# Software Architecture project: the spreadsheet

## Introduction. Rules

The goal of the project is to build some of the core components of a spreadsheet, which can be used through a textual interface (it shall not be necessary to build a graphic user interface).

## Brief description of a basic spreadsheet

A spreadsheet is a group of cells arranged in a rectangular way.

Each cell is identified by its coordinate within the spreadsheet.

A cell coordinate shall be a tuple of two components. The first one shall identify the column. Its value shall be one letter or a sequences of letters (A, B, C ..., Z, AA, AB, etc). The second component shall identify the row, and its value shall be a number starting in 1. With this notation, the identifier for the cell in the first column and first row is A1.

Each cell shall have a content. The content of a cell may be of different types. At this point in time we will consider only three types of cell contents: **numerical**, **text**, and **formulas**.

Each content shall have a value.

The value of a text content shall be a string. They allow to organize the spreadsheet and add headers and notes to understand its contents.

The value of a numerical content shall be a number (integer or real).

Given a cell whose content is text, the program to be developed must provide mechanisms for obtaining either:

1. A number, if and only if the string is an empty string (in which case the number returned must be 0), or the string is a textual representation of a number (the string “1” would result in 1; the string “1.2” would result in number 1.2). If the program is requested to get a number when the string is not a textual representation of a number then the program must notify an error.
2. The string itself.

Given a cell whose content is numerical, the program to be developed must provide mechanisms for obtaining either:

1. The string being the textual representation of the number itself. If the cell is empty then the returned string shall be the empty string.
2. The number itself.

The value of a formula shall be the number resulting of computing the formula. A formula is a mathematical relationship or rule expressed in symbols, which in a spreadsheet always starts with the character '='. A simple example is "= 1 + 2" (hereinafter in the document, the text of the formula will appear within double quotes; however, these double quotes MUST NOT BE USED within the spreadsheet cell). The value returned by this formula is 3.

Formulas in a spreadsheet also allow us to calculate the value of a cell taking into account values of the contents in other cells; for example "= A1 + B1 + C1" indicates that the value of this formula is equal to the sum of the cells within the 3 first columns of the first row.

Finally, formulas in spreadsheet may also contain, as operands, functions. See section **Functions and Ranges** for more details on functions.

The program to be developed shall be able to process these formulas (likely coding them as objects) and compute the values returned by them.

## Computing values in cells

Let us assume that we have the following spreadsheet:

| | A | B | C |
|---|---|---|---|
| **1** | =C1+C2 | 4 | 1 |
| **2** | | | 2 |
| **3** | | TOTAL | =A1+B1 |

Note: notice that the string representing a formula is not enclosed within double quotes within the spreadsheet cells.

For calculating the value of the formula present in cell C3, the values of the contents of cells A1 and B1 are needed. It is said that the value of the content of C3 "depends" on them). In turn, the content of A1 is another formula (which depends on the contents of C1 and C2). The formula in cell A1 must be processed and the value returned by the formula computed first, so that A1 gets a numerical result. As C1 and C2 have numerical contents, no processing is needed at these cells. After that, the formula in C3 can be processed for getting its numerical value.

The program developed should compute the values of the contents of the cells as explained above.

Please also note that if, while we are working with a spreadsheet, we change the content of a cell, the program has to update the values of the contents of the cells that depend of the changed cell. In the previous example, for instance, if we change the number within C1, the program should re-compute the values returned by the formulas within A1 and C3, and in this order.

The program should efficiently perform this process (**it is not efficient to always re-compute all the spreadsheet every time the content of one cell is changed**).

**The design shall consider that users may make errors in writing formulas: the program should be able to identify syntax errors in formulas.**

If the user tries to set as content of a cell a formula that introduces circular dependencies in the spreadsheet, the program shall identify this situation and shall prevent the user to do it raising a notification. Below follows two examples of circular dependencies:

EXAMPLE 1:
A1 = A2 + A3

A2 = A1*2

Notice that the spreadsheet cannot successfully compute the values of both formulas.

EXAMPLE 2:

A1 = A2 + A3

A2 = A4 + A5

A4 =A1*2

Please note that the processing must take place just after loading a spreadsheet from a file: all non-empty cells must be processed and their values recomputed if necessary. Notice as well that, as the spreadsheet shall not contain any circular dependency, the cells may be uploaded to the spreadsheet and their values computed as they appear in the file. See section **Storage and retrieval of spreadsheets** for more details.

## Functions and ranges

The spreadsheet developed must support the following set of functions:

. SUMA(…): it computes the sum of the values of the cells identified in its argument.

. MIN(…): it returns the minimum value of the values of the cells identified in its argument.

. MAX(…): it returns the maximum value of the values of the cells identified in its argument.

. PROMEDIO(….): it returns the arithmetic mean of the values of the cells identified in its argument.

All the aforementioned functions shall have one or more arguments, separated by the character ';'. **The list of arguments of the 4 aforementioned functions may contain one or more arguments of each of the types listed below**:

- Numerical values. Example: SUMA(1;2;3).
- A range of cells. This is a "rectangular" set of cells. A range is defined by two cells identifiers separated by the character ':': the one corresponding to the cell occupying the upper left position of the rectangle, and the one corresponding to the cell occupying its bottom right position. Some examples follow below:
    - Range "A1:B3" is formed by cells A1, A2, A3, B1, B2, and B3.
    - Range "B2:E3" is formed by cells B2, B3, C2, C3, D2, D3, E2, and E3.
- Individual cells. In this case the argument is its coordinate. In essence, an individual cell is a degenerated case of range, where upper left cell and bottom right cell are actually the same cell.
- Other functions. Functions may have as arguments the result generated by invoking functions.

Below follows a complex example of a formula, which the program should be able to properly process:

=1 + A1*((SUMA(A2:B5;PROMEDIO(B6:D8);C1;27)/4)+(D6-D8))

The figure below shows the usage of function SUMA() with one range as argument:

| | A | B | C |
|---|---|---|---|
| **1** | 1 | 2 | |
| **2** | 3 | 4 | |
| **3** | =SUMA(A1:B2) | | |

The result of computing the function within cell A3 would be 10.

## Storage and retrieval of spreadsheets

Several formats have been defined for storing spreadsheets in files, among which the S2V (Semicolon Separated Values). A spreadsheet stored using this format shall be a text file. Each line in the file shall include the contents of one row of the spreadsheet. One line of the file shall contain the sequence of textual representations of the contents of the cells in the row separated by the character ';'.

As an example, consider the spreadsheet shown in the figure below

| | A | B | C |
|---|---|---|---|
| 1 | =C1+C2 *(3)* | 4 | 1 |
| 2 | | | 2 |
| 3 | | TOTAL | =A1+B1 (7) |

Note: in A1 and C3, the numbers appearing at the right-hand side of the cells enclosed in round brackets are the results obtained after computing the formulas that appear at their left hand sides.

This spreadsheet should be stored in a file whose contents shall be as indicated below:

```
=C1+C2;4;1
;;2
;TOTAL;=A1+B1
```

The first component in the first line represents the content of cell A1, which is a formula. It can be seen that this format stores the textual representation of a formula ("=C1+C2" within cell A1 in this case). After it, the textual representation of the numerical values of cells B1 and C1 appear separated by the character ';'. The last value of in the row is the content of the last non empty cell in the row.

As A2 and B2 are empty cells, no values are stored in the file for them (notice the two first ';' characters in second line). They are followed by the character '2', the textual representation of number 2, content of C2.

As A3 is empty, the first character of third line is ';'. After that, it appears "TOTAL", the string contained by cell B3. Finally, after the ';' separator appears the text corresponding to the formula contained by cell C3.

If a formula contains a function with more than one argument, then the ";" characters separating each argument shall be stored as "," in the file. When reading the file for creating the spreadsheet that it represents, the "," separating arguments in a function will be converted back into ";".

The program must be able to store the contents of a spreadsheet in the S2V format. The program must also be able to parse the contents of a file (stored in a folder of the machine running the program) whose contents are conformant to the S2V format, and create the corresponding spreadsheet.

## Interaction with user

No graphical user interface is required for this program.

For interacting with the users, the program shall present a textual menu of options. When the user selects one of them, the program shall process that option. If it requires further interactions with the user, such interactions shall occur as a sequence of questions/answers, until the program gets from the user all the information that it requires for processing the request.

If the program finds any problem that prevents to completely process the request, it shall inform to the user through a message in the console.

The user interface shall be designed in such a way that its replacement by a graphical user interface would require the minimum amount of work possible.

The set of specific options that the program shall present to the user must be defined during the analysis of the problem.

**Once all the groups will have delivered their lists of options to the professor, the professor shall impose a specific repertoire of options with a specific syntax, which all the groups must follow, in order to facilitate the review and marking of the projects.**