Langage C : TP 2

Juliusz Chroboczek

25 janvier 2026

Boucles définies

Exercice 1 (Première boucle)**.** Écrivez un programme `carres.c` qui affiche les carrés des 10 premiers entiers naturels, c'est-à-dire la suite d'entiers $1, 4, 9 \ldots 100$.

Exercice 2 (Accumulation)**.** Écrivez un programme `somme-cubes.c` qui demande à l'utilisateur un entier n puis qui affiche la somme des cubes des n premiers nombres entiers. Par exemple, si l'utilisateur entre 5, votre programme devra afficher 225, car

$$\begin{aligned}\sum_{k=1}^{5} k^3 &= 1^3 + 2^3 + 3^3 + 4^3 + 5^3 \\ &= 1 + 8 + 27 + 64 + 125 \\ &= 225\end{aligned}$$

Boucles imbriquées.

Exercice 3. Écrivez un programme `carre1.c` qui demande à l'utilisateur un entier n et affiche un carré plein de « `*` » de côté n.

```
*****
*****
*****
*****
*****
```

Exercice 4. Écrivez un programme `carre2.c` qui demande à l'utilisateur un entier n et affiche un carré creux de « `*` » de côté n.

```
*****
*   *
*   *
*   *
*****
```

Exercice 5. Écrivez un programme `triangle1.c` qui demande à l'utilisateur un entier n et affiche un triangle plein de « `*` » de largeur n avec la pointe en haut à gauche.

```
*
**
***
****
*****
```

Exercice 6. Écrivez un programme `triangle2.c` qui demande à l'utilisateur un entier n et affiche un triangle plein de « `*` » de largeur n avec la pointe en bas à droite.

```
*****
 ****
  ***
   **
    *
```

Fonctions

Exercice 7.

1. Écrivez une fonction de prototype

   ```
   int max(int x, int y);
   ```

   qui retourne le plus grand de deux entiers. Écrivez un programme (une fonction `main`) pour tester votre fonction.
2. En utilisant la fonction écrite ci-dessus, écrivez une fonction

   ```
   int max3(int x, int y, int z);
   ```

   qui retourne le plus grand de trois entiers. Comme toujours, écrivez un programme pour tester votre fonction.

Exercice 8. Écrivez une fonction de prototype

```
int premier(int n);
```

qui retourne 1 si l'entier naturel `n` est un nombre premier, 0 sinon (attention, 1 n'est pas premier). Écrivez un programme qui lit un entier m au clavier et affiche le nombre de nombres premiers compris entre 1 et m.

Boucles indéfinies

Exercice 9. Écrivez une fonction de prototype

```
int chiffres(int n);
```

qui prend en paramètre un entier naturel et retourne 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 de nombres à virgule flottante. Vérifiez que votre programme produit le bon résultat pour 0 (qui s'écrit avec un chiffre, pas zéro).

Exercice 10.

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 11 (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 une fonction itérative (qui ne fait pas d'appels récursifs) de prototype

```
int syracuse(int n);
```

qui retourne le temps de vol de la n-ième suite de Syracuse. Écrivez un programme qui affiche les temps de vol des entiers de 1 à 100. Pourquoi vous ai-je dit de ne pas calculer à la main $S^{(27)}$?

3. Écrivez un programme qui lit un entier k puis affiche un entier n tel que le temps de vol de n soit maximal parmi les temps de vol des entiers de 1 à k.

Exercice 12 (Suite de Fibonacci)**.** Leonardo di Pisa, dit Fibonacci, élève des lapins. Les lapins de Fibonacci se comportent de façon un peu particulière : sa première année, un lapin est immature, et ne se reproduit donc pas. À partir de sa deuxième année, un lapin produit chaque année un nouveau lapin par parthénogénèse. Il y a donc un lapin l'année zéro, un lapin la première année, deux lapins la deuxième année, trois lapins la troisième année, etc.

La *suite de Fibonacci* (f_n) est définie par la récurrence suivante :

$$\begin{aligned} f_0 &= 1; \\ f_1 &= 1; \\ f_n &= f_{n-2} + f_{n-1} && \text{pour } n \geq 2. \end{aligned}$$

1. Écrivez une fonction de prototype

```
long long int fib_r(int n);
```

qui calcule f_n en faisant une suite d'appels récursifs. Écrivez un programme pour tester votre fonction (vous pouvez afficher une valeur de type « `long long` » à l'aide du descripteur de format « `%lld` »).

2. Ajoutez un appel à `printf` au corps de la fonction `fib_r` et comptez combien de fois elle est appelée lors d'un appel à `fib_r(3)`, `fib_r(4)`, `fib_r(5)`. Que constatez-vous ?
3. Écrivez une fonction

```
long long int fib_i(int n);
```

qui calcule f_n itérativement en espace constant (sans utiliser de tableaux ou faire d'appels récursifs). Écrivez un programme pour tester votre fonction.

4. Enlevez l'appel à `printf` ajouté à la fonction `fib_r` puis recompilez vos programmes à l'aide de la commande « `gcc -Wall -O2` ». À l'aide de la commande `time` du *shell* comparez le temps d'exécution des deux programmes (utilisez une redirection du *shell* pour éviter de mesurer le temps nécessaire pour taper l'entrée). Que constatez-vous ? Est-ce normal ?

S'il vous reste du temps

Exercice 13 (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 ; et
- si $n = s$, votre programme affichera « Gagné ! » suivi du nombre de coups que l'utilisateur a mis pour deviner le nombre, et terminera.

Exercice 14 (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 $p = g$.

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