Lab 08b: Reading a Coma Separated Value (.csv) File

Continuing from the previous exercise, we will look at another type of text file, called coma-separated values (.csv).

Unlike a typical text file where its content is read in entirety or line by line, a .csv file stores its values continuously, and separated by a delimiter. A .csv file is commonly used to store values of a particular set of data.

Below is the example to contrast a typical text file and a .csv file:

Text file content

> This is a text file, with some text in it.

.csv file content

> B7365,C814,D744,E392,B7362,C875,D724,E343,B7363,C824,D714,E354,B7361,C864,D734,E432,
> B7368,C834,D244,E332

File needed

Create a .cvs file with the above sample content, and save it as TextWithValue.csv. Remember that the file extension can be customised to suit your preference.

Reading a .csv file – sequential values

A delimiter separates values in .csv file, and often these values are stored in a sequential manner. Hence, the content needs to be read using *.ReadAllText* method, with an additional sub-method *.Split ('<delimiter>');*

String[ ] as_Content = System.IO.File.ReadAllText ("Textwithvalue.csv").Split (',');

Authored by Iwan Tjhin
(Iwan.Tjhin@whitireia.ac.nz)

```
try
{
    //Similar to previous exercise application, read the entire .csv file into an array
    //Using .Split keyword to automatically split the content, placing each one into an array slot
    //The delimiter/separator can be anything, depending what is used in the file
    //This example uses come (,) as its delimiter, as found in the TextWithValue.csv file
    string[] as_Content = System.IO.File.ReadAllText("Textwithvalue.csv").Split(',');


    //Dsiplay the content into screen, to see if it works
    //Using For loop to cycle through the array index to display the content
    for (int i_count = 0; i_count < as_Content.Length; i_count++)
    {
        Console.WriteLine(as_Content[i_count]);
    }

    //The specific content of an array can also be displayed by specifying its index
    Console.WriteLine("Specific code: " + as_Content[15]);
}
catch
{
    //Give an error message if an exception occurs
    Console.WriteLine("File not found or index is out of bound!");
}
```

Reading a .csv file – record values

Values stored in a .csv file can sometimes be in a record format, i.e. in 2 dimensional directions. Example of such values is as follows:

B7365,C814,D744,E392

B7362,C875,D724,E343

B7363,C824,D714,E354

B7361,C864,D734,E432

B7368,C834,D244,E332

It gets more complicated to read the values and store them in an array of variable, even if the array is a multi-dimensional array. There is no direct simple built-in method to convert multi-lines coma-separated values into a multi-dimensional array.

The conversion can be achieved by using a 2-step approach, i.e. read line by line, and for each line read value by value.

```
try
{
    //This approach is similar to previous exercise, reading line by line and store in an array variable
    //This variable contains all the values, each line stored in each array slot
    string[] as_AllContentByLine = System.IO.File.ReadAllLines("textMultiRowsWithValues.coolext");

    //Create a 2-dimensional array variable, to store the final version of the values
    //The first part is the column (we use 4 here as it's only 4 items in the .cvs file)
    //The second part it the row, which could grow depending on the size of records in the .cvs file
    //We obtain the total row number through the .Length attribute of the array that stores all data above
    string[,] as_Parts = new string[4, as_AllContentByLine.Length];


    //Using a For loop to cycle through the array
    //This is to read line by line (slot by slot of the array), and attempt to split values by its delimiter
    for (int i_count = 0; i_count < as_AllContentByLine.Length; i_count++)
    {
        //For each slot (line), split it, and place it into a temporary array
        string[] as_temp = as_AllContentByLine[i_count].Split(',');

        //From the temporary array, place it back into the final arary variable created above
        as_Parts[0, i_count] = as_temp[0];
        as_Parts[1, i_count] = as_temp[1];
        as_Parts[2, i_count] = as_temp[2];
        as_Parts[3, i_count] = as_temp[3];
    }

    //This is an example way to display a value from a multi-dimensional array above
    //Value is selected by targeting its index numnber, in format of [x,y]
    Console.WriteLine(as_Parts[1, 3]);
}
catch
{
    //Give an error message if an exception occurs
    Console.WriteLine("File not found or index is out of bound!");
}
```