console.WriteLine(str);

