

ZZ1 – C – TP n°6

Calculatrice

Première partie :

Ecrire une fonction **identification** qui reçoit en entrée une chaîne de caractères correspondant à une expression mathématique simple et teste si elle appartient au tableau de chaînes de caractères suivant : { "x", "sin(x)", "cos(x)", "log(x)", "exp(x)" }. Si la chaîne lue est trouvée, on retourne son indice dans le tableau et -1 sinon.

Ecrire une fonction **eval** qui accepte en entrée une valeur réelle double précision et un indice correspondant à une expression mathématique identifiée par la fonction précédente. Cette fonction **eval** retourne le résultat en double précision : `expression(valeur)`.

Ecrire une fonction **calcul** qui reçoit en entrée un intervalle [a, b], un pas de progression `delta` dans cet intervalle et un identificateur d'expression. Cette fonction évalue l'expression "identifiée" pour toutes les valeurs entre a et b par pas de `delta` et affiche sur la sortie standard chaque résultat d'évaluation (séparer les résultats par des espaces).

Ecrire un programme principal qui teste ces fonctions (limitez les saisies au clavier en fixant les paramètres d'appels).

Fonctions de conversion de chaînes de caractères en nombre :

<code>double atof(char *chaine);</code>	convertit chaîne en un double
<code>int atoi(char *chaine);</code>	convertit chaîne en un int
<code>long atol(char *chaine);</code>	convertit chaîne en un long int

Deuxième partie :

La chaîne lue comporte maintenant deux expressions du type défini dans la première partie séparées par un des opérateurs suivants : +, -, *, /.

Vous utiliserez la fonction `fgets` pour lire la chaîne et la fonction `sscanf` de la manière suivante pour séparer les expressions et l'opérateur :

```
sscanf (chaine,"%s %c %s", exp1, &op, exp2);
```

Réaliser un programme qui évalue l'expression lue entre a et b avec un pas de `delta`. Identifiez de vous même les fonctions élémentaires dont vous avez besoin pour limiter le travail, et celles qu'il faudra modifier pour réaliser ce programme. Limitez au maximum des erreurs (segmentation fault, erreurs de lecture, ...).

Troisième partie :

Remplacer `switch` par l'utilisation des pointeurs de fonction. Pour ceux qui le souhaitent (compliquer l'expression à souhait, ajout de constantes d'opérateurs, de nombres d'opérateurs, ...).

Déboguer

Continuer l'utilisation de déboguer.

CONSEILS :

Commenter correctement votre code, les fonctions de ce TP seront utilisées ultérieurement dans un autre TP.