Langage C
TP 4

Juliusz Chroboczek

12 février 2026

Exercice 1 (Fractions)**.** Une *fraction* est une paire d'entiers (n, d) $(d \neq 0)$ qui représente le rationnel n/d. On dit que deux fractions sont *équivalentes* si elles représentent le même rationnel. On dit qu'une fraction (n, d) est sous *forme canonique* si $d > 0$ et $\text{pgcd}(n, d) = 1$. Dans cet exercice, nous représenterons les fractions par des structures :

```
struct fract {
    long long int num, dem;
}
```

1. Écrivez une fonction

   ```
   void print(struct fract x)
   ```

   qui affiche une fraction.
2. Écrivez une fonction

   ```
   int pgcd(long long int a, long long int b);
   ```

   qui retourne le PGCD de deux entiers en utilisant l'algorithme d'Euclide. Testez votre fonction.
3. Écrivez une fonction

   ```
   struct fract canonique(struct fract x);
   ```

   qui retourne une fraction canonique équivalente à `x`. Testez votre fonction.
4. Écrivez une fonction

   ```
   struct fract add(struct fract x, struct fract y);
   ```

   qui ajoute deux fractions et retourne un résultat sous forme canonique. Testez votre fonction.
5. Écrivez une fonction

   ```
   struct fract div(struct fract x, struct fract y);
   ```

   qui divise x par y et retourne un résultat sous forme canonique.

6. Pour tout rationnel a, on définit la fonction

$$f^{(a)}(x) = \frac{1}{2}(x + \frac{a}{x}).$$

   Pour quelles valeurs de a la fonction $f^{(a)}$ admet-elle un point fixe dans **Q** ? Dans **R** ?
7. Pour tout rationnel a, on définit la suite $(x_n^{(a)})$:

$$\begin{aligned} x_0^{(a)} &= a \\ x_n^{(a)} &= f^{(a)}(x_{n-1}) \qquad\qquad (n > 0) \end{aligned}$$

   Écrivez un programme qui lit une fraction représentant a puis affiche la liste des 20 premiers termes de $(x^{(a)})$ sous forme de fractions canoniques ainsi que leurs approximations en virgule flottante [1]. Pourquoi est-ce que des nombres négatifs apparaissent ?

Exercice 2 (Tableaux)**.**

1. Écrivez une fonction `int max(int *a, int n)` qui renvoie le plus grand élément d'un tableau d'entiers. Écrivez un programme qui lit un entier n, puis qui lit une suite de n entiers qu'il stocke dans un tableau de taille n, et qui affiche la valeur de `max`.
   On aurait pu calculer le maximum à la volée, sans stocker les valeurs dans un tableau. Quels sont les avantages et les désavantages de chaque technique ?
2. Écrivez une fonction `int max_pos(int *a, int n)` qui renvoie un *indice* du plus grand élément du tableau d'entiers passé en paramètre. Écrivez un programme pour tester votre fonction. Que fait votre fonction si le tableau contient plusieurs occurrences du plus grand élément ?

Exercice 3 (Crible d'Ératosthène)**.**

1. Écrivez une fonction `int prime(int n)` qui retourne 1 si n est premier, 0 sinon (vous pouvez faire un copier-coller du TP 2). Écrivez un programme qui lit un entier n et affiche le nombre de nombres premiers entre 2 et n (au sens large).

Ératosthène écrit les entiers de 2 à 100 dans la cour de la Bibliothèque d'Alexandrie (voir figure 1). Il voit que la première valeur est 2, il envoie un esclave barrer tous les multiples non-triviaux de 2 (4, 6, 8 etc.). Il voit que la valeur suivante qui reste est 3, il envoie un esclave barrer tous les multiples non-triviaux de 3. Il voit que valeur suivante qui reste est 5, il envoie un esclave barrer tous les multiples non-triviaux de 5, et ainsi de suite. Les entiers qui restent à la fin sont exactement les nombres premiers compris entre 2 et 100.

2. Écrivez un programme qui lit un entier n puis crée un tableau d'entiers de taille $n + 1$ dont il initialise toutes les cases à 1. Ensuite, pour chaque entier i allant de 2 à n,
   - si la case d'indice i vaut 0, il n'y a rien à faire ;
   - si la case d'indice i vaut 1, alors on met à 0 les cases d'indice $2i, 3i, \ldots$ sans faire aucune multiplication (uniquement des additions).

1. On appelle cet algorithme la *méthode babylonienne* de calcul des racines carrées, et c'est une application de la méthode de Newton-Raphson.

2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	3		5		7		9		11		13		15
2	**3**		5		7				11		13		

FIGURE 1 — Le crible d'Eratosthène

Votre programme affichera ensuite le nombre de cases d'indices 2 à n qui valent 1.

Vérifiez que le crible d'Eratosthène produit les mêmes résultats que l'algorithme naïf. À l'aide de la commande `time` du *shell*, comparez le temps d'exécution de vos deux programmes. Qu'en est-il de la quantité de mémoire utilisée ?

Exercice 4 (Schéma de Horner)**.** Une *fonction polynomiale* est une fonction qui a la forme

$$f(x) = \sum_{i=0}^{n-1} a_i \, x^i$$

où les *coefficients* a_i sont des valeurs arbitraires. Par exemple, la fonction

$$g(x) = 14x^2 + 13x + 32$$

est une fonction polynomiale où $a_0 = 32$, $a_1 = 13$ et $a_2 = 14$.

Dans cet exercice, nous représenterons une fonction polynomiale par le tableau de ses coefficients. Par exemple, la fonction g ci-dessus sera représentée par le tableau `{32, 13, 14}`.

1. Écrivez une fonction `eval(double x, double *a, int n)` qui calcule la valeur de la fonction polynomiale représentée par `a` au point `x` ; vous n'êtes pas autorisés à utiliser la fonction `pow` de la bibliothèque standard. Combien votre fonction fait-elle de multiplications ? (Pensez à réutiliser la valeur de x^k pour calculer x^{k+1}.)
2. Remarquez que

   $$a_0 + a_1x + a_2x^2 + a_3x^3 + \cdots + a_{n-1}x^{n-1} = a_0 + x(a_1 + x(a_2 + x(a_3 + x(\cdots + xa_{n-1} \cdots)))).$$

   Déduisez-en un algorithme qui évalue un polynôme en faisant $n - 1$ multiplications et implémentez-le. Vérifiez que votre programme calcule les mêmes valeurs que le programme écrit à la question précédente.

Exercice 5 (S'il vous reste du temps)**.** Un étudiant place 1 zł dans une banque. Cette somme sera rémunérée au taux annuel de 100%, l'étudiant se retrouvera donc en possession de 2 zł au bout d'une année. Un deuxième étudiant choisit de placer son złoty dans une banque lui offrant un taux de rémunération de 50% tous les six mois. Ce dernier se retrouvera en possession de 2, 25 zł à la fin de l'année.

Écrivez une fonction qui calcule la somme e_n que possède au bout d'une année l'étudiant n, qui à placé son złoty dans une banque lui offrant un taux de rémunération de $1/n$ toutes les $1/n$ années. Écrivez un programme qui affiche les valeurs de e_n pour n allant de 1 à 100. Devinez la limite de la suite (e_n) ; sa convergence vous semble-t-elle rapide ?