TD 5 : Chaînes de caractères

Programmation en C (LC4)

Semaine du 26 février 2007

1 Chaînes de caractères

On va utiliser les fonctions de la bibliothèque standard destinées à la manipulation des chaînes de caractères. Il ne faut pas oublier d'inclure <string.h> avant de les utiliser.

On se donne la structure suivante :

struct traduction { **char** *a; **char** *b; };

On peut alors représenter un dictionnaire bilingue par un tableau de **struct** traduction, chaque case du tableau contenant une paire de mots se traduisant l'un en l'autre. On supposera que les dictionnaires sont triés par ordre lexicographique (l'ordre alphabétique) sur le champ a.

Exercice 1 Écrire une fonction **char** *recherche(**struct** traduction *dico, **int** n, **char** *s) qui recherche le mot s dans les champs a des **struct** traduction du dictionnaire dico (de taille n), et renvoie en résultat sa traduction, ou NULL si elle ne trouve pas s. Comme le dictionnaire est supposé trié, utilisez une recherche dichotomique, ce qui est nettement plus efficace qu'une recherche linéaire.

Utilisez la fonction **int** strcmp(**char** *s1, **char** *s2) pour comparer deux chaînes s1 et s2 : elle renvoie un nombre négatif si la chaîne s1 est avant s2 (pour l'ordre lexicographique), 0 si elles sont égales, et un nombre positif sinon.

Exercice 2 Écrire une fonction **int** nombre_espaces(**char** *s) qui renvoie le nombre de caractères « espace » présents dans la chaîne s.

Utilisez la fonction **char** *strchr(**const char** *s, **int** c) qui renvoie l'adresse de la première occurrence du caractère c dans la chaîne s en partant du début de la chaîne.

Exercice 3 En utilisant la fonction précédente, écrire une fonction **char** **decoupe(**char** *s) qui découpe la chaîne s en mots, en fait un tableau de **char** * dont le dernier élément est égal à NULL.

Par exemple, si s = "ga␣bdu␣meu", elle devra renvoyer le tableau :

{ "ga", "bdu", "meu", NULL}

Vous pourrez utiliser :
- la fonction **char** *strcpy(**char** *dest, **const char** *src) qui copie la chaîne src dans dest, y compris le caractère de fin de chaîne (et qui renvoie dest). *Attention :* strcpy() n'alloue pas de mémoire, il faut donc que dest désigne une chaîne de caractère suffisamment longue,
- la fonction **char** *strncpy(**char** *dest, **const char** *src, size_t n), identique sauf que pas plus de n caractères de src ne seront copiés (donc, s'il n'y a pas de caractère '\0' dans les n premiers caractères de src, le résultat n'aura pas de caractère de fin de chaîne).

Exercice 4 Écrire une fonction **char** *reconstruit(**char** **tab) qui effectue la transformation inverse.

Vous pourrez utiliser :
- la fonction size_t strlen(**const char** *s) qui renvoie le nombre de caractères de s sans tenir compte du caractère de fin de chaîne,

– la fonction **char** ∗strcat(**char** ∗dest, **const char** ∗src) qui concatène la chaîne src à la suite de dest (et renvoie dest). *Attention :* même remarque que pour strcpy(), il faut que dest soit suffisamment grand. strcat() écrit par dessus le caractère '\0' à la fin de dest puis ajoute un '\0' à la fin de la concaténation.

Exercice 5 Écrire une fonction **char** ∗traducteur(**struct** traduction ∗dico, **int** n, **char** ∗s) (à l'aide des fonctions précédentes) qui traduit mot à mot la chaîne s à l'aide du dictionnaire dico contenant n mots.

2 Mémoire

En incluant <string.h>, on a aussi accès à des fonctions destinées à la manipulation de la mémoire. La fonction **void** ∗memcpy(**void** ∗dest, **const void** ∗src, size_t n) permet de copier n octets de la zone mémoire pointée par src vers la zone mémoire pointée par dest (et renvoie dest). *Attention :* il ne faut pas que les deux zones mémoires se chevauchent.

Exercice 6 En utilisant la fonction memcpy(), écrire une fonction **void** init_tab(**int** ∗tab, **int** n, **int** val) qui initialise les n éléments du tableau vers lequel pointe tab avec la valeur val. Essayez de faire le *moins d'appels possibles* à la fonction memcpy().

Exercice 7 La fonction **void** ∗memset(**void** ∗s, **int** c, size_t n) remplit les n premiers octets de la zone mémoire vers laquelle pointe s avec le caractère dont le code ASCII est l'entier c (et renvoie s).

– Peut-on utiliser cette fonction pour réécrire la fonction init_tab() de la question précédente ?
– Peut-on se servir de memset() pour faire l'équivalent de ce que fait calloc() ?