TP n°2

INTERPRETE D'EXPRESSIONS ARITHMETIQUES

A REALISER DANS HOP3X DANS LA SESSION 201X-TP2-TDA

Objectifs :

- Utilisation des TDA PILE et LISTE présentés en cours
- Utilisation de programmes écrits par d'autres sans les modifier
- Mise en œuvre d'algorithmes classiques sur les expressions arithmétiques
- Réflexion sur les structures de données adaptées à la représentation des expressions arithmétiques
- Gestion de la mémoire dynamique

Sujet :

Le but de ce TP est d'écrire un programme qui lit une expression entrée au clavier par l'utilisateur, affiche la valeur de celle-ci et attend pour lire l'expression suivante.

Une expression est :

- soit un symbole (chaîne de caractères ne commençant ni par une parenthèse ouvrante, ni par un chiffre).
- soit un nombre entier relatif
- soit une expression arithmétique en notation infixée . Exemple : 13*5+34-7. On se limitera à des calculs sur des entiers relatifs avec les opérateurs binaires +, -, *, / (division euclidienne). Les opérateurs multiplicatifs sont prioritaires par rapport aux opérateurs additifs.
- soit une opération d'affectation de la forme :
  `<symbole> = <nombre entier> | <expression arithmétique>.`

L'interprète retourne la valeur d'un symbole si celui-ci a reçu une affectation, sinon il affiche "Valeur indéfinie" ; la valeur d'un nombre entier est lui-même.
Si un symbole x contenu dans une expression arithmétique n'a pas de valeur, l'interprète affiche " x symbole indéfini", sinon, il remplace le symbole par sa valeur dans son processus d'évaluation de l'expression.

A) Dans un premier temps, on ne lit que des expressions arithmétiques sans symbole. L'interprète transforme l'expression infixée en expression postfixée, puis évalue celle-ci (Cf. TD n°2).

```
> 57 * 3 - ( 15 + 43 - 24)
= 137
>
```

B) Adapter les algorithmes précédents de façon à pouvoir interpréter des expressions contenant des symboles

```
>  ( toto + b + c * d ) * e
= toto symbole indéfini
> toto = 1
> b = 20
> c = 3
> d = 4
> e = -10
>  ( toto + b + c * d ) * e
= -330
> e = 2
>  ( toto + b + c * d ) * e
= 66
```

A préciser dans les commentaires :

- Réfléchir à la structure de données adoptée pour représenter le TDA ELEMENT (char *, union etc.) et justifier votre choix.
- Détailler la gestion de la mémoire dynamique : pour chaque objet alloué en mémoire dynamique (pile, liste, chaîne de caractères etc.) préciser qui alloue la mémoire, qui désalloue la mémoire.
- Evaluer la complexité de l'évaluation d'une expression.