TP3

Calculatrice polonaise inversée

## Notation polonaise inversée

En arithmétique, la notation à laquelle nous avons tous été habitués est la suivante :

2 + 2

Simple. Mais lorsqu’on veut y aller avec une expression complexe, des problèmes émergent. Par exemple :

2 + 3 \* 4 + 5

Nous savons que nous devons multiplier 3 par 4 et ensuite additionner les trois termes. Mais que se produit-il lorsqu’on veut multiplier le ***résultat*** de 2 + 3 par le ***résultat*** de 4 + 5? On doit mettre des parenthèses.

(2 + 3) \* (4 + 5)

Il existe une notation qui évite les ambiguïtés : la notation polonaise inversée.

Prenons le cas de base, 2 + 2 :

2 2 +

Les opérandes ***précèdent*** l’opérateur.

2 + 3 \* 4 + 5 devient 2 3 4 \* + 5 +

(2 + 3) \* (4 + 5) devient 2 3 + 4 5 + \*

Lorsqu’on rencontre un opérateur, on prend les deux opérandes les plus récents et on effectue l’opération. Donc le dernier et l’avant-dernier opérande qu’on a rencontrés. Quelle structure de données avons-nous vue qui nous permet d’avoir accès aux éléments les plus récents en premier? La pile!

Par exemple :

3 2 / 5 4 + \*

On empile 3 et le 2, on arrive au /. On effectue l’opération en désempilant 2 et 3 et on empile le résultat, soit 1.5 (l’élément sur le haut de la pile est le diviseur, l’élément sous lui, le dividende). On empile 5 et 4, on arrive au +. On effectue l’opération en désempilant 4 et 5 et on empile le résultat, soit 9. On arrive au \*. On effectue l’opération en désempilant 9 et 1.5. On empile le résultat (13.5). On a terminé la lecture de l’expression. On désempile le résultat. S’il n’y pas un et un seul élément dans la pile, il y a une erreur dans l’expression.

États de la pile :

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | / |  |  |  | + |  | \* |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | 4 | 4 |  |  |  |
|  | 2 | 2 |  | 5 | 5 | 5 | 9 | 9 |  |
| 3 | 3 | 3 | 1.5 | 1.5 | 1.5 | 1.5 | 1.5 | 1.5 | 13.5 |

## Détails d’implémentation

Vous devez créer un programme multi-fichier.

calculatrice.cpp

Vous devez créer un module pour la calculatrice polonaise inversée. La pile (LIFO) devrait être ici et prendre la forme d’une liste chaînée (donc pas de tableaux). Elle doit manipuler des structs.

La fonction suivante DOIT exister avec la signature suivante :

bool evaluerExpression(const char \*expression, int taille, double \*resultat)

* La valeur de retour est un bool qui sera vrai si l’expression est valide et faux, sinon.
* char \*expression : chaîne de caractères qui contient l’expression.
* int taille : la taille de la chaîne « expression », en nombre de caractères.
* int \*resultat : pointeur vers une variable de type int qui contiendra le résultat.

L’expression arithmétique devra être parsée dans calculatrice.cpp. Certains des exercices et vus en classe et vos TPs s’avéreront extrêmement utiles. Vous devez absolument gérer (allouer ET récupérer) la mémoire. Vous devez également soulever les erreurs. On peut assumer que seuls les espaces (' ' == 32) seront utilisés par l’utilisateur pour séparer les termes de l’expression qu’il ou elle entrera. Sinon, erreur!

calculatrice.h

La seule fonction qui est exposée est:

bool evaluerExpression(const char \*expression, int taille, double \*resultat)

Il peut, par contre, s’y trouver des définitions de constantes.

main.cpp

Vous devez créer un main() qui effectue la demande d’expression à l’utilisateur. S.V.P. utiliser la fonction getligne(…) que nous avons vue en classe. Codez-la dans main.cpp. Lorsque l’utilisateur appuie sur Entrée, vous devez envoyer l’expression entière à la calculatrice et afficher le résultat, à 2 décimales près. Vous devez permettre à l’utilisateur d’entrer autant d’expressions qu’il veut, mais de quitter lorsqu’il en a envie. L’utilisateur doit séparer les éléments de son expression à l’aide d’espaces.

Il ne doit donc se trouver que main() et getligne() dans main.cpp.

## Librairies et fonctions

## Vous avez droit aux librairies suivantes :

## stdio.h

## stdlib.h

## De ces librairies, vous pouvez uniquement utiliser les fonctions suivantes :

* getchar()
* printf()
* atof()
* malloc()
* free()

Les constantes EOF et NULL sont permises.

## Remise

Vous devez remettre un fichier .zip contenant le code source et UNIQUEMENT le code source de votre calculatrice (main.cpp, calculatrice.cpp et calculatrice.h). Vous pouvez remettre le projet à tout moment d’ici le 8 décembre, à 14h24, inclusivement. Nous corrigerons ensemble durant le cours du 8 décembre, donc aucun retard ne sera permis.