TP « Calculatrice »

N. Delestre

Objectif

L'objectif de ce TP est de développer un programme capable de calculer les valeurs d'expressions arithmétiques sans prise en compte des priorités des opérateurs : ce sont l'utilisation des parenthèses qui permet d'enlever toute ambiguïté [1]. Ainsi de telles expressions arithmétiques sont des opérations binaires où les opérandes sont des nombres (entiers ou réels) ou des expressions arithmétiques entre parenthèses [2].

L'expression arithmétique à évaluer pourra être fournie en paramètre du programme ou bien être lue 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*0.25)+(25.-17))"
22.187500
$ echo "2,5*((3.5*0.25)+(25.-17))" | bin/calc
22.187500
```

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

1. Nous verrons au semestre prochain, dans le cours de compilation, comment les gérer
2. Ceci est une définition récursive qui va nous amener à concevoir des algorithmes récursifs

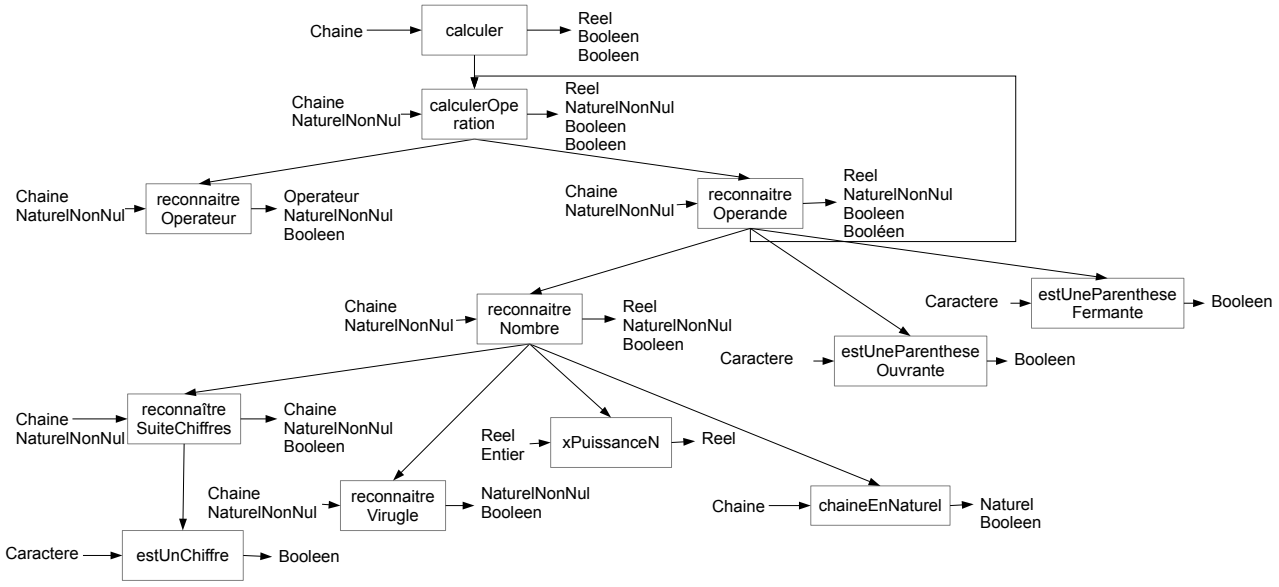

FIGURE 1 – Analyse descendante

— Le naturel non nul en sortie de chaque opération représente l'indice de la chaîne où commencera la prochaine reconnaissance dans le cas où le premier booléen vaut VRAI. Si ce dernier vaut FAUX, ce naturel non nul prend la même valeur que celle fournie en entrée.

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 afin de prendre en compte le fait qu'une opérande puisse être une expression arithmétique entre parenthèses. Voici son algorithme :

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

Déclaration debutOperation, debutParentheseFermee : **NaturelNonNul**

debut
semantiquementOK ← VRAI
reconnaitreNombre(leTexte,debut,leReel,prochainDebut,syntaxiquementOK)
si non syntaxiquementOK **alors**
reconnaitreParentheseOuvrante(leTexte,debut,prochainDebut,syntaxiquementOK)
si syntaxiquementOK **alors**
debutOperation ← prochainDebut
calculerOperation(leTexte,debutOperation,leReel,syntaxiquementOK,
semantiquementOK,prochainDebut)
si syntaxiquementOK et semantiquementOK **alors**
debutParentheseFermee ← prochainDebut

```
            reconnaitreParentheseFermee(leTexte,debutParentheseFermee,prochainDebut,syntaxiquementOK)

            si non syntaxiquementOK alors
                prochainDebut ← debut
            finsi
        sinon
            prochainDebut ← debut
        finsi
    sinon
        prochainDebut ← debut
    finsi
finsi
fin
```

Le programme C

Le programme est composé des fichiers suivants :

— `include/stringext.h` qui déclare la fonction `strsubstring` permettant de récupérer la sous-chaîne d'une 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 la fonction `strsubstring` ;

— `src/calc.c` qui définit la fonction `CALC_calculer` et toutes celles issues de l'analyse descendante ;

— `src/test_calc.c` le programme des tests unitaires des fonctions de `calc.c` ;

— `src/main.c` le programme principal.

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

Travail à réaliser

1. Expliquez chaque ligne de la fonction `strsubstring` ;
2. Ajoutez des tests unitaires à la suite de tests unitaires "boîte noire" pour vérifier que la fonction `CALC_calculer` fonctionne correctement avec (pour chaque opération) :
   — Deux entiers ;
   — Deux réels ;
   — Un entier et un réel ;
   — Un réel et un entier ;
   — Une expression arithmétique équilibrée et un nombre entier ;
   — Un entier et une expression arithmétique entre parenthèses ;
   — Deux expressions arithmétiques entre parenthèses.
3. Développez le corps des fonctions de `calc.c` pour que les tests unitaires passent ;
4. Développez le programme principal `main.c` pour que le programme `bin/calc` fonctionne comme demandé dans le sujet.