Université de Nîmes

Introduction à la programmation objet en C++

Feuille de TD N°2 : Les pointeurs en C
(Extraite d'exercices proposés par F. Faber)

Exercice 1 : Tableaux et pointeurs

Écrire un programme qui lit deux tableaux A et B et leurs dimensions N et M au clavier et qui ajoute les éléments de B à la fin de A. Utiliser le formalisme pointeur à chaque fois que cela est possible.

Exercice 2 : Arithmétique des pointeurs

Soit P un pointeur qui 'pointe' sur un tableau A:

```
int A[] = {12, 23, 34, 45, 56, 67, 78, 89, 90};
int *P;
P = A;
```

Quelles valeurs ou adresses fournissent ces expressions:

a) ***P+2**
b) ***(P+2)**
c) **&P+1**
d) **&A[4]-3**
e) **A+3**
f) **&A[7]-P**
g) **P+(*P-10)**
h) ***(P+*(P+8)-A[7])**

Exercice 3 : Occurrences d'un entier dans un tableau

Écrire un programme qui lit un entier X et un tableau A du type **int** au clavier et élimine toutes les occurrences de X dans A en tassant les éléments restants. Le programme utilisera les pointeurs P1 et P2 pour parcourir le tableau.

Exercice 4 : Ordre inverse

Écrire un programme qui range les éléments d'un tableau A du type **int** dans l'ordre inverse. Le programme utilisera des pointeurs P1 et P2 et une variable numérique AIDE pour la permutation des éléments.

Exercice 5 : Ajout de deux tableaux

Écrire un programme qui lit deux tableaux d'entiers A et B et leurs dimensions N et M au clavier et qui ajoute les éléments de B à la fin de A. Utiliser deux pointeurs PA et PB pour le transfert et afficher le tableau résultant A.

Exercice 6 : Chaîne palindrome

Écrire de deux façons différentes, un programme qui vérifie sans utiliser une fonction de <*string*>, si une chaîne CH introduite au clavier est un palindrome:

a) en utilisant uniquement le formalisme tableau

b) en utilisant des pointeurs au lieu des indices numériques

Rappel: Un palindrome est un mot qui reste le même qu'on le lise de gauche à droite ou de droite à gauche:

Exemples:

PIERRE ==> n'est pas un palindrome

OTTO ==> est un palindrome

23432 ==> est un palindrome

Exercice 7 : Allocation dynamique

Écrire un programme qui lit 10 phrases d'une longueur maximale de 200 caractères au clavier et qui les mémorise dans un tableau de pointeurs sur **char** en réservant dynamiquement l'emplacement en mémoire pour les chaînes. Ensuite, l'ordre des phrases est inversé en modifiant les pointeurs et le tableau résultant est affiché.

Exercice 8 : dés allocation dynamique

Écrire un programme qui lit 10 mots au clavier (longueur maximale: 50 caractères) et attribue leurs adresses à un tableau de pointeurs MOT. Effacer les 10 mots un à un, en suivant l'ordre lexicographique et en libérant leur espace en mémoire. Afficher à chaque fois les mots restants en attendant la confirmation de l'utilisateur (par 'Enter').

Exercice 9 : Allocation et libération dynamique et String

Écrire un programme qui lit 10 mots au clavier (longueur maximale: 50 caractères) et attribue leurs adresses à un tableau de pointeurs MOT. Copier les mots selon l'ordre lexicographique en une seule 'phrase' dont l'adresse est affectée à un pointeur PHRASE. Réserver l'espace nécessaire à la PHRASE avant de copier les mots. Libérer la mémoire occupée par chaque mot après l'avoir copié. Utiliser les fonctions de <*string*>.

Exercice 10 : Allocation dynamique et phrases

Écrire un programme qui lit 10 phrases au clavier (longueur maximale: 50 caractères) et attribue leurs adresses à un tableau de pointeurs MOT. Réserver dynamiquement l'emplacement en mémoire pour les mots. Trier les phrases lexicographiquement en n'échangeant que les pointeurs. Utiliser la méthode de tri par propagation