

# Citire/Scriere fisiere – partea a 2a

## Scrierea si citirea fisiereleor CSV:

Fisierele CSV sunt un tip de fisiere folosite pentru a stoca date de tipuri diferite asemanatoare datelor in bazele de date. Cel mai mult sunt folosite in a stoca date care contin campuri de acelasi tip. Aceste fisiere pot fi citite si create folosind Microsoft Office Excel. Ex. de date in CSV: Id,Nume,Prenume,Data Nasterii etc. Spre deosebire de bazele de date unde fiecare camp isi are casuta lui intr-un tabel, in CSV un camp este delimitat de o virgule.

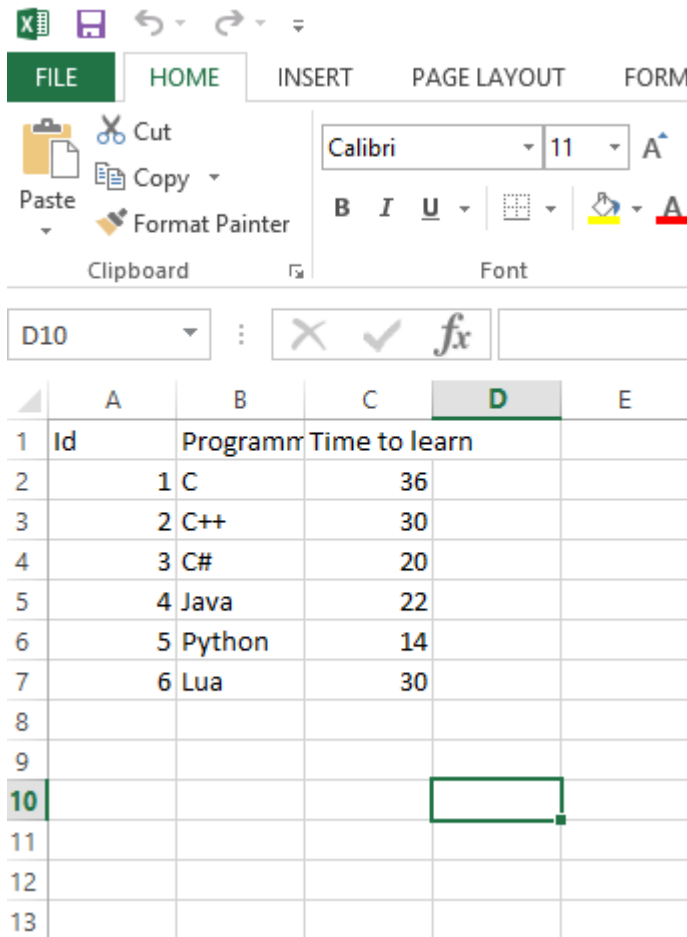
Pentru a citi dintr-un fisier CSV in C# vom folosi o librarie care face parte din limbajul Visual Basic si anume `using Microsoft.VisualBasic.FileIO;`

Aceasta librarie ne va lasa sa folosim clasa `TextFieldParser`, o clasa foarte utila in parsarea fisiereleor care respecta o secventa sau un pattern.

```
ReadWrite - ReadWrite.Form1 - button1_Click(object sender, EventArgs e)
64
65     }
66     if (comboBox1.SelectedItem.ToString() == "CSV")
67     {
68         op = new OpenFileDialog();
69         op.InitialDirectory = Path.Combine(Environment.CurrentDirectory, @"..\..\");
70         if (op.ShowDialog() == DialogResult.OK)
71         {
72             textBox3.Text = op.FileName;
73             TextFieldParser csvParser = new TextFieldParser(op.OpenFile());
74             csvParser.TextFieldType = FieldType.Delimited;
75             csvParser.SetDelimiters(",");
76             while (!csvParser.EndOfData)
77             {
78                 //Processing row
79                 string[] fields = csvParser.ReadFields();
80                 foreach (string field in fields)
81                 {
82                     textBox1.Text += field + "|";
83                 }
84                 textBox1.Text += Environment.NewLine;
85             }
86             csvParser.Close();
87         }
88     }
```

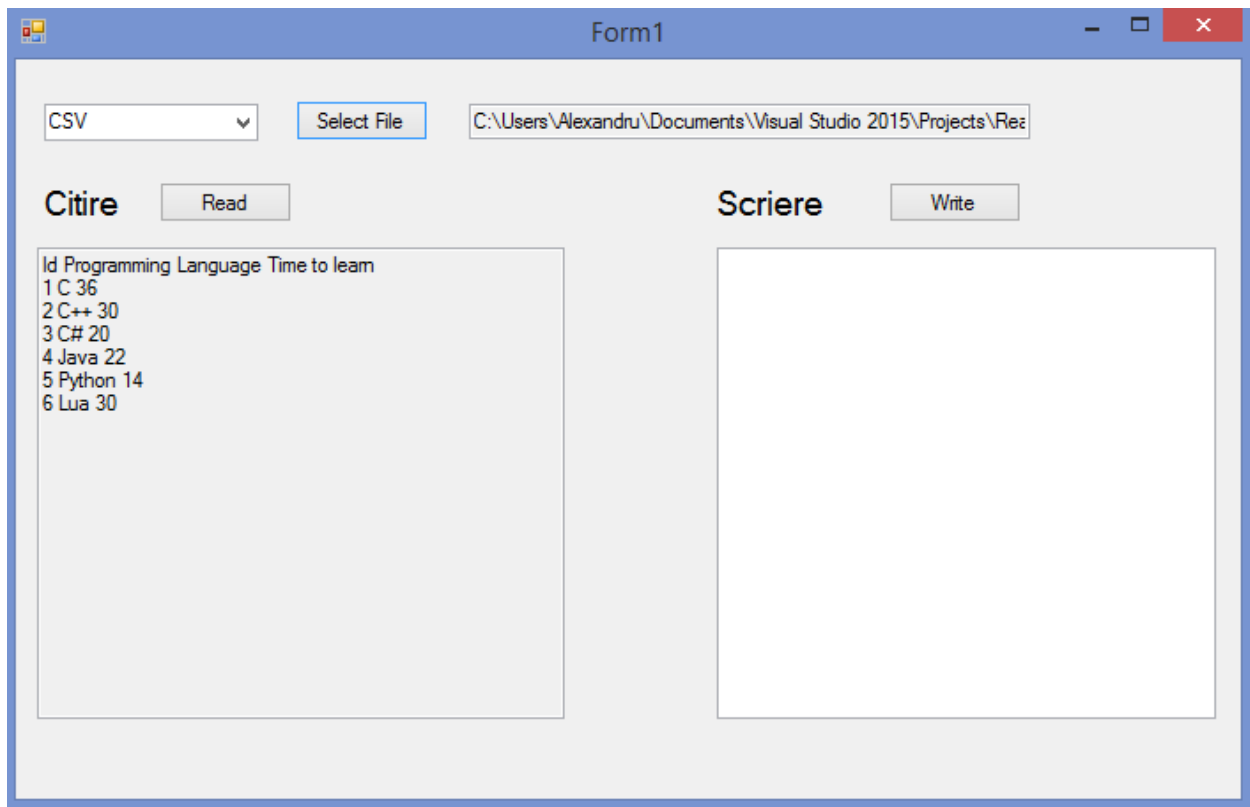
In continuarea proiectului din lucrarea precedent vom scrie urmatorul cod. Spre deosebire de citirea unui fisier text sau unui fisier XML vom folosi clasa `TextFileParser`, careia ii pasam un `Stream` (`op.OpenFile()`). Apoi vom seta parserul ca avand field-uri delimitate de virgula.

Intr-un while care va parsa fisierul de la inceput pana la sfarsit vom adauga intr-un vector de tip string fiecare camp de pe un rand, iar apoi vom parcurge vectorul si il vom afisa. Dupa ce s-a terminat fisierul de parsat, vom inchide parser-ul folosind metoda Close(). Iar intr-un CSV care va avea forma din imaginea de mai jos, rezultatul in aplicatie va trebui sa fie cel din cea de-a doua imagine.



The image shows a screenshot of the Microsoft Excel interface. The ribbon at the top includes 'FILE', 'HOME', 'INSERT', 'PAGE LAYOUT', and 'FORM'. The 'HOME' ribbon is active, showing options for 'Clipboard' (Cut, Copy, Paste, Format Painter) and 'Font' (Calibri, size 11, Bold, Italic, Underline, text color, background color). The active cell is D10. Below the ribbon is a formula bar with a dropdown menu showing 'D10', a clear button (X), a checkmark, and a function button (fx). The main area is a spreadsheet with columns A through E and rows 1 through 13. The data is as follows:

|    | A  | B        | C             | D | E |
|----|----|----------|---------------|---|---|
| 1  | Id | Programm | Time to learn |   |   |
| 2  | 1  | C        | 36            |   |   |
| 3  | 2  | C++      | 30            |   |   |
| 4  | 3  | C#       | 20            |   |   |
| 5  | 4  | Java     | 22            |   |   |
| 6  | 5  | Python   | 14            |   |   |
| 7  | 6  | Lua      | 30            |   |   |
| 8  |    |          |               |   |   |
| 9  |    |          |               |   |   |
| 10 |    |          |               |   |   |
| 11 |    |          |               |   |   |
| 12 |    |          |               |   |   |
| 13 |    |          |               |   |   |

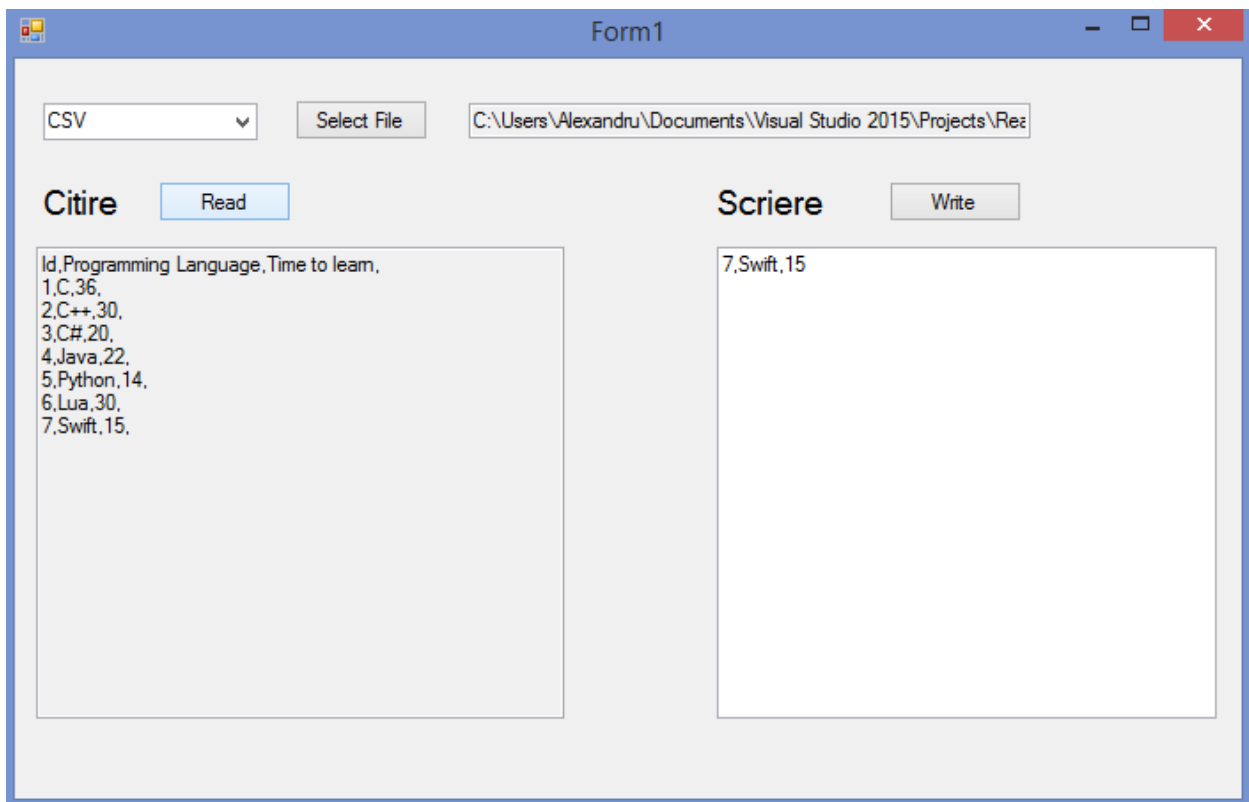
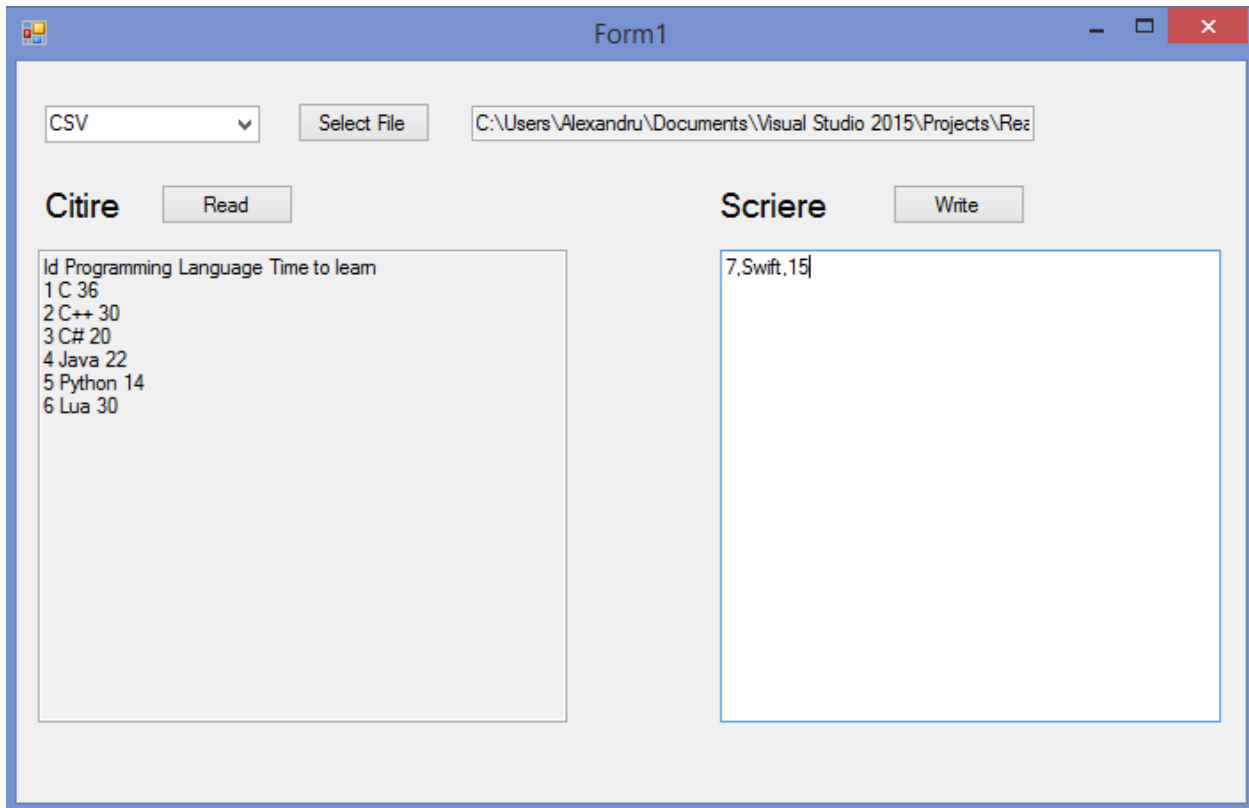


Pentru o scrierea intr-un fisier CSV, cea mai usoara metoda este aceea de a folosi scrierea de la fisierele text, iar virgula trebui pusa manual sau facuta o functie pentru a pune virgula dupa fiecare camp introdus.

```
if (comboBox1.SelectedItem.ToString() == "CSV")
{
    Stream stream = new FileStream(op.FileName, FileMode.Append, FileAccess.Write, FileShare.Write);
    StreamWriter write = new StreamWriter(stream);

    write.WriteLine(textBox2.Text);
    write.Close();
}
```

Iar pentru cea de-a doua metoda se poate folosi: `textBox2.Text.Replace(" ", ",");` functie care va inlocui spatiile cu virgule.



## Scrierea si citirea din fisiere Json

Pentru scrierea si citirea din fisire Json vom folosi o librerie externa:

1. Click dreapta pe Referenced si click stanga pe "Manage NuGet Packages..."
2. Apoi cautati dupa Newtonsoft.Json si instalati pachetul
3. In cadrul fisierului unde vrem sa facem citirea vom introduce `using Newtonsoft.Json;`

```
if (comboBox1.SelectedItem.ToString() == "JSON")
{
    op = new OpenFileDialog();
    op.InitialDirectory = Path.Combine(Environment.CurrentDirectory, @"..\..\");
    if (op.ShowDialog() == DialogResult.OK)
    {
        textBox3.Text = op.FileName;
        StreamReader streamReader = new StreamReader(op.OpenFile());
        JsonTextReader jsonParser = new JsonTextReader(new StringReader(streamReader.ReadToEnd()));
        while (jsonParser.Read())
        {
            if (jsonParser.Value != null)
            {
                textBox1.Text += jsonParser.Value;
            }
            textBox1.Text += Environment.NewLine;
        }
        jsonParser.Close();
        streamReader.Close();
    }
}
```

Spre deosebire de restul fisierelor, la Json este putin mai complicat. Noi cand selectam un fisier ne vine ca un Stream, pe cand clasa JsonTextReader primeste ca argument un StringReader. Astfel pentru a putea folosi fisierul, vom converti prima data stream-ul intr-un StreamReader (aceasta se face prin crearea unei instate de tipul StreamReader careia ii trimitem ca parametru Stream-ul care vine din citirea fisierului) si apoi convertim din nou StreamReader-ul intr-un StringReader prin crearea unei noi instante de tip StreamReader careia ii dam ca parametru streamReader.ReadToEnd().

Intr-un while parsam datele primite de parser si daca valoarea pe care a citit-o este valida o afisam. Rezultatul pentru un Json simplu este urmatorul:

```
144     }
145     if (comboBox1.SelectedItem.ToString() == "JSON")
146     {
147         string[] text = textBox2.Text.Split(' ');
148         Stream stream = new FileStream(op.FileName, FileMode.Truncate, FileAccess.Write, FileShare.Write);
149         StreamWriter write = new StreamWriter(stream);
150         JsonWriter jsonWriter = new JsonTextWriter(write);
151         jsonWriter.WriteStartObject();
152         for (int i = 0; i < text.Length; i += 2)
153         {
154             jsonWriter.WritePropertyName(text[i]);
155             jsonWriter.WriteValue(text[i+1]);
156         }
157         jsonWriter.WriteEndObject();
158         jsonWriter.Close();
159         write.Close();
160         stream.Close();
161     }
162 }
163
164 }
```

Form1

JSON Select File C:\Users\Alexandru\Documents\Visual Studio 2015\Projects\Rea

**Citire** Read

title  
Product  
description  
A product from Acme's catalog  
type  
object

**Scriere** Write

Pentru scriere vom folosi clasa `JsonWriter`, dar la fel ca si la citire trebuie facute cateva conversii. Din `Stream` de data asta facem o conversie intr-un `TextWriter`, iar apoi din `TextWriter` in `JsonWriter`.

The screenshot shows the Visual Studio IDE with a C# file named Form1.cs. The code is as follows:

```
144     }
145     if (comboBox1.SelectedItem.ToString() == "JSON")
146     {
147         string[] text = textBox2.Text.Split(' ');
148         Stream stream = new FileStream(op.FileName, FileMode.Truncate, FileAccess.Write, FileShare.Write);
149         StreamWriter write = new StreamWriter(stream);
150         JsonSerializer jsonWriter = new JsonSerializer(write);
151         jsonWriter.WriteStartObject();
152         for (int i = 0; i < text.Length; i += 2)
153         {
154             jsonWriter.WritePropertyName(text[i]);
155             jsonWriter.WriteValue(text[i+1]);
156         }
157         jsonWriter.WriteEndObject();
158         jsonWriter.Close();
159         write.Close();
160         stream.Close();
161     }
162 }
```

Pentru a incepe un document Json trebuie folosita metoda WriteStartObject() si pentru a inchide un document Json trebuie folosita metoda WriteEndObject(). Iar pentru a adauga o proprietate si o valoare se folosesc WritePropertyName(), respective WriteValue().

The screenshot shows a Windows Form application titled "Form1". At the top, there is a dropdown menu set to "JSON", a "Select File" button, and a text box containing the file path "C:\Users\Alexandru\Documents\Visual Studio 2015\Projects\Rea". Below this, there are two sections: "Citire" with a "Read" button, and "Scriere" with a "Write" button. The "Citire" section contains a text area with the following JSON output:

```
title
Product
description
A product from Acne's catalog
type
object
```

The "Scriere" section contains a text area with the input text: "Title Home Description House".

Form1

JSON Select File C:\Users\Alexandru\Documents\Visual Studio 2015\Projects\Rea

**Citire** Read

**Sciere** Write

title  
Product  
description  
A product from Acne's catalog  
type  
object

Title  
Home  
Description  
House

Title Home Description House