TP 6 : Récursivité
Utilisation d'un débogueur

Objectifs :

- Pratiquer la programmation récursive en C
- Se familiariser avec l'utilisation d'un débogueur

Débogueur ddd

1. Eléments pour commencer

1.1. Principe et compilation

Un débogueur est un outil permettant d'exécuter un programme de manière contrôlée en observant, en particulier, l'évolution de ses variables. L'utilisation d'un débogueur n'est possible que si le code exécutable produit est enrichi avec des instructions complémentaires rendant ce contrôle possible. Le compilateur `gcc` fait cet enrichissement au moyen de l'option `-g`. La commande de compilation à utiliser lorsqu'on veut utiliser un débogueur est donc :

```
gcc -g -c prog.c -o prog.o
```

Nous utiliserons le débogueur `ddd` en exécutant : `ddd prog`

Il y a trois fonctions essentielles dans un débogueur :

- **La définition d'un point d'arrêt (breakpoint)** : pour cela on sélectionne la ligne du code où l'on souhaite que l'exécution s'arrête (soit N le numéro de cette ligne) et on pose un point d'arrêt ; la commande `run` démarre l'exécution du programme et la suspend à la ligne N.
- **L'exécution pas à pas** : à partir de la ligne N, on peut exécuter la commande `next`, auquel cas l'instruction de la ligne N sera exécutée et le débogueur passera à la prochaine instruction du programme où il suspendra son exécution en attendant une nouvelle commande. La commande `step` exécutera également l'instruction de la ligne N mais, contrairement à `next`, si N est un appel de fonction, le débogueur entrera dans cette fonction et s'arrêtera à sa première lige de code.
- **L'observation des variables** : on utilise les commandes `display` et `print` pour observer la valeur des différentes variables du programme.

Exercice

Exercice 1 : Post-fixe avec ddd

Reprendre l'exercice sur le calcul post-fixe du TP 5 et modifier votre `Makefile` de sorte à rendre l'utilisation de `ddd` possible. En utilisant cet exemple, pratiquer des exécutions contrôlées avec `ddd` et déterminer l'utilité des commandes suivantes :

`next`, `step`, `stepi`, `nexti`, `print`, `display`, `continue`, `finish`.

Récursivité

Rappel : Une fonction récursive est une fonction qui a pour propriété de s'appeler elle-même. Le principe de la récursivité est similaire à celui de la récurrence en mathématiques. On identifie un (ou plusieurs) cas de base (i.e. la définition du résultat de la fonction ne nécessite pas d'appel récursif) avant de définir la solution dans le cas général (récursif).

Exercice

Exercice 2 : Divers

1. Écrire une fonction récursive calculant le terme de rang n de la suite de Fibonacci.
2. Écrire une fonction récursive calculant le pgcd de deux entiers.
3. Écrire une fonction récursive vérifiant qu'un tableau de caractères contenant un mot de taille n donné est un palindrome (se lit de la même manière dans les deux sens ; p.ex. « eluparcettecrapule »).

Exercice 3 : Tour de Hanoi

On dispose de trois "tours" A, B et C formées d'un empilement plus ou moins grand de disques de telle sorte que chaque disque ait un diamètre inférieur à son prédécesseur. Ainsi, le plus grand disque de chaque tour se situe à leur base et le plus petit sur leur sommet. Au départ, tous les disques (supposons qu'il y en ait n) se trouvent empilés suivant les règles précédemment décrites sur la tour A. L'objectif est de déplacer tous les disques de la tour A vers la tour C en s'aidant uniquement de la tour B, tout en respectant les règles suivantes :

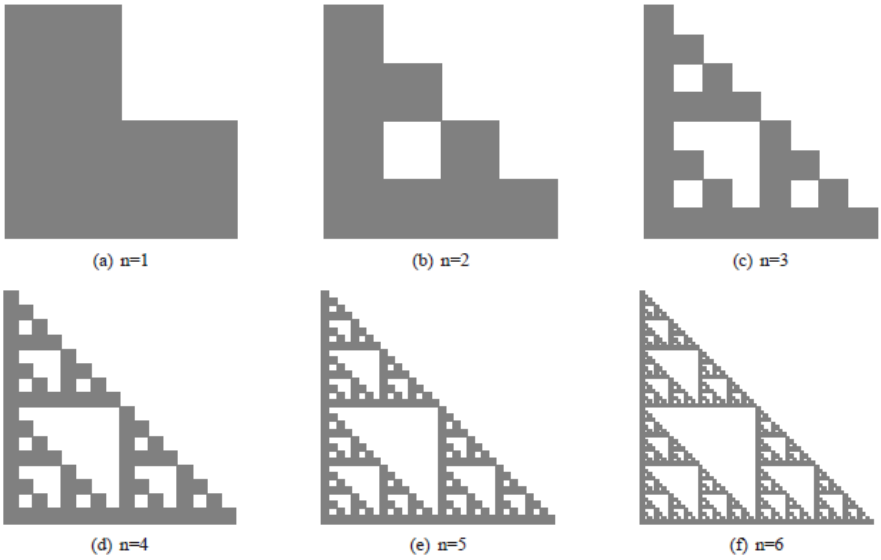
- On ne peut bouger qu'un seul disque par étape et cela doit toujours être le plus petit (celui qui se trouve au sommet).
- À chaque étape, la règle d'organisation des tours décrite précédemment doit être respectée.

a. Écrire une procédure récursive Hanoi recevant un paramètre entier (le nombre de disques initialement empilés sur la tour A) et les noms, dans l'ordre nécessaire, des trois tours ("A", "B" et "C") et affichant tous les déplacements de tours (par exemple "A -> C") jusqu'à ce que le problème soit résolu (i.e. tous les disques empilés sur la tour C).

b. (*optionnel*) En utilisant la bibliothèque graphlib proposer une version ou l'affichage des trois tours se fait de manière graphique.

Exercice 4 : Dessin d'une forme récursive (fractale)

Une figure fractale est composée de figures identiques à une échelle réduite. Un exemple de telle figure est le triangle de Sierpinski. Le triangle de Sierpinski au niveau 0 est un carré plein, d'une taille T donnée. Pour passer au niveau 1, on décompose le carré en quatre carrés de taille T/2 et on "laisse de côté" le carré en haut à droite. Pour passer au niveau 2, on répète l'opération pour les trois autres carrés et ainsi de suite (voir figure).



Ecrire une procédure récursive à un paramètre entier n dessinant le triangle de Sierpinsky de rang n.