Programmation en C et structures de données

guillaume.revy@univ-perp.fr

Tableaux et chaîne de caractères

Exercice 1. Programmes simples sur les tableaux d'entiers

- ► **1.** Écrire un programme qui lit puis affiche le contenu d'un tableau d'entiers de taille 20.
- ► **2.** Modifier le programme pour rechercher et afficher l'indice de la valeur minimum du tableau d'entiers. Faire de même pour la valeur maximum.
- ► **3.** Modifier le programme pour lire une valeur entière n, et rechercher le nombre de valeurs du tableau plus petites que n.
- ► **4.** Écrire un programme qui lit le contenu d'un tableau d'entiers et une valeur k, et qui effectue une rotation des éléments du tableau de k rangs vers la droite.

Exercice 2. Permutation

Soit t un tableau de taille n. Le tableau t est une permutation de taille n si

$$\forall\, e \in \{0, \cdots, n-1\},\ \exists\, i \in \{0, \cdots, n-1\} \mid t[i] = e.$$

- ► **1.** Écrire un programme qui lit un tableau de taille $n = 10$, puis détermine et affiche si le tableau t est une permutation de taille n.

Exercice 3. Programmes sur les chaînes de caractères

- ► **1.** Écrire un programme qui lit une chaîne de caractères, puis calcule et affiche la longueur de cette chaîne de caractères.
- ► **2.** Modifier le programme pour afficher le nombre de chiffres et le nombre de lettres de {'a',$\cdots$,'z','A',$\cdots$,'Z'} qu'elle contient.
- ► **3.** Écrire un programme qui lit une chaîne de caractères, puis qui la renverse avant de l'afficher à nouveau.
- ► **4.** Écrire un programme qui lit une chaîne de caractères et un caractère , puis qui compte et affiche le nombre d'occurrences de ce caractère dans la chaîne de caractères.
- ► **5.** Écrire un programme qui lit une chaîne de caractères et un caractère, puis qui recherche et affiche l'indice de la première occurrence du caractère dans la chaîne de caractères (et -1 si le caractère n'existe pas).
- ► **6.** Écrire un programme qui lit deux chaînes de caractères s1 et s2, puis qui compare ses deux chaînes de caractères dans l'ordre lexicographique. Le résultat sera
  - — positif ($\rightsquigarrow = 1$) si s1 $>$ s2,
  - — négatif ($\rightsquigarrow = -1$) si s1 $<$ s2,
  - — et nul ($\rightsquigarrow = 0$) si s1 $=$ s2.

En langage C, un caractère (**`char`**) peut être manipulé comme un entier.

- ► **7.** Écrire un programme qui lit une chaîne de caractères en minuscule, puis qui la convertit en majuscule, avant de l'afficher à nouveau.

Exercice 4. Fréquence d'apparition

- ► **1.** Écrire un programme qui lit une chaîne de caractères contenant que des lettres minuscules, et qui détermine la fréquence d'apparition de chaque caractère dans cette chaîne (stockées dans un tableau de taille appropriée).
- ► **2.** Modifier le programme pour afficher les caractères qui apparaît le plus et le moins dans cette chaîne.

Exercice 5. Chiffrement de César (ou chiffrement par décalage)

En cryptographie, le chiffrement de César est une méthode de chiffrement très simple, par laquelle le texte chiffré s'obtient en remplaçant chaque lettre du texte initial par une lettre à distance k fixe, toujours du même côté, dans l'ordre de l'alphabet, et de manière circulaire (c'est-à-dire, décaler 'z' de $k = 1$ rang vers la droite renvoie le caractère 'a'). Cette méthode a notamment été utilisée par Jules César dans ses correspondances secrètes (d'où son nom).

► **1.** Écrire un programme qui lit une chaîne de caractères et la valeur de k, puis qui affiche la chaîne de caractères chiffrée.

► **2.** Écrire le programme inverse, c'est-à-dire, qui lit une chaîne de caractères et la valeur de k, puis qui affiche la chaîne de caractères originale.

Exercice 6. Multiplication de matrice 2×2

► **1.** Écrire un programme qui lit le contenu de deux matrices 2×2, puis qui calcule et affiche le résultat du produit de ces deux matrices.

Exercice 7. Encore une manière de calculer le nombre de Fibonacci

Soit un entier $n \geq 2$ donné. Le nombre de Fibonacci est défini de la manière suivante :

$$\text{Fibo}(n) = \text{Fibo}(n-1) + \text{Fibo}(n-2), \quad \text{avec} \quad \text{Fibo}(0) = 0 \quad \text{et} \quad \text{Fibo}(1) = 1.$$

► **1.** Rappeler la version itérative, c'est-à-dire, qui lit une valeur entière n, puis qui calcule de manière itérative Fibo(n) définit précédemment.

L'objectif de cet exercice est de calculer le nombre de Fibonnacci Fibo(n), en utilisant une multiplication de matrice 2×2. Pour ce faire, on considère la matrice 2×2 suivante :

$$F = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} \text{Fibo}(2) & \text{Fibo}(1) \\ \text{Fibo}(1) & \text{Fibo}(0) \end{pmatrix}.$$

► **2.** Sur papier, calculer le produit matriciel $F \times F$. À quoi correspond chaque élément de la matrice résultat.

► **3.** Proposer un programme qui lit une valeur entière n, puis qui calcule le nombre de Fibonnacci Fibo(n) de manière itérative par multiplications matricielles 2×2 successives par F.

Le produit de matrice étant associatif, on souhaite améliorer la version itérative précédente pour les valeurs entières $n = 2^p + 1$ (avec $p \geq 1$).

► **4.** Sur papier, représenter le calcul du nombre Fibo(9) sous la forme d'un schéma.

► **5.** Déduire un programme qui lit une valeur entière p, puis qui calcule le nombre de Fibonnacci Fibo(n), avec $n = 2^p + 1$, de manière itérative par multiplications matricielles 2×2, en utilisant au mieux l'associativité du produit matriciel.

Exercice 8. Le triangle de Pascal

Le triangle de Pascal est de la forme suivante.

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1 ...
```

Il permet une présentation des coefficients binomiaux dans un triangle. On peut représenter les n premières lignes sous forme d'un tableau 2D, où chaque case est initialisée à 0. Ensuite

— $t[i][0] = t[i][i] = 1$ pour toute valeur $i \in \{0, \cdots, n-1\}$,
— $t[i][j] = t[i-1][j-1] + t[i-1][j]$ pour toutes valeurs $i \in \{1, \cdots, n-1\}$ et $j \in \{1, \cdots, i-1\}$.

► **1.** Écrire un programme qui lit une valeur entière $n > 0$, puis qui construit et affiche le tableau définit précedemment, en utilisant un tableau à taille variable.