TP 7 : Révisions, listes chaînées

Programmation en C (LC4)

Semaine du 12 mars 2007

Exercice 1 — *Reprise du TD 7*

1. Écrire une fonction qui prend en argument une chaîne de caractères et qui renvoie une nouvelle chaîne de caractères contenant l'inversion de la chaîne de caractères donnée en argument.
2. On appelle palindrome une suite de caractères qui se lit de la même façon dans les deux sens (exemple : "laval", "ressasser"). Écrire une fonction qui teste si une chaîne est un palindrome.
3. En utilisant la fonction **int** getchar(**void**) (déclarée dans <stdio.h>) qui lit un caractère saisi au clavier et le renvoie (sous forme d'un **int**), écrire une fonction qui lit et renvoie une chaîne de caractères tapée au clavier. Le caractère de saut de ligne '\n' indiquera la fin de la saisie.
4. Écrire un programme qui demande à l'utilisateur de saisir une chaîne de caractères et affiche un message indiquant si la chaîne est un palindrome.

Exercice 2 On souhaite classer par ordre alphabétique une liste de personnes dont on a stocké le nom et le prénom. On respecte les règles de classement suivantes :

– les personnes sont classées suivant leurs noms de famille,
– si deux personnes ont le même nom de famille, elles sont classées d'après leur prénom.

La structure de données suivante permet de stocker les renseignements concernant une personne :

```
typedef struct personne {
    char *nom;
    char *prenom;
} personne;
```

1. Écrire une fonction change_chaine() qui met en majuscule le premier caractère d'une chaîne de caractères passée en paramètre et en minuscules le reste des caractères de cette chaîne. Écrire une fonction change_personne() qui applique la conversion précédente aux champs d'une variable personne.
2. Écrire une fonction **int** compare_personnes(personne p1, personne p2) qui compare deux variables personnes comme indiqué dans l'énoncé. La valeur de retour de cette fonction aura la même signification que celle de la fonction strcmp() :
   – < 0 si p1 est situé avant p2,
   – > 0 si p2 est situé avant p1,
   – $== 0$ si p1 et p2 sont identiques.
   On supposera que les champs nom et prenom ont d'abord été convertis comme à la question précédente.
3. Écrire une fonction echange_personnes() qui échange les contenus de deux variables de type personne.
4. Écrire une fonction tri_personnes() qui trie un tableau de données de type personne comme indiqué dans l'énoncé (on utilisera un tri facile à implémenter...).

5. Écrire une fonction qui récupère une chaîne de caractère saisie au clavier, le caractère '\n' indiquant la fin de la saisie (on pourra bien sûr réutiliser la fonction de l'exercice précédent...).
6. Écrire une fonction main() qui effectue la saisie d'un tableau de personnes (il faudra demander la taille du tableau ou bien indiquer un moyen d'arrêter la saisie des noms et prénoms), convertit les prénoms et les noms comme à la première question, trie le tableau et l'affiche.

Exercice 3 — *Reprise du TD 7* Soit P un polynôme de degré d et de coefficients p_i ($i \in [\![0, d]\!]$) : $P[X] = \sum_{i=0}^{d} p_i X^i$

On souhaite représenter P en occupant le moins d'espace mémoire possible, en particulier quand le degré est élévé et/ou qu'il y a peu de coefficients non nuls.

L'idée consiste à ne considérer que les monômes $p_i X^i$ « utiles » c'est-à-dire tels que $p_i \neq 0$. Les monômes « utiles » seront stockés dans une liste chaînée dont chaque élément contiendra un **double** (le coefficient p_i du monôme) et un **int** (l'exposant i du monôme). Chaque coefficient d'un monôme « utile » sera ainsi chaîné via un pointeur au prochain monôme « utile » de degré strictement inférieur (ou NULL s'il n'y a plus de monôme « utile »).

1. Écrire précisément la structure de liste chaînée à utiliser.
2. Écrire une fonction qui alloue et initialise la mémoire nécessaire à un monôme.
3. Écrire une fonction qui libère toute la mémoire utilisée par un polynôme.
4. Écrire une fonction d'affichage.
5. Écrire une fonction convertissant un tableau de coefficients en liste de monôme « utiles ».
6. Écrire une fonction de dérivation d'un polynôme sous cette forme.
7. Écrire une fonction d'addition de deux polynômes sous cette forme.