Pascal Champagne
IFT2035 – Rapport TP1.

Ce rapport a pour buts d'expliquer brièvement le fonctionnement du programme résultant de la compilation de tp1.c, un programme d'interprétation d'expression postfix qui formule l'expression en entrée en d'autres formats préfix, postfix et infix ainsi que l'évaluation de l'expression pour en donner le résultat arithmétique. Il faudra aussi expliquer comment les arbres de syntaxe abstraite (ASA) sont représentés, comment se fait l'analyse syntaxique, la détection d'erreurs de syntaxe, et la construction de l'ASA, comment se fait la conversion en syntaxe C en minimisant les parenthèses, comment se fait le traitement des erreurs, et, finalement, comment se fait la récupération de la mémoire.

Fonctionnement du programme

Avant d'exécuter le programme, il doit être compilé. Nous pouvons compiler le programme sans définir de macros. L'exécution du programme se fera en se comportement par défaut, soit en respectant le format de sortie exigé par l'énoncé du TP. Voici un exemple de ligne de commande pour un comportement par défaut lors de l'exécution.

```
$ gcc tp1.c -o tp1.exe
```

GCC permet de définir des macros à partir de la ligne de commande. Pour ce faire, on peut utiliser l'option -D pour définir tp1_debug. Voici un exemple d'une commande de compilation qui définit la macro.

```
$ gcc tp1.c -D tp1_debug -o tp1.exe
```

Bien qu'optionnel, la macro tp1_debug permet à l'utilisateur du programme de voir plus d'information quand à la nature du problème rencontré lors du traitement des expressions.

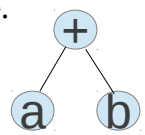
Les expressions à traiter doivent se trouver dans un fichier, à raison d'une par ligne. Le programme s'exécute en lui passant le nom du fichier contenant les expressions postfix à traîter, en paramètre. Voici un exemple d'invocation du programme à partir de la ligne de commande linux.

```
$ ./tp1.exe input.txt
```

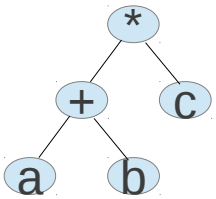
Résolution de problèmes de programmation

Comment les ASA sont représentés?

Les ASA sont représenté par des arbres binaires de façon à ce que les opérandes soient les feuilles et l'opérateur soit la racine. Par exemple, pour représenter l'expression de l'addition de a et b, nous auront un nœud interne ou la racine où nous retrouverons l'opérateur d'addition et où les opérandes seront les nœuds enfants ou des feuilles de ce nœud interne ou de cette racine.



Les nœuds enfants peuvent aussi être des opérateurs, et, donc des nœuds internes, qui auront d'autres nœuds internes ou feuilles. L'ASA de l'expression de l'addition de a et b, le résultat multiplié par c sera représenté de la façon suivante.



Les feuilles représenteront toutes des nombres et les nœuds internes et la racine, des opérateurs. Afin de permettre le traitement de l'expression, l'ordre d'évaluation se fait des feuilles vers la racine. Lorsque l'ASA est bien construit, la précédence des opérateurs sera respectée puisque les opérations à traiter plus tôt seront plus près des feuilles et celles à traiter plus tard seront plus près de la racine.

Comment se fait l'analyse syntaxique, la détection des erreurs de syntaxe et la construction de l'ASA?

Ces trois thèmes sont associés et leur traitement est assuré grâce à une pile d'expressions et à la filtration de caractères hors vocabulaire. Le vocabulaire est défini par des nombres <nombre>, soit et des opérateurs <op>. Les nombres sont définit par une série d'un ou plusieurs chiffres <chiffre>. Les <op> sont les opérateurs arithmétiques de base. Les expressions sont définies par la grammaire définie par <expr>. Les nombres sont délimités par un <op> ou par une ou plusieurs espaces. Les expressions sont composées à partir de 4 symboles opérateurs, de 10 symboles chiffre et le symbole espace pouvant délimiter les nombres ou les opérateurs formant des phrases <expr>. Notez aussi qu'un opérateur en délimite un autre, car il ne forme qu'un caractère. Si un caractère présent dans l'expression ne fait pas parti des 15 énumérés plus haut, l'expression est invalide.

La grammaire des expressions est définie soit par un nombre soit par deux expressions suivies d'un opérateur. La pile est utilisée dans la détection des erreurs de syntaxe. Lorsque le programme détecte un nombre, il ajoute une expression de type nombre dans la pile. Lorsqu'il rencontre un opérateur, il retire deux expressions de la pile, les associe à l'expression créée pour l'opérateur rencontré. Les expressions retirées sont associés de manière à ce que la première retirée soit l'opérande de droite et l'autre de gauche. Les concepts d'opérande de droite et de gauche réfèrent à la notation infix. Ils pourraient être vus comme branche de gauche et branche de droite de l'arbre. Si, lorsque le programme retire une expression de la pile et que la pile est vide, le programme considère l'expression traitée invalide. Si, lorsque le programme trouve la fin de l'expression, il reste plus d'un élément dans la pile, il considère aussi l'expression invalide. Le seul item restant, lorsque la fin de l'<expr> est détecté, représente l'ASA.

Comment se fait la conversion en syntaxe C en minimisant les parenthèses?

La conversion en syntaxe C, ou infix, se fait en traversant l'ASA. Les nœuds internes sont des expressions de type OP et les feuilles, des expressions de type NOMBRE. Le programme traverse l'ASA de façon récursive et traitera les feuilles, puis les nœuds internes en remontant jusqu'à la racine. Lorsque le programme rencontre un nœud interne, il doit vérifier la précédence des opérateurs entre le nœud interne traité et les possibles nœud internes enfants. Si nœud traité est associé à un opérateur ayant précédence sur celui du possible nœud interne enfant, des parenthèses sont ajouté à la représentation texte infix de l'enfant. Si, pour une branche droite, la précédence est égale, mais que l'opérateur est définis comme « dépendants », des parenthèses sont aussi ajoutées. Il y a deux niveaux de précédence, un pour l'addition et la soustraction et un pour la multiplication et la division. Les

opérateurs de soustraction et de division sont « dépendants ».

Comment se fait le traitement des erreurs?

Le traitement des erreurs se fait en vérifiant les codes de retour des fonctions et dans une fonction, la pile d'activation est déroulé pour retourner jusqu'à la boucle principale pour passer à la prochaine expression.

Comment se fait la récupération de l'espace mémoire?

Il y a deux façon de récupérer l'espace mémoire. La première, lorsque le traitement d'une expression s'exécute avec succès, se fait en appelant la fonction de récupération d'un expression qui traverse l'ASA de façon récursive afin de libérer les feuilles en premier en remontant jusqu'à la racine. Si le programme détecte une erreur de syntaxe, l'espace mémoire est libérer en appelant la fonction de récupération de la mémoire d'une expression pour chaque expression contenues dans la pile.