ЛЕКЦИЯ 13

- 18.10.2022
- Формат файла CSV
- Кортежи значений

ЦЕЛИ ЛЕКЦИИ

- Обсудить чтение и запись тестовых файлов
- Познакомится с CSV-форматом файла
- Познакомится с типом данных «кортеж значений»

НЕМНОГО ТЕРМИНОВ

Возврат каретки (Carriage Return, CR) (`\r`, `0x0D` в шестнадцатеричной, 13 в десятичной системе счисления) – перемещает курсор в начало строки, не переходя на следующую строку

Перевод строки (Line Feed, LF) (`\n`, `0x0A` в шестнадцатеричной, 10 в десятичной системе счисления – перемещает курсор на следующую строку, не возвращаясь в начало строки

CRLF = CR + LF

ФОРМАТ CSV

COMMA-SEPARATED VALUES (CSV)

CSV – Comma Separated Values - термин, используемый для обозначения файлов, в которых табличные данные представлены в виде текста

```
eruid,description
batman,uses technology
superman,flies through the air
spiderman,uses a web
ghostrider, rides a motorcycle
#GROUP_OBJECT_PROFILE#accessgroupGroupProfile
cn,description
daredevil,this group represents daredevils
superhero,this group represents superheroes
```

ПРЕДСТАВЛЕНИЕ ТЕКСТОВЫХ ТАБЛИЧНЫХ ДАННЫХ

This is the CSV file viewed as spread sheet:

Login Email	Identi-fier	One-time password	Recov-ery code	First name	Last name	Department
rachel@yourcompany.com	9012	12se74	rb9012	Rachel	Booker	Sales
laura@yourcompany.com	2070	04ap67				
craig@yourcompany.com	4081	30no86				
mary@yourcompany.com	9346	14ju73				
jamie@yourcompany.com	5079	09ja61				

email-password-recovery-code.csv - Notepad

File Edit Format View Help

```
Login email;Identifier;One-time password;Recovery code;First name;Last name;Department;Location
rachel@example.com;9012;12se74;rb9012;Rachel;Booker;Sales;Manchester
laura@example.com;2070;04ap67;lg2070;Laura;Grey;Depot;London
craig@example.com;4081;30no86;cj4081;Craig;Johnson;Depot;London
mary@example.com;9346;14ju73;mj9346;Mary;Jenkins;Engineering;Manchester
jamie@example.com;5079;09ja61;js5079;Jamie;Smith;Engineering;Manchester
```

Ln 3, Col 61 | 100% | Unix (LF) | UTF-8

Скачать файл можно по ссылке (https://support.staffbase.com/hc/en-us/article_attachments/360009197091/email-password-recovery-code.csv)

COMMA-SEPARATED VALUES (CSV)

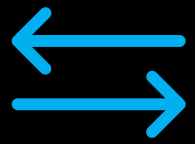
Неформальное описание:

CSV – Comma Separated Values (http://data.okfn.org/doc/csv)

- *RFC **4180** – **Common** Format and MIME Type for **Comma-Separated Values** (CSV) Files* (http://tools.ietf.org/html/rfc4180)
- *RFC **7111** – URI Fragment Identifiers for the text/csv Media Type* (http://tools.ietf.org/html/rfc7111)

ГДЕ ПРИМЕНЯЕТСЯ CSV-ФОРМАТ ФАЙЛОВ

Задачи управления данными



Импорт / Экспорт данных

Создания бекапов

Миграция данных

Задачи [интеллектуального] анализа данных

Анализ данных

Задачи машинного обучения

Машинное обучение

СТРОКИ CSV-ФАЙЛОВ

На одной строке – одна запись

- Строки разделяются символом LF или CRLF
- Разделитель строк может отсутствовать у последней строки файла

```
100,2000,55,78,,0
0,1,200,99,,44
```

CRLF

CRLF

```
100,2000,55,78,,0
0,1,200,99,,44
```

CRLF

Разделитель строки может быть включён в данные, т.е. на одной строке помещается больше записей – это тоже приемлемо

РАЗДЕЛИТЕЛИ В CSV-ДАННЫХ

- **Запятая**
- **Точка с запятой**
- **Вертикальная черта**
- **Табуляция**

```
Console.WriteLine('\u002c'); // ,
Console.WriteLine("\u0009"); // Tab
Console.WriteLine("\u003b"); // ;
Console.WriteLine("\u007c"); // |
```

```
100,2000,55,78,,0
0,1,200,99,,44
```

```
100;2000;55;78;;0
0;1;200;99;;44
```

```
100|2000|55|78||0
0|1|200|99||44
```

CSV-формат – не стандартизирован, могут встречаться и другие разделители

ЦИТАТЫ В CSV-ДАННЫХ

Если в данных встречаются запятые, кавычки, переводы строк и др., они должны быть **окружены двойными кавычками**

- Цитата – это валидный способ добавить пробелы в начало или конец строки, не входящие в цитату начальные и конечные пробелы игнорируются
  - *Аналогично с нулями слева в нумералах*

```
1;Иванов;Иван;”Москва,Покровский б-р,11”
2;Сидоров;Сидор;”Москва,Мясницкая ул,19”
```

```
Console.WriteLine('\u0022');
```

```
1101;”0001”;1010
0101;”1010”;1100
```

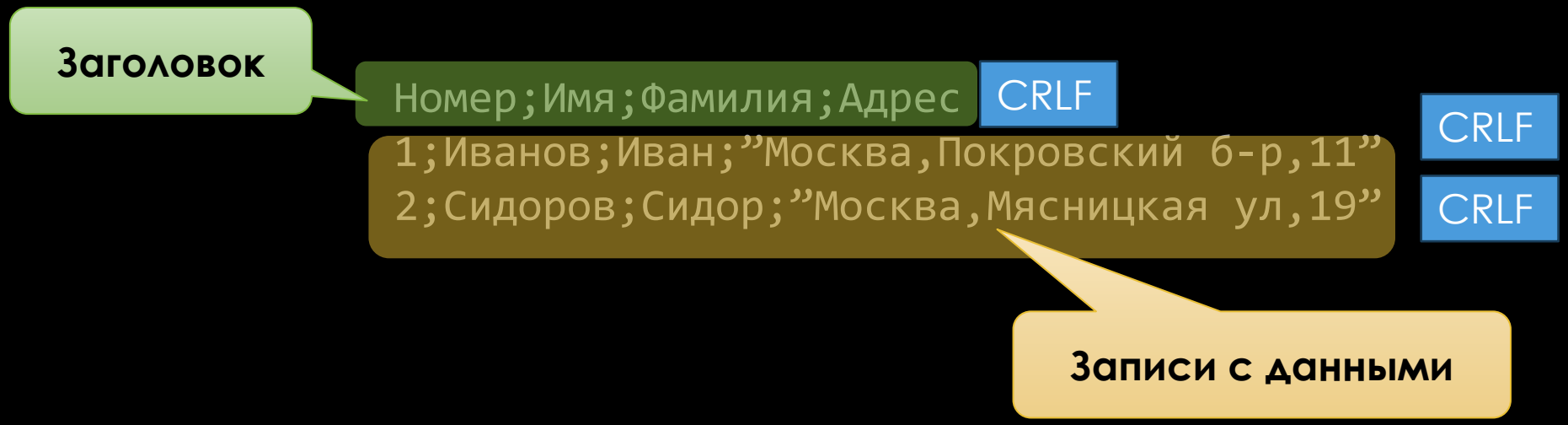
Чтобы подчеркнуть важность интерпретации «цитаты» буквально, иногда её предваряют знаком = (\u003d)

```
1;Иванов;Иван;=”Москва,Покровский б-р,11”
2;Сидоров;Сидор;=”Москва,Мясницкая ул,19”
```

ЗАГОЛОВОК CSV-ФАЙЛА

Заголовок содержит имена полей данных записи

- добавляется перед первой строкой с записями о данных в том же порядке, в котором данные представлены в записи, имена отделяются разделителем



ПРОБЛЕМЫ ПРИ ИСПОЛЬЗОВАНИИ CSV-ФАЙЛОВ

Отсутствие стандарта	К получаемым данным приходится относиться максимально лояльно
Требовательность к памяти	Многие стандартные обработчики лимитируют объем CSV-файлов
Проблемы передачи данных	Нарушение формата представления данных приводит к ошибкам импорта

ВОПРОСЫ БЕЗОПАСНОСТИ

Целостность данных

Утечки данных

Нехватка механизмов валидации данных

CSV Injection aka Formula Injection

ПРИМЕР. ЧТЕНИЕ CSV-ФАЙЛА «В ЛОБ»

```
public class Program{
    static string path = @"email-password-recovery-code.csv";
        static char[] invalidPathChars = Path.GetInvalidPathChars();
        public static void Main()
    {
        string[] rowData = null;
        if (path.IndexOfAny(invalidPathChars) != -1 || File.Exists(path))
        {
            rowData = File.ReadAllLines(path);
        }
        foreach(string str in rowData)
        {
            Console.WriteLine(str);
        }
    }
}
```

```
Login email;Identifier;One-time password;Recovery code;First name;Last name;Department;Location
rachel@example.com;9012;12se74;rb9012;Rachel;Booker;Sales;Manchester
laura@example.com;2070;04ap67;lg2070;Laura;Grey;Depot;London
craig@example.com;4081;30no86;cj4081;Craig;Johnson;Depot;London
mary@example.com;9346;14ju73;mj9346;Mary;Jenkins;Engineering;Manchester
jamie@example.com;5079;09ja61;js5079;Jamie;Smith;Engineering;Manchester
```

ПРИМЕР. ЧТЕНИЕ CSV-ФАЙЛА «В ЛОБ»

```
public class Program {
    static string path = @"email-password-recovery-code.csv";
    static char[] invalidPathChars = Path.GetInvalidPathChars();
    public static void Main()
    {
        string[] rowData = null;
        if (path.IndexOfAny(invalidPathChars) != -1 || File.Exists(path))
        {
            rowData = File.ReadAllLines(path);
        }
        string[][] splittedData = new string[rowData.Length][];
        for (int i = 0; i < splittedData.Length; i++)
        {
            splittedData[i] = rowData[i].Split(';');

        }

    }
}
```

```
foreach(string[] row in splittedData)
        {
            foreach(string item in row)
            {
                Console.Write($"{item}\t");
            }
            Console.WriteLine();
        }
```

```
Login email     Identifier      One-time password       Recovery code   First name      Last name       Department
ocation
rachel@example.com      9012    12se74  rb9012  Rachel  Booker  Sales   Manchester
laura@example.com       2070    04ap67  lg2070  Laura   Grey    Depot   London
craig@example.com       4081    30no86  cj4081  Craig   Johnson Depot   London
mary@example.com        9346    14ju73  mj9346  Mary    Jenkins Engineering     Manchester
jamie@example.com       5079    09ja61  js5079  Jamie   Smith   Engineering     Manchester
```

ПРИМЕР. ЧТЕНИЕ CSV-ФАЙЛА «В ЛОБ»

```
static string path = @"email-password-recovery-code.csv";
static char[] invalidPathChars = Path.GetInvalidPathChars();
```

Вывод:

```
Personal Data:
(First name, Last name, Login email)
(Rachel, Booker, rachel@example.com)
(Laura, Grey, laura@example.com)
(Craig, Johnson, craig@example.com)
(Mary, Jenkins, mary@example.com)
(Jamie, Smith, jamie@example.com)
```

```
string[] rowData = null;
if (path.IndexOfAny(invalidPathChars) != -1 || File.Exists(path))
{ rowData = File.ReadAllLines(path);  }
string[][] splittedData = new string[rowData.Length][];
ValueTuple<string, string, string>[] personalData = new ValueTuple<string, string,
string>[splittedData.Length];
for (int i = 0; i < splittedData.Length; i++)
{
   splittedData[i] = rowData[i].Split(';');
   personalData[i] = new
ValueTuple<string,string,string>(splittedData[i][4],splittedData[i][5],splittedData[i][0]);
}
Console.WriteLine("Personal Data:");
foreach(var item in personalData)     {    Console.WriteLine(item);    }
```

КОРТЕЖИ

Структура ValueTuple

ИМЕНОВАНИЕ ПОЛЕЙ КОРТЕЖЕЙ-ЗНАЧЕНИЙ

Кортеж [tuple] длины n (также упорядоченный набор длины n или -ка) – упорядоченная последовательность ровно из n элементов любой природы (где n – натуральное число).

- Отметим, что n-ками чаще всего обозначают короткие кортежы: «пара», «тройка» и т.д.
- В математике индексы элементов кортежа называют координатами. Кортежи часто становятся основой построения составных типов

Свойства кортежей как типа:

- Тип кортежа является изменяемым типом значения
- Элементы кортежа – открытые поля, т.е. хранят данные
- Для использования требуют тип `System.ValueTuple` и связанные обобщённые типы

ИМЕНОВАНИЕ ПОЛЕЙ КОРТЕЖЕЙ-ЗНАЧЕНИЙ

Допустимо явно указывать имена полей кортежа-значения как при объявлении типа, так и при его инициализации (при использовании `var`)

- Начиная с C# 7.1, компилятор может выводить имена переменных кортежей-значений из имён соответствующих переменных при инициализации.

```
// Явное именование параметров при объявлении.
(double perimeter, double area) valueTuple1 = (11.1, 22.2);
```

```
// Автоматическое определение, с именованными полями.
var valueTuple2 = (x: 1, y: 2);
```

Имена полей кортежа-значения существуют только на этапе компиляции, не учитываются при сравнении на равенство и впоследствии заменяются именами по умолчанию (Item1, Item2, …)

ИМЕНОВАНИЕ ПОЛЕЙ КОРТЕЖЕЙ

```
// Явное именование параметров при объявлении.
(double perimeter, double area) valueTuple1 = (11.1, 22.2);
```

```
// Автоматическое определение, с именованными полями.
var valueTuple2 = (x: 1, y: 2);
```

```
// Использование явно заданного имени и имени по умолчанию.
Console.WriteLine(valueTuple2.x + valueTuple2.Item2);
```

```
// Хотя имена различные, типы полностью соответствуют.
(double density, double weight) valueTuple3 = valueTuple1;
```

```
// C# 7.1: Определение имён полей по переданным переменным.
int mass = 30, energy = 9000;
var valueTuple4 = (mass, energy);
Console.WriteLine(valueTuple4.mass + valueTuple4.energy);
```

КОРТЕЖИ-ЗНАЧЕНИЯ В МЕТОДАХ: ПРИМЕР

```
static (int Min, int Max, bool HasMinMax) MinMaxElement(int[] array)
{
    if (array == null || array.Length == 0)
        return (-1, -1, true);

    var output = (Min: array[0], Max: array[0], IsEmpty: false);
    for (int i = 1; i < array.Length; i++) {
        if (array[i] < output.Min)
            output.Min = array[i];
        if (array[i] > output.Max)
            output.Max = array[i];
    }
    return output;
}
```

В типе возвращаемого значения можно указывать имена элементов

Явное именование элементов при инициализации

СРАВНЕНИЕ КОРТЕЖЕЙ-ЗНАЧЕНИЙ

В C# 7.3 добавлено **сравнение кортежей-значений** с помощью операторов == и !=

Сравнение производится *поэлементно, без учёта имён параметров* и возможно *только для кортежей-значений с одинаковым количеством элементов*

При сравнении будут вычислены все выражения, результаты вычисления которых используются в качестве элементов

СРАВНЕНИЕ КОРТЕЖЕЙ-ЗНАЧЕНИЙ: ПРИМЕР

```
using System;
// Для сравнения кортежей-значений типы элементов не
// обязаны совпадать - достаточно неявного приведения.
(int, short) valueTuple1 = (21, 12);
(long, byte) valueTuple2 = (21, 12);
// Выведет True.
Console.WriteLine(valueTuple1 == valueTuple2);
```

```
True
False
```

```
// Все выражения будут вычислены для сравнения:
// Выведется False - по умолчанию Math.Round округляет
// до ближайшего чётного, кортежи равны.
Console.WriteLine((Math.Round(2.5), Math.Truncate(3.5)) != (Math.Round(1.5),
Math.Truncate(3.2)));
```

ДЕКОНСТРУКЦИЯ КОРТЕЖЕЙ-ЗНАЧЕНИЙ

Для записи элементов кортежей-значений в переменные одной операцией можно использовать деконструкцию

- Синтаксис деконструкции :

```
(int, string, double) valueTuple = (101, "dalmatian", 46.68);
```

Правила деконструкции:

- Перед круглыми скобками нельзя указать общий тип для всех переменных, такой синтаксис корректен только для var
- Каждый элемент кортежа-значения необходимо присвоить переменной, но можно использовать пустую переменную «_»
- Смешение объявления переменных и присваивания значений уже существующим при деконструкции не допускается (до C# 10)

ПРИМЕР ДЕКОНСТРУКЦИИ КОРТЕЖЕЙ

```
(int, string, double) valueTuple = (101, "dalmatian", 46.68);
// I способ: типы всех переменных указываются явно.
(int count1, string breed1, double mass1) = valueTuple;
// II способ: тип определяется компилятором.
var (count2, breed2, mass2) = valueTuple;
```

```
// III способ: деконструкция в уже существующие переменные.
int count3;
string breed3;
double mass3;
(count3, breed3, mass3) = valueTuple;
// IV способ: смешение явного определения типов и автоматического
// Довольно громоздко, по этой причине не рекомендуется.
(int count4, var breed4, var mass4) = valueTuple;
// Деконструкция с использованием пустых переменных:
var (_, breed, _) = valueTuple;
```

ИСПОЛЬЗОВАННАЯ ЛИТЕРАТУРА

- https://docs.microsoft.com/en-us/dotnet/standard/io/common-i-o-tasks
- https://docs.microsoft.com/en-us/dotnet/api/system.io.fileshare?view=net-6.0
- *CSV File Formats* (http://www.discoverysoftware.co.uk/SGHelp/dwLoader%20Format.htm)
- https://www.oneschema.co/blog/csv-files
- Import a CSV file in Google Ads Editor
- Prepare a CSV file https://support.google.com/google-ads/editor/answer/56368
- CSV file columns https://support.google.com/google-ads/editor/answer/57747?hl=en&ref_topic=2986631
- https://www.ibm.com/docs/en/sim/6.0.2?topic=schedules-example-comma-separated-value-csv-file
- ValueTuple Структура (System) | Microsoft Learn