Université des Sciences et de la Technologie d'Oran Mohamed Boudiaf -USTOMB
Faculté de Génie Electrique
Master RT/II/ESE
Module : Programmation Orientée Objet

TP3 : ALLOCATION DYNAMIQUE, STRUCTURES, CHAINES DE CARACTERES EN C++

EXERCICE 1:

Ecrire une fonction en C++ qui permet d'éliminer les éléments nuls dans un vecteur V en les translatant vers la gauche et en utilisant la classe « **vector** ».

Exemple:

5 0 0 2 3 0 1 → 5 2 3 1 0 0 0

EXERCICE 2:

Ecrire un programme en C++ qui demande à l'utilisateur un nombre décimal et qui affiche alors sa conversion en binaire. Par exemple, si l'utilisateur entre 12 le programme devra afficher: 1100 en binaire.

EXERCICE 3:

Écrire un programme qui lit 20 nombres entiers dans un tableau avant d'en rechercher le plus grand et le plus petit et leurs indices:
Le max, le min ainsi que leurs indices doivent être retournés par des fonctions.

EXERCICE 4 :

Ecrire un programme qui demande à l'utilisateur de taper 10 entiers qui seront stockés dans un tableau. Le programme doit trier le tableau par ordre croissant et doit afficher le tableau.

Algorithme suggéré :

On cherche l'indice du plus petit élément parmi les indices de 0 à 9 et on échange cet élément avec t[0].
On cherche l'indice du plus petit élément parmi les indices de 1 à 9 et on échange cet élément avec t[1].
On cherche l'indice du plus petit élément parmi les indices de 2 à 9 et on échange cet élément avec t[2].
…
On cherche l'indice du plus petit élément parmi les indices de 8 à 9 et on échange cet élément avec t[8].

EXERCICE 5 :

Ecrire un programme qui demande à l'utilisateur de taper 10 entiers qui seront stockés dans un tableau. Le programme doit trier le tableau par ordre croissant et doit afficher le tableau.

Algorithme suggéré (tri bulle) :

On parcourt le tableau en comparant t[0] et t[1] et en échangeant ces éléments s'ils ne sont pas dans le bon ordre.
On recommence le processus en comparant t[1] et t[2],… et ainsi de suite jusqu'à t[8] et t[9]. On compte lors de ce parcours le nombre d'échanges effectués. On fait autant de parcours que nécessaire jusqu'à ce que le nombre d'échanges soit nul : le tableau sera alors trié.

EXERCICE 6:

Histogramme : écrire un programme qui permet d'afficher l'histogramme d'une image connaissant les niveaux de gris et le nombre de pixels correspondant. L'affichage se fera selon le schéma suivant :

Niveau de gris Nombre de Pixels Histogramme

5	9	*********
8	17	**********

- Prendre comme exemple les valeurs suivantes :

Niveaux de gris	0	1	2	3	4	5	6	7	8	9
Nombre de pixels	19	5	8	15	7	2	9	17	12	1

EXERCICE 7:

Écrire un programme qui demande à l'opérateur de rentrer deux matrices à éléments réels A(NxK) et B(KxM) puis calcule et affiche la matrice produit P = A x B.

a. en utilisant le « formalisme de vector des tableaux à deux indices »

b. en utilisant le « formalisme pointeur ».

EXERCICE 8:

Écrire un programme qui demande à l'opérateur de rentrer deux matrices à éléments réels A(NxK) et B(NxK) puis génère une nouvelle matrice AB(N x 2*K) obtenue par concaténation des deux matrices A et B (collage horizontal des matrices).

Ecrire un programme en c++ qui :

- Remplit une matrice carrée **M**.
- Transforme cette matrice en sa transposée **MT** par rapport à la deuxième diagonal
- Permet de trier toutes les lignes de la matrice **M** en créant une nouvelle matrice triée **ML**.
- Affiche les 3 matrices « **M, MT, ML** » sous leur vrai format matriciel.

EXERCICE 9:

Écrire en C++ un programme main qui fait l'appel des 3 fonctions suivante permettant de:

1. Saisir les informations de N étudiants en utilisant un tableau dynamique, sachant que la structure de chacun doit comporter les champs suivants :

nom : chaine de caractères.
prenom : chaine de caractères.
note_test : réel, coef_test : 0.4
note_ex am: réel, coef_exam: 0.6
moyenne : réel.

2. Trier le tableau de N étudiants par ordre croissant des moyennes.

3. Afficher le contenu du tableau avant et après le tri.

EXERCICE 10:

Définir en C++ un tableau dynamique de taille ***N*** et de structure ***Date*** contenant trois champs de type entier pour identifier le jour, le mois et l'année.

Après avoir rempli tous les éléments du tableau, rechercher puis afficher la date la plus récente.

LISTES CHAINEES ET CHAINES DE CARACTERES

Exercice 1

Écrire une fonction qui permet de fusionner deux listes chaînées d'entiers après les avoir triées par ordre croissant en une troisième liste telle que celle-ci contienne les éléments de la *liste1* et les éléments de la *liste2* triés par ordre croissant.

Exercice 2

On se propose de modéliser la gestion des patients dans un cabinet médical. Un patient est caractérisé par: le nom, le prénom et un champ ***rdv*** (pour rendez-vous) de type entier indiquant si le patient a un rendez-vous ou pas: 0 si le patient est sans rendez-vous, 1 si le patient est avec rendez-vous.

Avant d'être consultés par le médecin, les patients sont entrés dans une salle d'attente qui sera modélisée par une liste chaînée de patients. Une secrétaire fait entrer les patients ayant un ***rdv=1*** selon leur ordre d'arrivée, ensuite elle fait entrer les autres patients (ceux dont le ***rdv=0***) selon leur ordre d'arrivée aussi.

Définir la structure de données ***Patient***.

Définir la structure de données ***Cellule***.

Définir le type ***liste*** comme un pointeur sur la structure ***Cellule***.

Dans une deuxième étape, on vous demande d'écrire les fonctions suivantes :

1. ***liste AjoutPatient (liste tete, Patient P)*** , qui permet d'ajouter un nouveau patient à la fin de la liste identifiée par son pointeur tete.
2. ***void RendezVous (liste tete, int *rdv, int *sansRdv)***, qui compte et retourne le nombre de patients avec rendez-vous, et le nombre de patients sans rendez-vous.
3. ***liste SupprimePatient (list tete)*** , qui permet de faire entrer un patient en consultation. Cette opération est effectuée de la manière suivante : s'il n'y a aucun patient avec rendez-vous alors c'est le premier patient de la liste qui est supprimé.

Sinon on supprime le premier patient qui a un rendez-vous.

4. ***void ConsulterSalleAttente (liste tete)*** , qui affiche tout d'abord les patients avec rendez-vous, ensuite les patients sans rendez-vous.

EXERCICE 3:

Ecrire un programme C++ qui décide si un mot est un palindrome. Notre programme doit appeler une fonction prenant en entrée une chaîne de caractères et renvoyant un booléen « Fonction **palindrome**(mot : chaîne de caractères) : booléen ».

On appelle palindrome une chaine qui se lit de la même façon de gauche à droite ou de droite à gauche. Par exemple:

« ***Esope reste ici et se repose***», « ***Engage le jeu, que je le gagne*** » sont des chaines palindromes.

EXERCICE 4:

Ecrire un programme C++ qui appelle les fonctions suivantes :

1. On cherche à compter le nombre d'occurrences d'un mot donné dans un texte. On va donc écrire une fonction sous la forme : Fonction **compte_occurences**(mot, texte : chaînes de caractères) : Entier
2. Une fonction void **concatene_chaines**(char *debut_chaine, char *ajout) qui concatène les deux chaînes de caractères (la seconde à la suite de la première).
3. Ecrire une autre fonction qui supprime le caractère blanc (espace) dans une phrase.