Rapport du Travail Pratique 1

(TP1)

Par

Sulliman Aïad et François Poitras

Rapport présenté à

M. Marc Feeley

Dans le cadre du cours de

Concepts des langages de programmation (IFT2035)

Remis le jeudi 22 octobre 2015

Université de Montréal

Table des matières

1 Fonctionnement du programme

Le programme est une calculatrice en notation postfixe, aussi appelée notation polonaise inversée. On peut également affecter des résultats de calculs à des variables, qui sont des lettres miniscules, de a à z. Lors de l'éxecution du programme, celui-ci attend une entrée de l'usager. Cette entrée peut être soit un simple calcul, une variable – pour voir sa valeur – ou une affectation à une variable. Pour affecter une variable, on place =<lettre> à la fin de l'expression. Il est possible d'affecter plusieurs variables simultanément en enchaînant les =<lettre>. Par exemple 1 1 + =a =b =c affectera la valeur 2 aux variable a,b et c.

En cas de problème, soit avec la syntaxe ou avec l'utilisation de variables non initialisées, l'utilisateur est informé de l'erreur qu'il a commise. Il se peut aussi que le programme ne dispose pas suffisament de mémoire pour effectuer un calcul. Si c'est le cas, l'utilisateur recoit un message et le programme attend la prochaine instruction. Ce comportement est bien évidemment une simulation, car dans un cas réel, il serait surprenant que le programme trouve subitement de la mémoire pour un autre calcul.

2 Problèmes de programmation

2.1 Représentation des nombres et variables

Le nombre en tant que tel est une liste doublement chaînée. Ainsi, chaque cellule possède un pointeur vers la prochaine cellule, un pointeur vers la cellule précédente et une valeur, qui est un chiffre.Un object nombre est constitué d'un pointeur vers la tête d'une liste doublement chaînée, d'un pointeur vers le dernier élément de la liste, d'une longueur et d'un compteur de références. La raison pour laquelle nous avons choisi une telle implémentation est pour faciliter le calcul. Nous lisons les nombres de gauche à droite, mais le calcul de chaque opération se fait de droite à gauche. Les nombres sont donc inversés en mémoire. L'unité la plus significative est à la fin de la liste et l'unité la moins significative est au début. Pour pouvoir imprimer le nombre efficacement sans parcourir deux fois la liste, nous gardons une référence sur le dernier élément et nous imprimons chaque cellule, à reculons. La propriété de longueur sert à ajuster les nombres au besoin dans les opérations. Pour plus de détails, voir la section 2.4.

2.2 Analyse de l'entrée et calcul

Nous commençons par séprarer l'entrée (si elle est valide) en différentes parties, séparées par des espaces, à l'aide de la fonction *strtok*. Tant qu'il y a des parties à lire, on effectue différentes opérations avec la pile selon le cas. Si c'est une valeur ou une variable, on pousse la valeur (ou la valeur de la variable, si elle est définie) sur la pile. L'autre possibilité est un opérateur. Dans ce cas, on dépile deux valeurs et on effectue leur calcul. Le résultat est ensuite poussé sur la pile. À la toute fin, si l'expression était bien formée, il ne reste que le résultat final sur la pile. Si l'expression contenait une affectation d'une (ou plusieurs) variables, on affecte la valeur sur le dessus de la pile à la variable (ou aux variables, si c'est le cas).

Pour imprimer le nombre, nous devons imprimer charactère par charactère, puisque nous ne savons pas *a priori* il y a combien. En partant de la fin, tant que le nombre courant a un prédécesseur, nous imprimons le nombre courant. Puis, lorsqu'il n'a plus de prédécesseur, nous imprimons le nombre courant.

2.3 Gestion de la mémoire

Dans les opérateurs, la mémoire est allouée avec *malloc* lorsque nécéssaire, c'est-à-dire lorsqu'il faut une unité supplémentaire car le calcul n'est pas fini. Aussi, à la fin des calculs, on alloue de l'espace pour le carry seulement si nécéssaire. Pour libérer un nombre, nous utilisons les fonctions *superFree* et *recursiveSuperFree*. La première appelle la seconde et la seconde s'appelle elle-même récursivement, tant qu'il y une cellule suivante à libérer. Si ce n'est pas le cas, la cellule courante est libérée et sa valeur change à NULL. Lorsque toutes les cellules on été libérées, le nombre en tant que tel est libéré.

La pile est une structure possédant un pointeur de pointeur de nombres, qui sert à stocker les valeurs calculées. une taille et une taille maximale. La taille de la pile est incrémentée à chaque valeur poussée et décrémentée à chaque valeur dépilée. Dans un souci d'efficaité de mémoire, si la pile est trop grosse, sa mémoire est réallouée dans un bloc plus petit. Cela se produit lorsque la taille maximale de la pile est au moins deux fois plus grande que la taille actuelle. De manière similaire, si il n'y a plus de place pour empiler un élément, la taille de la pile double.

2.4 Algorithmes de calcul

2.4.1 Addition

Avant de calculer quoique ce soit, nous vérifions les signes des nombres passés en paramètres. En effet, dans certains cas, comme, $a + -b$, il est plus simple de faire une soustraction. Nous commençons le processus d'addition en vérifiant que les deux nombres ont la même longueur. Cela doit être fait pour éviter qu'une des listes se aie encore des suivants, sans que l'autre en aie.On aurait alors une addition entre un chiffre et un NULL. Tout d'abord, on vérifie quel nombre, entre a et b, est le plus long. Si b est le plus long, on inverse les deux nombres. Par la suite, on rajoute des 0 à la fin de b tant qu'il n'a pas atteint la longueur de a. On peut ensuite commencer l'addition elle-même. En commençant par l'unité la moins significative, on additione chiffre par chiffre, plus le carry. Si le résultat est plus grand que 9, on soustrait 10 et on change la valeur de la variable carry à 1. La retenue ne peut pas être plus grande que 1, car on additionne au maximum 9 et 9. Si le calcul n'est pas fini, on alloue un espace pour la prochaine cellule et on ajuste les pointeurs en conséquence. La dernière étape consiste à vérifier si il ya une ultime retenue. Si c'est le cas, on fait un processus similaire a celui décrit plus haut. La somme est ensuite retournée. Cette opération s'effectue en O(n), où n est la longueur du plus grand des deux nombres.

2.4.2 Soustraction

La soustraction utilise un processus similaire à celui de l'addition dans la phase initiale. On commence par vérifier la longueur de chaque nombre et si $a < b$, on inverse les deux nombres en tenant compte du fait que $(a-b) = -(b-a)$. Ensuite, des zéros sont ajoutés à la fin de b jusqu'à ce que les longueurs soient

égales. Lors du processus de soustraction, si la différence des chiffres de deux cellules est plus petite que 0, on emprunte dans l'unité suivante et on recommence le calcul afin d'avoir une valeur de résultat positive. Le résultat final est ensuite retourné. Cette opération est également en O(n).

2.4.3 Multiplication

Pour multiplier deux nombres, nous utilisons la technique «grade-school». La multiplication ne nécéssite pas d'ajustement initial, car le calul arrête quand il n'y a plus de chiffres de b à multiplier. On commence par initialiser un tableau de nombres, pour stocker les résultats partiels. La longueur de ce tableau est connue d'avance, il s'agit de la longueur de b, car il y aura autant de résultats partiels que de chiffres dans b. Le calul de la multiplication peut commencer. Un résultat partiel est créé et les valeurs nécéssaires au calcul sont initialisées au début de la boucle. Pour chaque chiffre de a, on calcule le résultat de la multiplication avec l'élément courant de b. Dans un processus similaire à celui de l'addition, on ajuste la retenue au besoin. Dans ce cas, il est impossible de prévoir à l'avance la quantité de la retenue, ce qui nous demande d'utiliser une boucle. Tant que le résultat intermédiaire est supérieur à 9, on ajoute 1 à la retenue et on soustrait 10 du résultat intermédiaire. À la fin de cette série de calcul, il nous faut ajuster le décalage du résultat partiel. Ce décalage est d'un 0 par unité de b, commençant aux dizaines. Par exemple, s'il y a trois chiffres dans b, le premier résultat partiel n'aura pas de décalage, le deuxième aura 1 zéro de décalage et le troisième aura 2 zéros de décalage. Ces zéros sont ajoutés à la partie la moins significative du résultat partiel, c'est-à-dire au début de la liste.

Lorsque tous les résultats partiels on été calculés, nous créeons un nombre initialisé à 0. Puis, pour chaque élément du tableau de résultats partiels, fait l'addition de la somme et du i-ème élément du tableau. La mémoire allouée à chaque résultat partiel est ensuite libérée, puis le tableau est libéré. La somme est finalement retournée. La multiplication se fait en $O(n^2)$.

2.5 Traitement des erreurs

Lors du calcul en tant que tel, la seule erreur qui peut se produire est une insuffisance de mémoire. Si c'est le cas, la fonction de calcule indique un message et le programme attends la prochaine instruction. Les variables intermédiaires servant au calculs sont libérées avant l'arrêt de la fonction.

Dans l'entrée, il peut y avoir des erreurs de syntaxes, ou encore un appel à une variable qui n'est pas encore initialisée. Si l'utilisateur fait l'une ou l'autre de ces actions, il en est informé et l'interprétation de l'entrée arrête. Le programme attends la prochaine instruction. La mémoire allouée à l'entrée et aux variables stockées est libérée avant d'attendre de nouvelles instructions.