RAPPORT SUJET 2 : SYRACUSE

Le but du sujet est bien de vérifier la suite de Syracuse le plus loin possible. Pour cela nous avons procédé par étapes.

1ère étape :

Commencer par écrire l'algorithme de Syracuse. Pour aller plus loin que les 'int' classiques, nous avons décidé de prendre des 'long'. En effet, nous passons d'une plage de valeurs de 32 767 à 2 147 483 647. Puis en faisant de recherches, nous avons finalement fonctionné avec des 'unsigned long' qui ont une plage de valeurs allant de 0 à 4 294 967 295.

2e étape :

Ensuite, pour aller encore plus loin, nous avons décidé de passer en base hexadécimale. En effet, une très grande valeur peut alors s'écrire en peu de caractères. Nous réduisons donc la mémoire utilisée. Pour aller encore plus loin, nous avons stocké chaque caractère de cette chaîne hexadécimale dans un tableau de char. En effet, un 'long' de base occupe 4 octets (32 bits) alors qu'un 'char' n'utilise qu'un octet (8 bits).

3e étape :

Ecrire la fonction qui fait les calculs sur le nombre : /2 si c'est un nombre pair, *3+1 si c'est un nombre impair.

Pour cela, nous l'avons écrit de manière « classique » (la fonction Syracuse), c'est-à-dire une boucle while au sein de laquelle on applique les calculs effectués sur 'n', le nombre entré en paramètres. Nous avons également fait cette méthode de manière récursive (la fonction SyracuseRecurcif). Ces deux méthodes ne s'appliquent que pour les nombres inférieurs à 1 431 655 764.

Pour les nombres supérieurs à cette valeur, nous sommes en hexadécimal et les calculs sont faits via une autre méthode (unTour), qui fait directement les calculs sur des valeurs hexadécimales (inutile donc de repasser en décimal, on ne perd pas de mémoire).

Comment fonctionne le programme ?

Le programme est divisé en 2 parties :

1. La première partie s'occupe des nombres inférieurs à 1 431 655 764 (on part du max des 'unsigned long'-1, le tout divisé par 3 au cas où on tombe sur un impair). Le programme inscrit le nombre dans un fichier texte, afin de stocker l'ensemble des valeurs vérifiant la suite de Syracuse. ;

2. La deuxième partie est un peu plus complexe. Tout d'abord, nous avons traduit en base hexadécimale la valeur 1 431 655 764 que nous avons stocké dans un tableau. Le tableau a une taille de 20 (c'est une taille arbitraire, qui permet de stocker un très très grand nombre traduit en base hexadécimale, nous n'avons pas voulu prendre trop grand pour être sûres que tous les ordinateurs puissent le supporter). Puis, nous écrivons cette valeur dans le fichier texte. Ensuite, nous copions ce tableau dans un autre tableau. Enfin, dans la boucle while, nous écrivons la valeur hexadécimale, puis nous effectuons l'ensemble des opérations et enfin, nous ajoutons 1 à la valeur du tableau afin de continuer. La condition de la boucle

est toujours 1 donc vraie, mais on n'aura pas de boucle infinie puisqu'on vérifie dans la boucle while si un des calcule à donner une valeur plus grande que la taille du tableau donc on n'a plus d'espace dans le tableau et on revoie 0 avec l'affichage "retenu differente de 0 donc on s'arrête puisqu'on n'a plus d'espace dans le tableau", et qui stop la boucle while et le programme.

Comment utiliser le code ?

Dans chaque partie on a fait 2 exemples :

1. Partie 1 : avec les unsigned long : elle contient :
    - exemple 1 : avec i qui va jusqu'à 29 pour tester vite fait les fonctions de la partie 1.
    - exemple 2 avec i qui va jusqu'à le grand nombre
2. Partie 2 : on travaille avec les tableaux et l'hexadécimal : elle contient :
    - exemple 1 avec un tout petit tableau de 3 cases qui prend 5 secondes pour faire les calculs en utilisant les fonctions de la partie 2.
    - exemple 2 avec un tableau de 20 cases et qui commence par le max long -1 /3 en hexadécimal, mais qui prend beaucoup de temps pour calculer toutes les valeurs (On a testé pour 20 minute et il n'a même pas fait 1/5 de toutes les valeurs possibles à peu près)

Et bien sûr, pour la 2ème partie, on peut augmenter la taille du tableau par rapport à la taille de la mémoire qu'on possède, mais il ne faut pas oublier que la taille du fichier augmente avec les calculs, puisqu'on stocke les nombres qu'on a testés.

Donc pour exécuter le code veuillez dé commenter l'exemple à tester.