# Programmation C
# TP 8

Récursion, tableau et memoïsation avec la suite de Syracuse...

C'est-à-dire des fonctions récursives, un petit tableau et de l'optimisation pour réduire les temps de calcul. Faire de la memoïsation consiste à stocker des résultats pour éviter de les recalculer plus tard (on parle aussi d'utilisation de mémoire cache).

La suite de Syracuse est une suite dont la valeur de départ est donnée : une entier positif $a_0$ et pour laquelle on dispose d'une règle donnant la valeur suivante à partir de la valeur précédente :

$$a_{n+1} := \begin{cases} \frac{a_n}{2} & \text{si } a_n \text{ est pair} \\ 3a_n + 1 & \text{si } a_n \text{ est impair} \end{cases}$$

Par exemple, pour la valeur de départ 12, la suite de Syracuse se déroule comme il suit :
12 - 6 - 3 - 10 - 5 - 16 - 8 - 4 - 2 - 1 - 4 - 2 - 1 - 4 - 2 - 1 ...

Posé il y a environs 2000 ans, le problème consistant à montrer que cette suite se termine systématiquement par la séquence bouclante 4 - 2 - 1 est toujours une question ouverte. L'outil informatique conforte la communauté mathématique que cette conjecture est vraie.

Comme il est peu probable que vous trouviez un contre exemple à la conjecture de Syracuse, nous allons supposer que quelle que soit la valeur de départ, la suite va arriver sur la valeur 1. Cela permet de définir la longueur de vol. La longueur de vol est le nombre d'étapes nécessaires pour arriver sur la valeur 1.

Pour 12, on a:
12 -> 0 étape
6 -> 1 étape
3 -> 2 étapes
10 -> 3 étapes
5 -> 4 étapes
16 -> 5 étapes
8 -> 6 étapes
4 -> 7 étapes
2 -> 8 étapes
1 -> 9 étapes

La longueur de vol de 12 est donc 9. La longueur de vol de 1 est 0, celle de 2 est 1, etc...

**L'objectif de ce TP est de calculer le maximum de la longueur de vol pour tous les entiers inférieurs à 200 000 000.**

Première phase d'attaque : brute force

Brute force, ça veut dire tout calculer violemment avant de faire le tri. Voici quelques conseils d'amis (je dis ça, je dis rien mais ça peut être utile pour y arriver...)

- Utilisez des entiers longs non signés. En effet, certains nombres plus petits que 200 millions dépassent la capacité des entiers C (4,2 milliards environs (Cours 2)) avant d'atteindre 1.
- Faire une fonction qui affiche la suite de Syracuse pour une valeur donnée, avant de foncer dans le mur, vous pouvez vérifier votre code avec le nombre 27.

```
nborie@perceval:$ ./test 27
27 82 41 124 62 31 94 47 142 71 214 107 322 161 484 242 121 364 182
91 274 137 412 206 103 310 155 466 233 700 350 175 526 263 790 395
1186 593 1780 890 445 1336 668 334 167 502 251 754 377 1132 566 283
850 425 1276 638 319 958 479 1438 719 2158 1079 3238 1619 4858 2429
7288 3644 1822 911 2734 1367 4102 2051 6154 3077 9232 4616 2308 1154
577 1732 866 433 1300 650 325 976 488 244 122 61 184 92 46 23 70 35
106 53 160 80 40 20 10 5 16 8 4 2 1
Fly length : 111
```

- Faire une fonction qui calcule la longueur de vol pour un entier.
- Avec une boucle `for` et une variable, retrouver le maximum pour les valeurs parcourues par votre boucle.

La mémoïsation commence par la remarque suivante : quand on calcule la longueur de vol pour 27 (qui vaut 111) et quand on voit le parcours, on peut en déduire la longueur de vol pour 82 (qui sera donc $111 - 1 = 110$), de 41 (qui vaudra 109), de 124, etc... Pourquoi les recalculer plus tard alors qu'on a déjà fait le boulot ? Mettons en cache ces observations.

Pour cela, utilisez un tableau d'entiers de taille votre objectif (200 millions ou moins). Initialisez les entrées de votre tableau à -1. Ensuite, faire une fonction récursive qui calcule la longueur de vol pour un entier en cherchant dans le tableau si c'est déjà fait, sinon calcule récursivement la longueur de vol et met en cache si possible les résultats calculés.

En utilisant la fonction `time` de votre terminal, mesurez les différentes approches. Surveillez aussi la mémoire (avec le moniteur système ou bien `top`).

```
nborie@perceval:$ time ./test
MAX_FLY : 953

real 0m9.151s
user 0m9.007s
sys 0m0.132s
```

Si vous vous ennuyez, faites des graphiques... (avec gnuplot par exemple)