Facebook.SpreadsheetConsole Project

Installation

In order to run this code, it is necessary to have the .NET Core 1.1.2 runtime installed. It is available on Windows, MacOS and Linux.

Setup instructions are here: https://www.microsoft.com/net/download/core#/runtime

How to Execute

To execute the application against all included test files using the Console host call the following script in the project's root folder:

```
./executeAllFiles.sh
```

To execute the application against all included test files using the xUnit Test Runner call the following script in the project's root folder:

```
./executeAllTests.sh
```

To execute the application against a particular file (from the root folder):

```
dotnet run --project ./Facebook.SpreadsheetConsole/Facebook.SpreadsheetConsole.csproj -- <inputFile>
<outputFile>
```

Assumptions

1. Cells can contain a `Formula` comprised of `Terms`, either references to other cells (`ReferenceTerm`), explicit values (`ValueTerm`) and operators (`OperandTerm`).
2. Negative values always have the minus (`-`) with no whitespace between it and the number.
3. Numbers are rounded to its 9th most significant decimal. Only significant digits will be printed.
4. There are no trailing commans (`,`) in any row. Trailing whitespace is ignored.
5. The last line of the input file may be empty. The output file will not have extra lines.
6. All arithmetic operations will be done using .NET's `decimal` type. This is a trade-off of memory in exchange for precision.
7. Rows may contain different numbers of cells. All cells must have a `Formula`.
8. All output files use UNIX File endings (`\n`) in all platforms (including Windows). Input files can have either UNIX or Windows Line Endings

Project Structure

Facebook.SpreadsheetConsole

A console application that hosts the `Facebook.Spreadsheets` code and manages `command-line arguments` and `File IO`.

Facebook.Spreadsheets

The main project. It is comprised of the following Key Components:

1. `Spreadsheet.cs`: Contains all the logic for Parsing (`Spreadsheet.Parsing.cs`), Evaluating (`Spreadsheet.Evaluation.cs`) and Writing (`Spreadsheet.Output.cs`) the results of a spreadsheet.

2. Terms/*: Contains classes to describe the terms that might appear in a cell's Formula. For this project, cells may contain either ReferenceTerms which point to another cell for its value, ValueTerms which contain an explicit numeric value or OperandTerms that specify an arithmetic operation (+ - * /).
3. Cells/Cell.cs: Defines a class to represent a Cell. Cells contain a set of Terms which describeFormula in Reverse Polish Notation.
4. Cells/CellParsing.cs: Has the logic required to parse a string representation of a Cell.
5. Exceptions/*: Contains several exceptions to describe error conditions in the code and provide specific feedback on each type of error.

Key Functions

- static Spreadsheet LoadSpreadsheetFromStream(...): Returns a new instance of Spreadsheet that contains all the cells that were parsed from the input stream. Does all parsing validations and builds a list of all cells that contain formulas that need to be evaluated. Any error throws an instance of SpreadsheetParserException. Runtime O(n) where n is the number of cells
- void Evaluate(): Evaluates the spreadsheet by iterating only over the cells that need evaluation. Cells are visited only once and the result of its evaluation is stored to eliminate redundancies. If a cell is visited twice during the same calculation a cycle is detected. Does all other evaluation validations. Any error throws an instance of SpreadsheetEvaluationException. Best-Case Runtime O(1) (when no cell needs to be evaluated). Worst-Case Runtime O(n).
- decimal CalculatePolishNotation(): Resolves a formula expressed in Reverse Polish Notation as a list of Terms. When this function is called, all ReferenceTerms have already been resolved. Runtime O(n) where n is the number of terms.

Facebook.Spreadsheets.Tests

A xUnit project to run all tests without hosting the console (doing all IO in memory).

Test Cases

The project includes 36 test cases. These are located in folders inside ./Facebook.SpreadsheetEvaluation/Facebook.Spreadsheets.Tests/testFiles

Valid Spreadsheets

The files located in ./valid/ are all Spreadsheets that can be sucessfully evaluated.

- 500KCells.txt: Test case with 504,000 Cells in which all but 1 are formulas.
- emptyFile.txt: To test the behavior with an empty File. Output should also be an empty file.
- facebook.txt: The example provided in the Spreadsheet Evaluation Problem Statement.
- fibonacci.txt: Calculating the Fibonacci Sequence up to the 73rd value.
- longReversePolish.txt: Calculating a simple spreadsheet using a formula with several terms.
- matrixMultiplication.txt: A matrix in which all values are multiplications of values from the first row with values from the first column.
- negativeMatrixDecimalDivision.txt: A matrix in which all values are divisions of values from the first row with values from the first column.
- negativeMatrixMultiplication.txt: A matrix in which all values are multiplications of values from the first row with values from the first column. In this test case some input and output values are negative.
- negativeMatrixMultiplicationLowerCase.txt: A matrix in which all values are multiplications of values from the first row with values from the first column. In this test case some Cell References are writen in lowercase.
- pow2Asc.txt: Calculate all the powers of two from 2^0 to 2^50.
- pow2Desc.txt: Divide 2^50 by two until reaching 1.
- randomNumbers.txt: Spreadsheet with varied operations using randomly generated numbers between -5000 and 5000.

- repeatEvaluation.txt: Test memoization of results when Cell was visited during branching in calculation.
- repeatEvaluation2.txt: Test memoization of results when Cell was visited during branching in calculation.
- simple.txt: A simple test case (similar to facebook.txt).
- singleReference.txt: A test for a cell that references another cell without any operations.

Spreadsheets with Parsing errors

The files located in ./invalidParsing/ are files that cannot be converted into a Spreadsheet for evaluation.

- invalidCellCharacters.txt, invalidCellCharacters2.txt, invalidCellCharacters3.txt: Contains characters that are never valid and inmediately invalidate a spreadsheet.
- invalidCellReference.txt, invalidCellReference2.txt: Contains a cell reference inside a formula that cannot be converted into a valid Cell Reference.
- invalidCellValue.txt: Contains a cell value that cannot be parsed into a valid number.
- invalidFormulaOperand.txt: Contains a formula that cannot be parsed because the operator is not + - * /.

Spreadsheets with Evaluation errors

The files located in ./invalidEval/ are files that can be parsed, but cannot be sucessfully evaluated.

- cycleInEval.txt: A simple cycle in which two cells reference each other.
- cycleInEvalComplex.txt: A cycle in which a cell A references another B that indirectly depends on A.
- divisionByZero.txt: A cell evaluates to zero and another cell uses that value as a divisor.
- divisionByZeroSimple.txt: A single cell which calculates 1 / 0.
- invalidCellReference.txt: A formula that references a cell that doesn't exist on the spreadsheet.
- invalidFormula.txt: Contains an invalid Formula.
- invalidFormulaOperandHuge.txt: A single cell in a file with 500K+ cells that contains a formula that cannot be evaluated because its missing an Operator.
- invalidPolish.txt: A cell that contains an invalid Polish Notation formula (missing a term)
- missingCellValue.txt, missingCellValue2.txt, missingCellValue3.txt: Contains a cell that does not have any value (formula or number).
- overflow.txt: A single cell which calculates decimal.MaxValue + 1.
- underflow.txt: A single cell which calculates decimal.MinValue - 1.