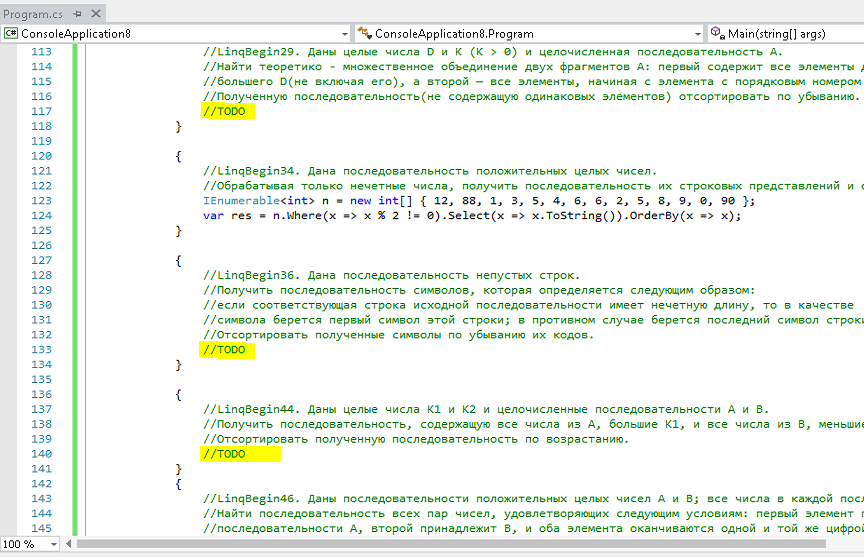
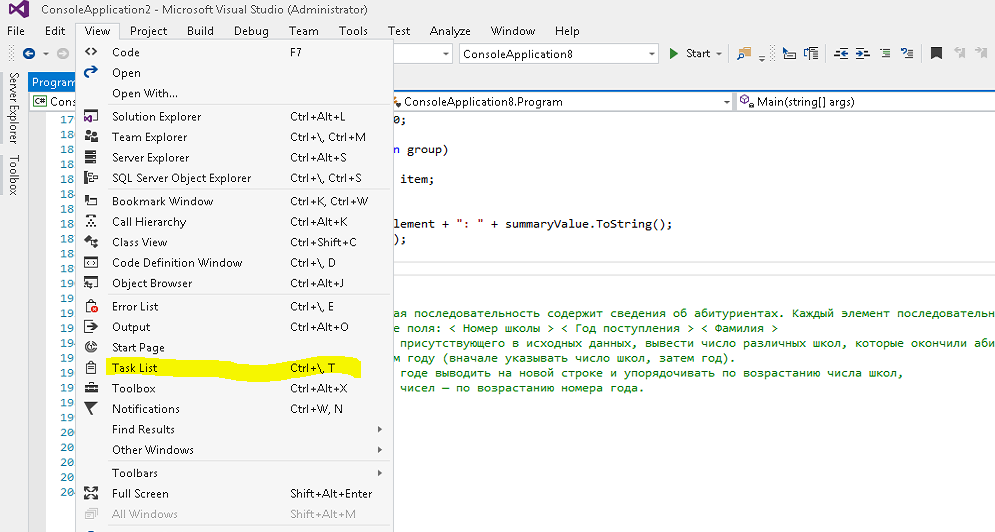
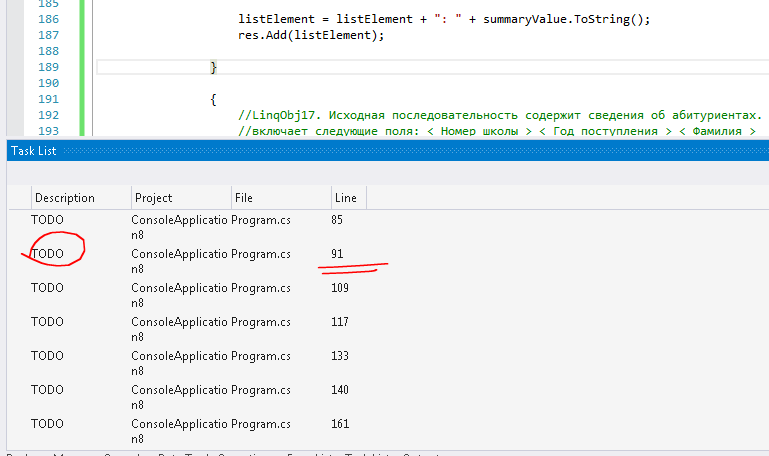
|  |
| --- |
|  |
| Практические задания  C# Базовый курс. #8 |

|  |  |  |  |
| --- | --- | --- | --- |
| REVISION | | | |
| Версия | Описание | Автор | Дата |
|
| 1.0 | Начальная версия | Анжелика Кравчук |  |
| 2.0 |  | Крамар Ю.М. |  |

Очень удобным является вставка специального комментария //TODO, чтобы отмечать в коде места, требующие их доработки:



Тогда в специальном окне Task List можно получить список таких мест кода без применения операций поиска в тексте:



## Задание

Доработать исходный код (скопировать в консольный проект и решить задачи с комментариями TODO):

using System;

using System.Collections.Generic;

using System.Linq;

namespace ConsoleApplication8

{

class Film

{

public string Name { get; set; }

public int Year { get; set; }

}

class Program

{

static void Main(string[] args)

{

var films = new List<Film>

{

new Film { Name = "Jaws", Year = 1975 },

new Film { Name = "Singing in the Rain", Year = 1952 },

new Film { Name = "Some like it Hot", Year = 1959 },

new Film { Name = "The Wizard of Oz", Year = 1939 },

new Film { Name = "It’s a Wonderful Life", Year = 1946 },

new Film { Name = "American Beauty", Year = 1999 },

new Film { Name = "High Fidelity", Year = 2000 },

new Film { Name = "The Usual Suspects", Year = 1995 }

};

//Создание многократно используемого делегата для вывода списка на консоль

Action<Film> print = film => Console.WriteLine($"Name={film.Name}, Year={film.Year}");

//Вывод на консоль исходного списка

films.ForEach(print);

//Создание и вывод отфильтрованного списка

films.FindAll(film => film.Year < 1960).ForEach(print);

//Сортировка исходного списка

films.Sort((f1, f2) => f1.Name.CompareTo(f2.Name));

//or

films.OrderBy(film => film.Name);

{

// OrderByDescending, Skip, SkipWhile, Take, TakeWhile, Select, Concat

int[] n = { 1, 3, 5, 6, 3, 6, 7, 8, 45, 3, 7, 6 };

IEnumerable<int> res;

res = n.OrderByDescending(g => g).Skip(3);

res = n.OrderByDescending(g => g).Take(3);

res = n.Select(g => g \* 2);

// to delete from array number 45

res = n.TakeWhile(g => g != 45).Concat(n.SkipWhile(s => s != 45).Skip(1));

}

{

//Дана последовательность непустых строк.

//Объединить все строки в одну.

List<string> str = new List<string> { "1qwerty", "cqwertyc", "cqwe", "c" };

string amount = str.Aggregate<string>((x, y) => x + y);

}

{

//LinqBegin3. Дано целое число L (> 0) и строковая последовательность A.

//Вывести последнюю строку из A, начинающуюся с цифры и имеющую длину L.

//Если требуемых строк в последовательности A нет, то вывести строку «Not found».

//Указание.Для обработки ситуации, связанной с отсутствием требуемых строк, использовать операцию ??.

int length = 8;

List<string> str = new List<string> { "1qwerty", "2qwerty", "7qwe" };

string res = str.FirstOrDefault(x => (Char.IsDigit(x[0])) && (x.Length == length)) ?? "Not found";

}

{

//LinqBegin5. Дан символ С и строковая последовательность A.

//Найти количество элементов A, которые содержат более одного символа и при этом начинаются и оканчиваются символом C.

char c = 'c';

List<string> str = new List<string> { "1qwerty", "cqwertyc", "cqwe", "c" };

int amount = str.Count(x => (x.StartsWith(c.ToString())) && (x.EndsWith(c.ToString())) && (x.Length > 1));

}

{

//LinqBegin6. Дана строковая последовательность.

//Найти сумму длин всех строк, входящих в данную последовательность.

//TODO

}

{

//LinqBegin11. Дана последовательность непустых строк.

//Получить строку, состоящую из начальных символов всех строк исходной последовательности.

//TODO

}

{

//LinqBegin30. Дано целое число K (> 0) и целочисленная последовательность A.

//Найти теоретико-множественную разность двух фрагментов A: первый содержит все четные числа,

//а второй — все числа с порядковыми номерами, большими K.

//В полученной последовательности(не содержащей одинаковых элементов) поменять порядок элементов на обратный.

int k = 5;

IEnumerable<int> n = new int[] { 12, 88, 1, 3, 5, 4, 6, 6, 2, 5, 8, 9, 0, 90 };

var res = n.Where(x => x % 2 == 0).Except(n.Skip(k)).Reverse();

}

{

//LinqBegin22. Дано целое число K (> 0) и строковая последовательность A.

//Строки последовательности содержат только цифры и заглавные буквы латинского алфавита.

//Извлечь из A все строки длины K, оканчивающиеся цифрой, отсортировав их по возрастанию.

//TODO

}

{

//LinqBegin29. Даны целые числа D и K (K > 0) и целочисленная последовательность A.

//Найти теоретико - множественное объединение двух фрагментов A: первый содержит все элементы до первого элемента,

//большего D(не включая его), а второй — все элементы, начиная с элемента с порядковым номером K.

//Полученную последовательность(не содержащую одинаковых элементов) отсортировать по убыванию.

//TODO

}

{

//LinqBegin34. Дана последовательность положительных целых чисел.

//Обрабатывая только нечетные числа, получить последовательность их строковых представлений и отсортировать ее по возрастанию.

IEnumerable<int> n = new int[] { 12, 88, 1, 3, 5, 4, 6, 6, 2, 5, 8, 9, 0, 90 };

var res = n.Where(x => x % 2 != 0).Select(x => x.ToString()).OrderBy(x => x);

}

{

//LinqBegin36. Дана последовательность непустых строк.

//Получить последовательность символов, которая определяется следующим образом:

//если соответствующая строка исходной последовательности имеет нечетную длину, то в качестве

//символа берется первый символ этой строки; в противном случае берется последний символ строки.

//Отсортировать полученные символы по убыванию их кодов.

//TODO

}

{

//LinqBegin44. Даны целые числа K1 и K2 и целочисленные последовательности A и B.

//Получить последовательность, содержащую все числа из A, большие K1, и все числа из B, меньшие K2.

//Отсортировать полученную последовательность по возрастанию.

//TODO

}

{

//LinqBegin46. Даны последовательности положительных целых чисел A и B; все числа в каждой последовательности различны.

//Найти последовательность всех пар чисел, удовлетворяющих следующим условиям: первый элемент пары принадлежит

//последовательности A, второй принадлежит B, и оба элемента оканчиваются одной и той же цифрой.

//Результирующая последовательность называется внутренним объединением последовательностей A и B по ключу,

//определяемому последними цифрами исходных чисел.

//Представить найденное объединение в виде последовательности строк, содержащих первый и второй элементы пары,

//разделенные дефисом, например, «49 - 129».

IEnumerable<int> n1 = new int[] { 12, 88, 11, 3, 55, 679, 222, 845, 9245 };

IEnumerable<int> n2 = new int[] { 123, 888, 551, 443, 69, 222, 780 };

var res = n1.Join(n2, x => x % 10, y => y % 10, (x, y) => x.ToString() + " - " + y.ToString());

}

{

//LinqBegin48.Даны строковые последовательности A и B; все строки в каждой последовательности различны,

//имеют ненулевую длину и содержат только цифры и заглавные буквы латинского алфавита.

//Найти внутреннее объединение A и B, каждая пара которого должна содержать строки одинаковой длины.

//Представить найденное объединение в виде последовательности строк, содержащих первый и второй элементы пары,

//разделенные двоеточием, например, «AB: CD». Порядок следования пар должен определяться порядком

//первых элементов пар(по возрастанию), а для равных первых элементов — порядком вторых элементов пар(по убыванию).

//TODO

}

{

//LinqBegin56. Дана целочисленная последовательность A.

//Сгруппировать элементы последовательности A, оканчивающиеся одной и той же цифрой, и на основе этой группировки

//получить последовательность строк вида «D: S», где D — ключ группировки (т.е.некоторая цифра, которой оканчивается

//хотя бы одно из чисел последовательности A), а S — сумма всех чисел из A, которые оканчиваются цифрой D.

//Полученную последовательность упорядочить по возрастанию ключей.

//Указание.Использовать метод GroupBy.

IEnumerable<int> n = new int[] { 12, 88, 11, 3, 55, 679, 222, 845, 9245 };

List<string> res = new List<string>();

IEnumerable<IGrouping<int, int>> groups = n.GroupBy(x => x % 10).OrderBy( x=> x.Key);

foreach (IGrouping<int, int> group in groups)

{

string listElement = group.Key.ToString();

int summaryValue = 0;

foreach (int item in group)

{

summaryValue += item;

}

listElement = listElement + ": " + summaryValue.ToString();

res.Add(listElement);

}

{

//LinqObj17. Исходная последовательность содержит сведения об абитуриентах. Каждый элемент последовательности

//включает следующие поля: < Номер школы > < Год поступления > < Фамилия >

//Для каждого года, присутствующего в исходных данных, вывести число различных школ, которые окончили абитуриенты,

//поступившие в этом году (вначале указывать число школ, затем год).

//Сведения о каждом годе выводить на новой строке и упорядочивать по возрастанию числа школ,

//а для совпадающих чисел — по возрастанию номера года.

//TODO

}

}

}

}

}