1 Choix de conception

Dans cette section, nous voulons partager un choix de conception que nous utilisons pour faire pointer une liste vers la/les plus grande(s) fractal(s). Il s'agit de la fonction bigList(list *list1, list *list2).

1.1 Arguments et structures utilisées

Les structures que nous utilisons dans cette fonctions sont les "list" et les "node".

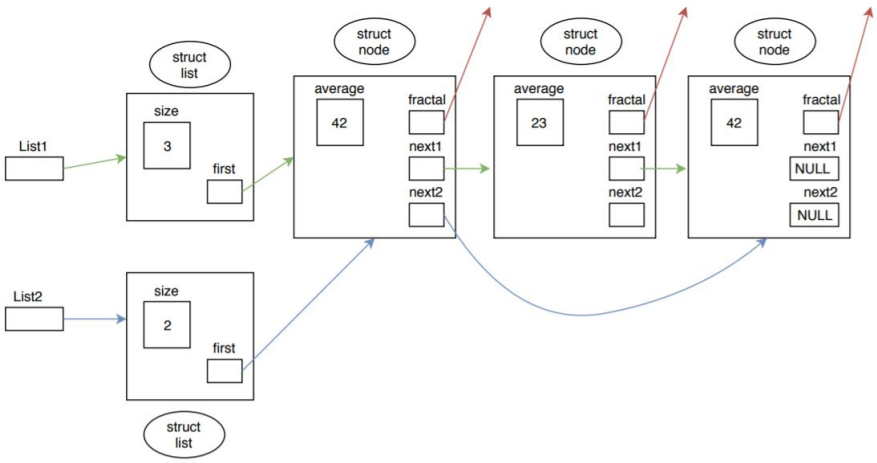
```
struct node{
    struct fractal *fract;
    struct node *next1;
    struct node *next2;
    double average;
};

struct list {
    struct node *first;
    int size;
};
```

1.2 Fonctionnement

A chaque que fois que nous allons calculer une fractale, nous allons l'ajouter à la "list1" en utilisant le pointeur "next1" de chaque "node" inséré dans la liste. Une fois que toutes les fractales sont calculées, on pourra appeler notre focntion bigList.

Le fait que l'on aie deux pointeurs "next1" "next2" dans la structure "node" est la clef de notre choix de conception. En effet, lorsque l'on appelle bigList, on va avoir la "list2" qui pointera vers un "nodes" qui lui même pointentera éventuellement vers d'autres "node"contenant des fractales avec la même moyenne par le biais du pointeur "next2". Cette technique nous permet d'avoir accès à toutes les fractales jusquà la fin du programme. Ceci facilite notre écriture dans les différents fichiers ".bmp"



2 Evalation

Malheureusement, nous ne sommes pas parvenu à afficher une fractal. En effet, à chaque fois que l'on voulait exécuter notre programme, notre code nous envoyait une erreur en nous disant qu'il n'avait pas de fractal à calculer. Nous avons longtemps cherché la solution au problème mais nous ne l'avons pas trouvée avant la deadline. Nous n'avons donc techniquement pas été aussi loin que demandé malgré les efforts fournis pour y arriver. Nous avons cependant implémenté l'ensemble des fonctions qui nous semblait nécessaire afin de réaliser la tâche demandée. Nous avons réalisé des tests en créant des fichiers d'entrée contenant des fractales avec des valeurs choisies aléatoirement. Ce sont ces tests qui nous ont permis de voir le message d'erreur dont nous avons parlé.

Nous avons plusieurs regrets en ce qui concerne la version finale de notre code. Le premier est évident et concerne le fait qu'il ne remplit pas la tâche voulue. Avec plus de temps, nous aurions sans doute pu trouver la solution à notre problème étant que nous avons mal estimé le temps nécéssaire pour le débuguage. Une deuxième deception concerne la forme de notre code. En effet, nous sommes conscient qu'il n'est pas très agréable à lire et pas forcément très bien structuré. Nous aurions bien évidemment préféré fournir un code avec une meilleure forme mais nous avons choisis de concentrer nos efforts sur le debuggage. Avec plus de temps devant nous nous aurions fait en sorte d'avoir un code mieux structuré.

En conclusion, même si notre programme n'est pas celui auquel nous esperions aboutir lorsque nous avons commencé le projet. Nous sommes magré tout content de notre implication dans ce projet que nous avons pris très aux sérieux et que nous avons tenté de réussir jusqu'au dernier moment.