

Jalon 2 : les cellules, les formules et leur évaluation

Cellules Au niveau de la structure de données, une cellule est définie par les informations suivantes (type `s_cell`) :

- la chaîne de caractères que l'utilisateur a saisi lors de l'édition de la cellule,
- la valeur numérique de type `double` associée à la cellule (ou 0.0 si celle-ci n'est pas calculable),
- la liste des jetons construite à partir de l'analyse de la formule (voir ci-dessous),
- la liste des cellules qui dépendent de la cellule considérée.

Une cellule existe dans la structure de données lorsque son contenu n'est pas vide et/ou lorsqu'elle est référencée par au moins une autre cellule.

Feuille de calcul À ce point du projet, les informations qui décrivent la feuille de calcul sont :

- le nom du fichier associé pour l'enregistrement de la feuille,
- le nombre de lignes et de colonnes,
- la liste des cellules existantes.

Ces informations seront enregistrées dans une variable externe (c'est à dire globale).

Formule Après l'édition d'une cellule, le nouveau contenu est décomposé en une liste de jetons (*token*). Un jeton représente soit un nombre, soit une référence vers une autre cellule, soit un opérateur.

Un jeton est décrit par la structure de données suivante :

```
typedef struct token {
    enum {VALUE, REF, OPERATOR} type;
    union {
        double cst;
        s_cell * ref;
        void (*operator)(my_stack_t * eval);
    } value;
} s_token;
```

Le champs `cst` enregistre une constante, `ref` un pointeur vers une autre cellule (le type `s_cell` est une description de cellule), et `operator` un pointeur vers une fonction qui implante un opérateur.

L'ensemble des opérations disponibles est enregistré dans un tableau de classe externe. Une opération est définie par un pointeur sur la fonction qui l'effectue, et un nom (une chaîne de caractères) qui la désigne.

Travail à réaliser

1. Écrire la fonction qui analyse la chaîne de caractères associée à une cellule (le contenu de la cellule).
2. Écrire la fonction qui évalue une cellule unique.
3. Tester et valider le comportement de ces 2 fonctions.