Sujets de mini-projet 2016

Progammation C++

VAYSSADE Jehan-Antoine

Le rapport du TP 3 à déjà été effectuer dans le TP1 et le TP2, je ne vais donc expliquer que la partie ajoute sur l'évaluation des fonctions récursives :

Ajouter la gestion de fonctions récursive

(attention, pas rapide …)

Description de votre travail et de vos choix de programmation.

Définition de Fibonacci : f(x)=f(x-1)+f(x-2) recursive function depth test: 1/(x<3) tant que x<3 n'est pas vérifiée alors 1/0 retourne +inf qui permet de continuer l'évaluation sinon 1/1 retourne 1 qui correspond au cas terminal en général on peut définir end_value / check_depth	Test : f(1) 1 f(2) 1 f(3) 2 f(4) 3 f(5) 5 f(6) 8 f(7) 13 f(8) 21 f(9) 34

j'ai juste ajouté la détection des appels récursifs à la définition de la fonction, si la lvalue est contenue dans la rvalue (respectivement membre de gauche et membre de droite) alors c'est une fonction récursive.

Dans ce cas, une fonction permettant de tester la fin de la récursivité est demandé à l'utilisateur et peut avoir la forme last_value / (end_case_check) dans notre cas, pour Fibonacci, nous donnons 1/(x<3), donc si x<3, on ferme la récursivité et on renvoie 1.

Lors de l'appelle l'attribue public depth définis dans Function permet de tester la profondeur d'appel de la fonction récursif, de cette façon, lorsque c'est un premier appel la mémoire de la fonction est bindée dans le contexte courant, sinon une nouvelle mémoire temporaire est créée et bindeée pour stoker les valeurs des paramètres de chaque appels récursifs. Ils sont alors libérés au fur et à mesure que les appels récursifs se terminent.

Actuellement la profondeur maximum est fixée à 30 pour éviter les temps de calcul trop long et la stackoverflow.

Malheureusement, ce stockage est effectué par des std::string dans cette section ce qui prend à la foi du temps (copie) et de l'espace mémoire, il me faudrait d'avantage de temps pour re-factoriser cela. C'est particulièrement visible pour Fibonacci

> Où l'on trouve la relation O(2^n)*4*(3char)
>
> Quand n = 20, cela prend donc 12582912 octet ~12 Mo, …

sans parler de la réévaluation des tokens et de leur analyse, …

Je pense ajouter la parallélisation plus tard avec openmp lors de l'évaluation des paramètres des opérateurs