

# TP « Calculatrice »

N. Delestre

## Objectif

L'objectif de ce TP est de développer un programme qui puisse calculer les valeurs d'expressions arithmétiques équilibrées. Une expression arithmétique est dite équilibrée si c'est une opération binaire tel qu les opérandes sont des nombres (entiers ou réels) ou des expressions arithmétiques équilibrés entre parenthèses. L'expression arithmétique à évaluer pourra être fournie en paramètre du programme ou bien être lu à partir de l'entrée standard. Le programme affichera le résultat de l'évaluation de l'expression arithmétique sur la sortie standard.

Par exemple :

```
$ bin/calc "2,5*((3.5*.25)+(25.-17))"
22.187500
$ echo "2,5*((3.5*.25)+(25.-17))" | bin/calc
22.187500
```

Lorsque la chaîne ne sera pas syntaxiquement correcte, le programme affiche "Erreur de syntaxe" sur la sortie d'erreur standard. Lorsque l'expression arithmétique ne sera pas sémantiquement correcte, le programme affiche "Erreur de sémantique" sur la sortie d'erreur standard.

## Analyse

Nous avons précédemment étudié une analyse descendante permettant de calculer une expression arithmétique (contenant uniquement des nombres positifs) représentée par une chaîne de caractères. La figure 1 présente cette analyse.

Pour rappel :

- la chaîne de caractères en entrée de chaque opération représente la chaîne dans sa globalité ;
- le naturel non nul en entrée de chaque opération représente l'indice de la chaîne où débute de l'analyse effectuée par l'opération ;
- le premier booléen en sortie de chaque opération permet de savoir si la chaîne en entrée représente bien syntaxiquement une expression arithmétique ;
- le deuxième booléen en sortie de certaines opérations permet de savoir si une erreur sémantique s'est produite ou pas ;
- le naturel non nul en sortie de chaque opération représente l'indice de la chaîne où se termine l'analyse effectuée par l'opération dans le cas où le premier booléen précédent vaut VRAI. Si ce dernier vaut FAUX, ce naturel non nul vaut la même valeur que celle fournie en entrée.

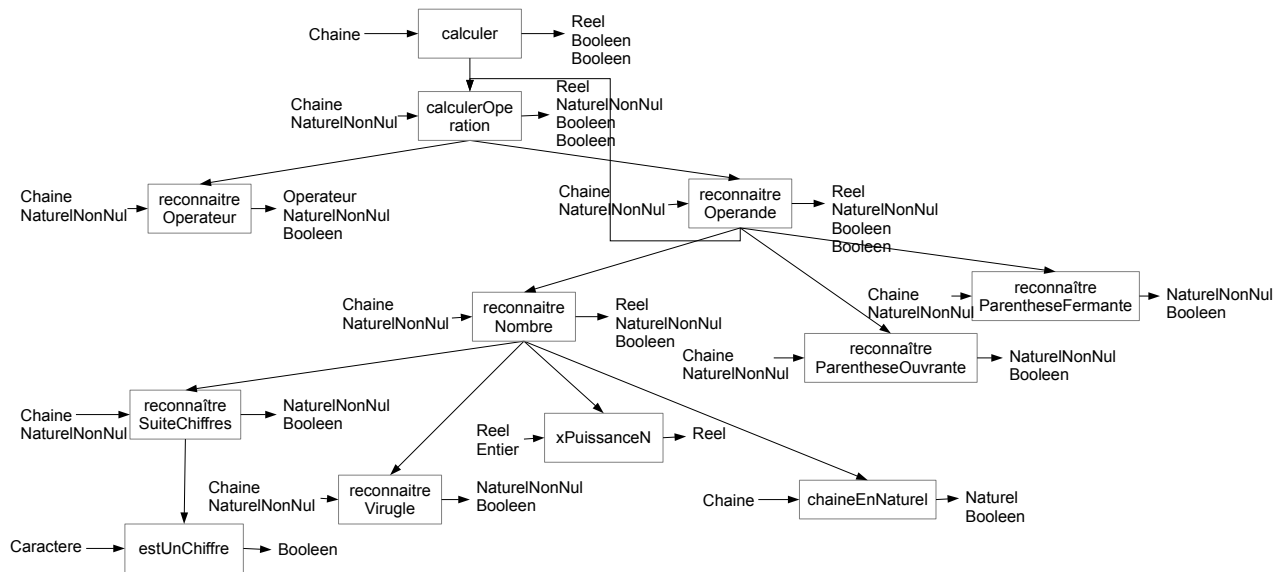


FIGURE 1 – Analyse descendante

## Conception

Les différents algorithmes des fonctions et procédures correspondant aux opérations de l'analyse descendante ont été vus en TD. Seule la procédure `reconnaitreOperande` a été complétée de façon à prendre en compte le fait qu'une opérande puisse être une expression arithmétique équilibrée. Voici son algorithme :

**procédure** `reconnaitreOperande` (**E** leTexte : **Chaine de caracteres**, debut : **Naturel** ; **S** leReel : **Reel**, prochainDebut : **NaturelNonNul**, syntaxiquementOK, semantiquementOK booleen)

**Déclaration** debutOperation, debutParentheseFermente : **NaturelNonNul**

**debut**

semantiquementOK  $\leftarrow$  VRAI

reconnaitreNombre(leTexte,debut,leReel,prochainDebut,syntaxiquementOK)

**si** non syntaxiquementOK **alors**

reconnaitreParentheseOuvrante(leTexte,debut,prochainDebut,syntaxiquementOK)

**si** syntaxiquementOK **alors**

debutOperation  $\leftarrow$  prochainDebut

calculerOperation(leTexte,debutOperation,leReel,syntaxiquementOK, semantiquementOK,prochainDebut)

**si** syntaxiquementOK et semantiquementOK **alors**

debutParentheseFermente  $\leftarrow$  prochainDebut

reconnaitreParentheseFermente(leTexte,debut,prochainDebut,syntaxiquementOK)

**si** non syntaxiquementOK **alors**

prochainDebut  $\leftarrow$  debut

**finsi**

**sinon**

prochainDebut  $\leftarrow$  debut

```

        finsi
    sinon
        prochainDebut ← debut
    finsi
fin

```

## Le programme C

Le programme est composé des fichiers suivants :

- `include/stringext.h` qui déclare la fonction `strsubstring` qui permet de récupérer la sous-chaîne d'un chaîne (à l'image des fonctions proposées par `string.h`, l'allocation de l'espace permettant de stocker la chaîne demandée est à la charge de l'utilisateur);
- `include/calc.h` qui déclare la signature de la fonction C `CALC_calculer`;
- `src/stringext.c` qui définit de la fonction `strsubstring`;
- `src/calc.c` qui définit la fonction déclarée dans `include/calc.h` et toutes celles issues de l'analyse descendante;
- `src/calcTU.c` le programme des tests unitaires des fonctions de `calc.c`;
- `src/main.c` le programme principal.

L'exécution du `make` générera le programme principal `bin/calc` et le programme des tests unitaires `test/calcTU`.

## Travail à réaliser

1. Ajouter des tests unitaires à la suite de tests "boite noire" pour vérifier que la fonction `CALC_calculer` fonctionne correctement avec (pour chaque opération) :
  - deux en entiers;
  - deux en réels;
  - un en entier et un en réel;
  - un en réel et un en entier;
  - une expression arithmétique équilibrée et un nombre entier;
  - un entier et une expression arithmétique équilibrée;
  - deux expressions arithmétiques équilibrées.
2. Développez le corps des fonctions de `calc.c` pour que les tests unitaires passent;
3. Développez le programme principal `main.c` pour que le programme `bin/calc` fonctionne comme demandé dans le sujet.