TP2: Algorithmes récursifs et tableaux

Préambule

L'objectif de cette séance est de se familiariser avec différents algorithmes :

- pour des tâches élémentaires sur des tableaux ;
- utilisant une approche récursive ;
- pour manipuler des tableaux et chaînes de caractères .

Quelques informations importantes :

- Le langage de programmation à utiliser est le C.
- Pour chaque exercice, écrivez toujours des tests afin de vérifier que le code est correct.
- Le nombre d'astérisques (*) indique le niveau de difficulté relatif, les exercices marqués (P) doivent être effectués sur PLaTon.
- Des outils pour tester les fonctions sont fournis dans le fichier zip `algo-tp02-code.zip`.

1 Tâches élémentaires sur des tableaux

Exercice 1: Affichage d'un tableau (*). Écrivez une fonction qui affiche les n premiers éléments d'un tableau de réels.

Prototype: `void affiche_tab(double t[], int n);`

Exercice 2: Mise à zéro d'un tableau (*). Écrivez une fonction qui remplit un tableau de n réels par des zéros (`0.0`).

Prototype: `double* raz(double t[], int n);`

Note: on retourne souvent pour une fonction opérant sur un tableau l'adresse de ce tableau, ce qui permet d'enchaîner les appels dans une seule ligne. Par exemple : `affiche_tab(raz(t, n), n);`

Exercice 3: Recherche d'éléments sur critère (*). Écrivez une fonction qui retourne l'indice du premier élément strictement négatif parmi les n premiers éléments d'un tableau de réels (-1 si aucun élément n'est négatif).

Prototype: `int indice_premier_negatif (double t[], int n);`

Exercice 4: Maximum d'un tableau (*). Écrivez une fonction qui retourne la valeur du plus grand des n premiers éléments d'un tableau de réels.

Prototype: `double valeur_plus_grand (double t[], int n);`

Exercice 5: Indice du maximum d'un tableau (*). Écrivez la fonction qui retourne l'indice du plus grand des n premiers éléments d'un tableau de réels (en cas d'ex-æquo, l'indice du premier d'entre eux).

Prototype: `int indice_plus_grand (double t[], int n);`

Exercice 6: Un tableau est-il trié ? (*). Écrivez la fonction qui retourne 1 si les n premiers éléments du tableau sont en ordre croissant (au sens large) et 0 sinon.

Prototype: `int est_trie (double t[], int n);`

Exercice 7: Trouver la moyenne (P). Écrivez un programme qui demande à l'utilisateur une liste de entiers et trouve un élément qui est égal à la moyenne du tableau.

Activité PLaTon : TP02

2 Algorithmes récursifs

Exercice 8: PGCD (P). Écrivez un programme qui calcule le PGCD de deux entiers donnés à l'aide de l'algorithme d'Euclide :

Algorithme 1 PGCD

Données: deux entiers $0 \leq a < b$
Résultat: le plus grand commun diviseur de a et b

1: **if** $a = 0$ **then**
2: **return** b
3: **else**
4: $r \leftarrow b\%a$
5: **return** $\mathrm{PGCD}(r, a)$
6: **end if**

Activité PLaTon : TP02

Exercice 9: Recherche dichotomique (***). Dans le fichier `algo-tp02-dicho.c`, complétez la fonction `recherche_dichotomique`.

Cette fonction prend en entrée un tableau *trié* t, ainsi qu'un entier x à rechercher, et les limites inférieure d et supérieure f, et doit renvoyer la position dans le tableau où se trouve la valeur x, ou -1 si x n'est pas dans t.

La fonction doit trouver l'élément x dans le tableau t entre les positions d (incluse) et f (exclue) comme suit : considérer la position m au milieu, entre d et f, si l'élément $t[m]$ est inférieur à x, les éléments de d à m sont trop petits, et nous recommençons donc la recherche entre m et f, de même, si $t[m]$ est supérieur à x, nous recommençons la recherche entre d et m.

Testez votre fonction en compilant et en exécutant le programme `algo-tp02-dicho.c`.

3 Manipulation de tableaux et de la mémoire

Exercice 10: Allocation dynamique de mémoire (P). Cet exercice est un test simple d'allocation dynamique de mémoire. Il vous sera demandé d'allouer plusieurs tableaux, et une fonction de test vérifiera que les opérations ont bien réussi.

Activité PLaTon : TP02

Exercice 11: Copie simple de tableaux (*). Écrivez une fonction qui copie les n premiers éléments d'un tableau `source` de réels dans le tableau `destination` et retourne l'adresse du tableau `destination`.

Prototype: `double* copie (double destination[], double source[], int n);`

Exercice 12: Copie sécurisée de tableaux (**). Écrivez une fonction qui copie de manière sécurisée les n premiers éléments d'un tableau `source` de réels dans le tableau `destination` et retourne l'adresse du tableau `destination`.
Attention: On devra prendre en compte le fait que les deux tableaux peuvent se recouvrir partiellement.

Prototype: `double* copie_sec (double destination[], double source[], int n);`

Exercice 13: Décalage droite d'un tableau de réels (*). Écrivez une fonction qui décale de k éléments vers la droite les éléments d'un tableau de réels compris entre l'indice d (inclus) et l'indice f (exclus) et retourne l'adresse du tableau.

Prototype: `double* decale_droite (double t[], int k, int d, int f);`

Exercice 14: Redimensionnement d'un tableau (***). Écrivez une fonction qui prend en entrée un tableau `tbl`, sa longueur n et un entier $m > n$, et qui crée une copie de `tbl` de taille m (c'est-à-dire que les n premiers éléments sont égaux à ceux de `tbl`). Votre programme doit libérer le tableau `tbl` et retourner un pointeur vers le nouveau tableau.

Prototype: `double* redim (double tbl[], int n, int m);`

4 Les chaînes de caractères

Exercice 15: Retournement d'une chaîne (*). Écrivez une fonction qui prend en argument une chaîne de caractères, la renverse sur elle-même (`"toto!"` $\rightarrow$ `"!otot"`) et retourne l'adresse de cette chaîne.

Prototype: `char* miroir (char* s);`

Exercice 16: Recherche de motif (1) (**). Écrivez une fonction qui prend en argument deux chaînes de caractères et retourne 1 si la première chaîne de caractères commence par la seconde et 0 sinon. Écrivez un programme pour tester cette fonction.

Prototype: `int debute_par (char* chaine1, char* chaine2);`

Par exemple, `debute_par("ESIEE","ES")` $\rightarrow 1$

Exercice 17: Recherche de motif (2) (**). Écrivez une fonction qui prend en argument deux chaînes de caractères et retourne la position de la première occurrence de la `chaine2` dans la `chaine1` si elle y est présente et -1 sinon.

Prototype: `int presence (char* chaine1, char* chaine2);`

Exercice 18: Conversion chaîne de caractères en entier (**). Écrivez une fonction qui prend en argument une chaîne de caractères représentant un entier en décimal et retourne l'entier équivalent (`"123"` $\rightarrow$ 123).

Prototype: `int chaine_vers_entier (char* s);`

Note: on supposera que la chaîne est correcte et représente bien un entier.

5 Pour aller plus loin

Exercice 19: Compteur de mots (***). Écrivez une fonction qui compte le nombre de mots dans une chaîne de caractères (la documentation devra définir exactement ce que l'on entend par mot). Écrivez un programme pour tester cette fonction.

Prototype: `int compte_mots (char* s);`

Exercice 20: Les huit reines (***). Écrivez une fonction calculant et affichant toutes les solutions au problème des huit reines.

Huit reines doivent être placées sur un échiquier 8×8 sans que deux d'entre elles soient en prise, c'est à dire sur une même ligne, même colonne ou même diagonale.

Prototype: `int huit_reines (void);`