L3 SPI Programmation en C/C++ - 2020-2021 session 1

Responsable : J. Mitéran - Durée : 2 h - Documents autorisés : cours et TD/TP. Pas d'ordinateur ni de téléphone ni de calculatrice.

Commentez vos programmes – rédigez vos réponses en les justifiant.

Question A : (15 points) (répondre en langage C)

Le but du programme est de lire dans un fichier une expression mathématique qui est une succession de nombres entiers et d'opérateurs et de calculer le résultat de l'opération. Dans un premier temps (question A1), les opérations ne seront que des additions et soustractions. Dans un second temps (question A2), le traitement des multiplications sera pris en compte. La syntaxe et la présentation devront être parfaites. Vos programmes doivent fonctionner pour n'importe quelle expression qui respecte la forme proposée.

Question A-1 (7 points)

Le fichier nommé « controleA1.txt » contient par exemple l'expression suivante
7+5-3-2+28
Le résultat de l'affichage de votre programme sera 35
Chaque nombre sera lu avec fscanf avec le format %d
Chaque opérateur sera lu avec fscanf avec le format %c
La ligne est supposée se terminer par le caractère **'\n'** après le dernier nombre (ce sera donc le dernier opérateur lu).
Rappel : la fonction fscanf retourne -1 en cas d'erreur de lecture, et 1 si elle a lu une valeur.
La longueur de l'expression n'est pas connue, mais on supposera qu'elle ne peut pas contenir plus de 1000 nombres.
On suppose que l'expression ne comporte ni espace, ni erreur de typographie.
Votre programme doit donc lire l'expression jusqu'à la fin de la ligne puis afficher le résultat de l'opération.

Question A-2 (8 points)

Le fichier nommé « controleA2.txt » contient par exemple l'expression suivante
7+5*3-2+28-2*2
Il faut prendre en compte la priorité de l'opération multiplication.
Le calcul sera 7 + 15 – 2 + 28 - 4 ce qui donnera à l'affichage la valeur 44.
La position des *, + ou – est ici un exemple, votre fonction doit être valable quelle que soit la position de ces opérateurs. Il peut y avoir plusieurs opérateurs * successifs. On suppose qu'il y a toujours au maximum 1000 opérandes.
L'algorithme est le suivant :
Vous lisez l'intégralité des opérandes et des opérateurs que vous stockez dans des tableaux de nombres et d'opérateurs avant de réaliser le calcul.
Ainsi le tableau de nombres contient au départ

7	5	3	2	28	2	2

Vous précalculez les multiplications, et remplacez le premier nombre de chaque multiplication par le résultat de la multiplication

Puis après les multiplications, le tableau contient

7	15	0	2	28	4	0

Vous calculez ensuite les additions et soustractions.

Question B : (5 points) (répondre en langage C++)

On suppose que l'on dispose de la classe **VectorDouble** qui permet de gérer des tableaux des variales de type double.
La déclaration d'un vecteur MonTableau contenant des ***double*** sera par exemple
VectorDouble MonTableau ;

On suppose que cette classe possède une fonction membre permettant d'ajouter un élément au tableau. Cette fonction se nomme **push_back** et reçoit en argument l'élément à placer dans la dernière case du tableau.
Exemple :
MonTableau.push_back(3) ;
MonTableau.push_back(5) ;

Pour lire un élément de ce tableau, on suppose que l'on peut utiliser l'opérateur [ ] exactement comme pour un tableau classique.
Exemple :
int valA = MonTableau[0] ; // ici valA prendra la valeur 3.
int valB = MonTableau[1] ; // ici valB prendra la valeur 5.

Note : il est ici inutile de gérer la réservation/libération de mémoire.
La classe **VectorDouble** possède également la fonction membre ***size*** qui retourne le nombre d'éléments contenus dans le tableau (le type de la variable retournée est un int).

Question B-1 (2 points)

Ecrire un programme ***main*** qui remplit, en utilisant la classe **VectorDouble**, un tableau de ***double*** avec les valeurs de la fonction **sin(2x+0.2)**
pour x variant de 0 à pi (3.14159) par pas de 0.1.

Question B-2 (3 points)

Ecrire et utiliser une fonction qui calcule et retourne la somme des valeurs stockées dans le tableau précédent. Cette fonction devra donc recevoir en argument le tableau initialisé et rempli dans le main. Elle ne reçoit donc **qu'un seul argument**. L'affichage du résultat sera fait dans le **main**.
Remarque : Une variable de type **VectorDouble** se passe en argument comme n'importe quel autre type de variable que vous avez utilisé jusqu'à maintenant.

1/1