Programmation : TP 4

Juliusz Chroboczek

3 octobre 2023

Boucles indéfinies

Exercice 1.

1. Écrivez un programme qui affiche « `J'adore ce TP !` » indéfiniment (jusqu'à ce qu'il n'y ait plus de courant).
2. Un appel à `time(NULL)` retourne l'heure courante, mesurée en secondes depuis le premier janvier 1970. Modifiez votre programme pour qu'il affiche « J'adore ce TP ! » pendant 10 secondes puis termine. Que se passe-t-il si on ajuste l'horloge de l'ordinateur pendant qu'il s'exécute ? Ce programme est-il adapté à un ordinateur alimenté par batterie (par exemple un *smartphone*) ?

Exercice 2.

1. Écrivez un programme qui lit des entiers au clavier jusqu'à ce que l'utilisateur tape un 0, puis affiche leur somme.
2. Modifiez votre programme pour qu'il affiche la somme et le produit des entiers rentrés. (Attention, la liste est toujours terminée par 0, pas par 1.)

Exercice 3. Écrivez un programme qui lit un entier au clavier et affiche le nombre de chiffres de cet entier en représentation canonique en base 10 (la représentation usuelle, sans zéros au début). Votre programme ne devra pas utiliser d'autres types que `int`.

Exercice 4 (Suite de Syracuse)**.** Pour chaque entier m, on définit la m-ième suite de Syracuse $(S^{(m)})$ par

$$\begin{aligned} S_0^{(m)} &= m \\ S_n^{(m)} &= S_{n-1}^{(m)}/2 && \text{si } S_{n-1}^{(m)} \text{ est pair} \\ S_n^{(m)} &= 3S_{n-1}^{(m)} + 1 && \text{sinon.} \end{aligned}$$

1. Écrivez (à la main) les premiers termes de $(S^{(6)})$ jusqu'à atteindre la valeur 1. Même question pour $(S^{(11)})$. (Ne le faites pas pour $(S^{(27)})$.)

La *Conjecture de Collatz* dit que pour tout entier m, $(S^{(m)})$ atteint 1. On appelle *temps de vol* de m le plus petit entier n tel que $S_n^{(m)}$ vaut 1. La Conjecture de Collatz dit donc que le temps de vol de tout entier est fini.

2. Écrivez un programme qui lit un entier m puis affiche le temps de vol de m. Pourquoi vous ai-je demandé de ne pas calculer $(S^{(27)})$?
3. Écrivez un programme qui vérifie la Conjecture de Collatz pour les entiers compris entre 1 et 100.

Exercice 5 (Le nombre juste)**.** Écrivez un programme qui choisit aléatoirement [1] un nombre s compris entre 1 et 100 (au sens large), et demande à l'utilisateur de le deviner. À chaque fois que l'utilisateur entre un nombre n,

- si $n < s$, votre programme affichera « Trop petit ! » et recommencera ;
- si $n > s$, votre programme affichera « Trop grand ! » et recommencera ; enfin,
- si $n = s$, votre programme affichera « Gagné ! » suivi du nombre de coups que l'utilisateur a mis pour deviner le nombre, et terminera.

Exercice 6 (Dichotomie)**.** Écrivez un programme qui devine un nombre entre 1 et 100 choisi par l'utilisateur. Votre programme utilisera 3 valeurs p (petit), m (moyen) et g (grand). À chaque étape, votre programme devinera l'entier m et demandera à l'utilisateur s'il est juste ; l'utilisateur devra entrer -1 (trop petit), 0 (juste) ou $+1$ (trop grand).

Initialement, p vaudra 1 et g vaudra 100. À chaque étape, votre programme fera $m \coloneqq \lfloor (p+g)/2 \rfloor$, et devinera m. Si m est juste, votre programme a gagné, et il termine. Si m est trop petit, votre programme fera $p \coloneqq m + 1$; sinon, $g \coloneqq m - 1$. Votre programme terminera en râlant lorsque $p = g$.

Interruptions de boucle

Exercice 7. Écrivez une version plus efficace du programme `premier.c` du TP précédent qui arrête l'itération dès lors qu'un diviseur a été trouvé.

Exercice 8.

1. La *moyenne harmonique* H d'une famille de nombres $(a_i)_{1 \leq i \leq n}$ est définie par

$$H = 0 \qquad \text{s'il existe } k \text{ tel que } a_k = 0$$
$$H = n / \sum_{i=1}^{n} \frac{1}{a_i} \qquad \text{sinon}$$

   Écrivez un programme qui lit un entier n, puis qui lit n nombres à virgule flottante et affiche leur moyenne harmonique. (Il faudra lire n nombres même si l'un d'entre eux est nul.)
2. La *moyenne logarithmique* L d'une famille de nombres $(a_i)_{1 \leq i \leq n}$ strictements positifs est définie par

$$L = e^{\left(\sum_{i=1}^{n} \log a_i\right)/n}$$

   Écrivez un programme qui lit un entier n, puis qui lit n nombres strictement positifs et affiche leur moyenne logarithmique. Votre programme devra afficher un message d'erreur compréhensible et s'arrêter dès lors qu'il aura lu un nombre négatif ou nul.

1. Faites d'abord `srand(time(NULL))` pour initialiser le générateur de nombres pseudo-aléatoires, puis `rand()` pour générer un entier pseudo-aléatoire.