Лабораторная работа №5

Файлы и классы

Вариант №15

*Постановка задачи*: База данных риэлторской компании хранится в файле. Сведения о каждом

объекте недвижимости включают: тип (квартира, комната, земельный участок, дом), количество комнат, этаж, площадь, цена, адрес, описание. Отдельной сущностью (классом) хранится информация о клиентах: № клиента, № телефона, ФИО, адрес, электронная почта.

*Входные данные*:

* строка;

*Выходные данные*:

* massivBezTochki – преобразованный массив;

*Рабочие переменные*:

* i – счетчик цикла;
* element;
* massivBezTochki.
* OneElement.
* Arr.

*Описание стандартных функций и библиотек*

* Console.WriteLine() – вывод данные на экран;
* ToCharArray() – копирует знаки в массив чар;
* Ref – передает ссылку на ячейку памяти в куче
* Out – передает ссылку на ячейку памяти в куче, объявление переменной в методе.
* Length – длинна массива
* Console.ReadLine() – считывание данные с клавиатуры;

*Классы и процедуры, созданные автором*

* Нет.

*Тестовый пример*

Если пользователь введёт 5 произвольных слов то программа отобразит <<Массив введенных слов, далее программа удалит слова идентичные первому введенному слову и те слова в которых первая буква встречается еще раз. >>.

*Текст программы:*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace DB\_RieltorCompany

{

class Program

{

static void Main(string[] args)

{

Console.WindowWidth = 120;

Flats myFlats = new Flats("data.csv");

Clients myClients = new Clients("clients.csv");

string choise;

do

{

showFlats(myFlats);

Console.WriteLine("Что ты хочешь делать?");

Console.WriteLine("1-удалить строку");

Console.WriteLine("2-добавить строку");

Console.WriteLine("3-изменить строку");

Console.WriteLine("4-показать таблицу клиентов");

Console.WriteLine("5-показать кто купил дом");

Console.WriteLine("6-хочу выйти.");

choise = Console.ReadLine();

if (choise == "1") { deleteRow(ref myFlats); }

if (choise == "2") { appendRow(ref myFlats); }

if (choise == "3") { changeRow(ref myFlats); }

if (choise == "4") { showClients(myClients); }

if (choise == "5") { showLinkedRow(myFlats, myClients); }

Console.Clear();

} while (choise != "6");

Console.WriteLine("Сохранить базу данных?(y/n)");

string isSaveData = Console.ReadLine().ToLower();

if (isSaveData == "y")//сохранение файла

{

myFlats.saveFlatsToFile();

Console.WriteLine("База сохранена...");

}

}

#region Вывод на консоль

static void showFlats(Flats flatsTable)

{

string tmpStr = "";

foreach (string item in flatsTable.Title.Split(';'))

{

tmpStr += item + "\t";

}

tmpStr += "\n";

for (int i = 0; i < flatsTable.flatsCount; i++)

{

tmpStr += flatsTable.getFlat(i).ToString("\t") + "\n";

}

Console.WriteLine(tmpStr);

}

static void showClients(Clients clientsTable)

{

string tmpStr = "";

foreach (string item in clientsTable.Title.Split(';'))

{

tmpStr += item + "\t";

}

tmpStr += "\n";

for (int i = 0; i < clientsTable.clientsCount; i++)

{

tmpStr += clientsTable.getClient(i).ToString("\t") + "\n";

}

Console.WriteLine(tmpStr);

Console.ReadKey();

}

#endregion

static void deleteRow(ref Flats flatsTable)//удаление строки

{

try

{

Console.WriteLine("Введи ID строки для удаления");

int numberForDelete = flatsTable.getRowNumberByID(Convert.ToInt32(Console.ReadLine()));

flatsTable.removeRow(numberForDelete);

}

catch (FormatException e)

{

Console.WriteLine("Необходимо вводить числа!!!!\n" + e);

Console.WriteLine(e.ToString());

Console.ReadKey();

}

catch (ArgumentOutOfRangeException)

{

Console.WriteLine("Нет строки с таким номером!!!");

Console.ReadKey();

}

catch (Exception e)

{

Console.WriteLine("Ошибка, которую я не смог предпологать");

Console.WriteLine(e.Data);

Console.WriteLine(e.Message);

Console.WriteLine(e.ToString());

Console.ReadKey();

}

}

static void appendRow(ref Flats flatsTable)//добавить строку

{

string newDataStringForRow = flatsTable.nextId.ToString() + ";";//новый айди добавил в строку

string s = "";

for (int i = 1; i < 9; i++)

{

Console.WriteLine("Введи " + i + "-ый элемент новой строки(" + flatsTable.Title.Split(';')[i] + ")");

s = Console.ReadLine();

if (s.Trim() == "")

{

Console.WriteLine("Введи хоть что нибудь!");

i -= 1;

}

else

{

newDataStringForRow += s.Trim() + ";";

}

}

newDataStringForRow = newDataStringForRow.Substring(0, newDataStringForRow.Length - 1);//убрал последнюю запитую

flatsTable.addRow(new Flat(newDataStringForRow));//добавил новый объект этой строкой в конструкторе в лист

}

static void changeRow(ref Flats flatsTable)//изменить строку

{

Console.WriteLine("Введи ID строки для изменения");

int numberForChange = flatsTable.getRowNumberByID(Convert.ToInt32(Console.ReadLine()));

try

{

string newDataStringForRow = flatsTable.getFlat(numberForChange).id.ToString() + ";";//айди + точка с запятой

string s = "";

for (int i = 1; i < 9; i++)

{

Console.WriteLine("Введи " + i + "-ый элемент новой строки(" + flatsTable.Title.Split(';')[i] + ")");

s = Console.ReadLine();

if (s.Trim() == "")

{

Console.WriteLine("Введи хоть что нибудь!!!");

i -= 1;

}

else

{

newDataStringForRow += s.Trim() + ";";

}

}

newDataStringForRow = newDataStringForRow.Substring(0, newDataStringForRow.Length - 1);

flatsTable.changeRow(numberForChange, new Flat(newDataStringForRow));

}

catch (Exception)

{

Console.WriteLine("При изменении произошла ошибка...скорее всего такого номера строки не существует");

Console.ReadKey();

}

}

static void showLinkedRow(Flats flats, Clients clients)//показывает связанные по айдишнику строки

{

Console.WriteLine("Введи ID дома");

int number = flats.getRowNumberByID(Convert.ToInt32(Console.ReadLine()));

if (number > -1)

{

int foundedNumber = -1;

int clientID = flats.getFlat(number).clientID;//сравнивает номера айдишников

for (int i = 0; i < clients.clientsCount; i++)

{

if (clientID == clients.getClient(i).id)

{

foundedNumber = i;

break;

}

}

if (foundedNumber == -1)

{

Console.WriteLine("Я не знаю кто купил этот дом");

}

else

{

Console.WriteLine(clients.getClient(foundedNumber).ToString("\t"));

}

}

else

{

Console.WriteLine("Такого ID нет");

}

Console.ReadKey();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.IO;

namespace DB\_RieltorCompany

{

class Flats

{

private List<Flat> listFlat = new List<Flat>();

const string titleConst = "id;realtyType;roomsCount;floor;area;price;address;description;clientID";

public int flatsCount { get { return listFlat.Count; } }//длинна листа

#region Properties

public string Title//возращает значение константы

{ get { return titleConst; } }

/// Вычисляет номер строки по ID, если не находит, то выводит -1

public int getRowNumberByID(int id)

{

for (int i = 0; i < this.listFlat.Count; i++)

{

if (this.listFlat[i].id == id) { return i; }

}

return -1;

}

public int nextId //поиск нового айди

{

get

{

int m = int.MinValue;

foreach (Flat f in this.listFlat)

{

if (f.id > m)

{

m = f.id;

}

}

return m + 1;

}

}

#endregion

public Flats(string fileName) //конструктор по строке

{

string[] massivStrok = File.ReadAllLines(fileName, Encoding.Default);

for (int i = 0; i < massivStrok.Length; i++)

{

listFlat.Add(new Flat(massivStrok[i]));

}

}

/////////// Вспомогательные методы

public Flat getFlat(int number)

{

try

{

return listFlat[number];

}

catch (Exception)

{

throw;

}

}

public void removeRow(int rowNumber)

{

try

{

this.listFlat.RemoveAt(rowNumber);

}

catch (ArgumentOutOfRangeException e)

{

throw new ArgumentOutOfRangeException("Ты не то удаляешь, скорее всего такого количества строк нет..." + e.Data);

}

catch (IndexOutOfRangeException)

{

throw new IndexOutOfRangeException("Ошибка индекс за пределами диапазона");

}

}

public void addRow(Flat f)

{

this.listFlat.Add(f);

}

public void addRow(string s)

{

this.listFlat.Add(new Flat(s));

}

public void changeRow(int rowNumber, Flat newFlat)

{

this.listFlat[rowNumber] = newFlat;

}

public void saveFlatsToFile()

{

string sOut = "";

foreach (Flat f in this.listFlat)

{

sOut = sOut + f.ToString(";") + "\n";

}

File.WriteAllText("data.csv", sOut, Encoding.Default);

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace DB\_RieltorCompany

{

class Flat

{

#region Свойства

public int id

{

get

{

return Convert.ToInt32(Items[0]);

}

set

{

Items[0] = value.ToString();

}

}

public string realtyType

{

get

{

return Items[1];

}

set

{

Items[1] = value.ToString();

}

}

public int roomsCount

{

get

{

return Convert.ToInt32(Items[2]);

}

set

{

Items[2] = value.ToString();

}

}

public int floor

{

get

{

return Convert.ToInt32(Items[3]);

}

set

{

Items[3] = value.ToString();

}

}

public int area

{

get

{

return Convert.ToInt32(Items[4]);

}

set

{

Items[4] = value.ToString();

}

}

public int price

{

get

{

return Convert.ToInt32(Items[5]);

}

set

{

Items[5] = value.ToString();

}

}

public string address

{

get

{

return Items[6];

}

set

{

Items[6] = value.ToString();

}

}

public string description

{

get

{

return Items[7];

}

set

{

Items[7] = value.ToString();

}

}

public int clientID

{

get

{

return Convert.ToInt32(Items[8]);

}

set

{

Items[8] = value.ToString();

}

}

#endregion

private string[] Items; //массив объектов строки

public Flat(string stroka)

{

Items = stroka.Split(';');

}

public string ToString(string delimeter)

{

string outS = "";

foreach (string item in this.Items)

{

outS += item + delimeter;

}

return outS.Substring(0, outS.Length - 1);

}

public override string ToString()

{

return this.ToString(";");

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.IO;

namespace DB\_RieltorCompany

{

class Clients

{

private List<Client> listClient = new List<Client>();

const string titleConst = "id;namber;name;adres;mail";

public int clientsCount { get { return listClient.Count; } }

#region Properties

public string Title

{ get { return titleConst; } }

public int nextId

{

get

{

int m = int.MinValue;

foreach (Client f in this.listClient)

{

if (f.id > m)

{

m = f.id;

}

}

return m + 1;

}

}

#endregion

public Clients(string fileName)

{

string[] massivStrok = File.ReadAllLines(fileName, Encoding.UTF8);

for (int i = 0; i < massivStrok.Length; i++)

{

listClient.Add(new Client(massivStrok[i]));

}

}

public Client getClient(int number)

{

try

{

return listClient[number];

}

catch (Exception)

{

throw;

}

}

public void removeRow(int rowNumber)

{

try

{

this.listClient.RemoveAt(rowNumber);

}

catch (ArgumentOutOfRangeException e)

{

throw new ArgumentOutOfRangeException("Ты не то удаляешь, скорее всего такого количества строк нет..." + e.Data);

}

catch (IndexOutOfRangeException)

{

throw new IndexOutOfRangeException("Ошибка индекс за пределами диапазона");

}

}

public void addRow(Client f)

{

this.listClient.Add(f);

}

public void addRow(string s)

{

this.listClient.Add(new Client(s));

}

public void changeRow(int rowNumber, Client newFlat)

{

this.listClient[rowNumber] = newFlat;

}

public void saveFlatsToFile()

{

string sOut = "";

foreach (Client f in this.listClient)

{

sOut = sOut + f.ToString(";") + "\n";

}

File.WriteAllText("data.csv", sOut, Encoding.Default);

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace DB\_RieltorCompany

{

class Client

{

#region свойства

public int id

{

get

{

return Convert.ToInt32(ItemsClient[0]);

}

set

{

ItemsClient[0] = value.ToString();

}

}

public int phoneNumber

{

get

{

return Convert.ToInt32(ItemsClient[1]);

}

set

{

ItemsClient[1] = value.ToString();

}

}

public string name

{

get

{

return ItemsClient[2];

}

set

{

ItemsClient[2] = value.ToString();

}

}

public string adress

{

get

{

return ItemsClient[3];

}

set

{

ItemsClient[3] = value.ToString();

}

}

public string eMail

{

get

{

return ItemsClient[4];

}

set

{

ItemsClient[4] = value.ToString();

}

}

#endregion

private string[] ItemsClient;

public Client(string stroka)

{

ItemsClient = stroka.Split(';');

}

public string ToString(string delimeter)

{

string outS = "";

foreach (string item in this.ItemsClient)

{

outS += item + delimeter;

}

return outS.Substring(0, outS.Length - 1);

}

public override string ToString()

{

return this.ToString(";");

}

}

}

*Результат выполнения программы:*

