TD4

Langage C (LC4)

semaine du 18 février 2013

1 Échauffement

Exercice 1 Complétez le tableau en indiquant les valeurs des différentes variables au terme de chaque instruction du programme suivant. On peut aussi indiquer sur quoi pointent les pointeurs :

programme	`a`	`b`	`c`	`p1, *p1`	`p2, *p2`
`int a, b, c, *p1, *p2;`	×	×	×	×	×
`a = 1, b = 2, c = 3;`					
`p1 = &a, p2 = &c;`					
`*p1 = (*p2)++;`					
`p1 = p2;`					
`p2 = &b;`					
`*p1 -= *p2;`					
`++*p2;`					
`*p1 *= *p2;`					
`a = ++*p2 * *p1;`					
`p1 = &a;`					
`*p2 = *p1 /= *p2;`					

2 Passage de paramètres par adresse

2.1 Avec des pointeurs...

Exercice 2 Soient `adra` et `adrb` les adresses de deux variables `a` et `b` de type `int`. Écrivez une fonction `void echange(int *adra, int *adrb)` qui échange les valeurs de `a` et `b`.

Exercice 3 L'instruction `scanf("%d %d", &n, &m);` lit deux valeurs entières écrites en base décimale et les place dans les variables `n` et `m`. À quoi servent les `&` ?

Exercice 4 Écrivez une fonction `int read_int(int *adr)` qui lit sur l'entrée standard (le clavier, par défaut) une valeur entière écrite en décimal et la stocke à l'adresse `adr`. En pratique, la fonction lira (au moyen de `getchar()`) une suite de caractères entre `'0'` et `'9'` et convertira cette suite en un entier. La valeur de retour de la fonction servira à signaler s'il y a eu une erreur de saisie. La fonction renverra 1 si la valeur entrée était bien un entier et 0 sinon.

Exercice 5 Ecrire une fonction `int compte(char *s, char c)` qui renvoie le nombre d'occurrences de `c` dans `s`.

3 Chaînes de caractères

Dans la bibliothèque `<string.h>`, les chaînes de caractères sont définies comme des tableaux de caractères dans lesquels la fin de la chaîne est signalée par le caractère de code 0, noté `'\0'`.

Quelques remarques préliminaires :

- pour stocker une chaîne de n caractères, il est nécessaire de réserver $n + 1$ caractères, pour pouvoir stocker le `'\0'` ;
- ne pas confondre :
  - le caractère `'\0'` ;
  - une chaîne vide, qu'on peut noter `""` et qui est représentée en mémoire par une chaîne dont la première case a pour valeur `'\0'` ;
  - la valeur `NULL`, qui peut être affectée à une variable de type `char *` et qui indique qu'elle ne pointe vers aucune zone de mémoire.

L'objectif de cet exercice est de manipuler des chaînes de caractères *sans utiliser les fonctions de la bibliothèque `string`*. En particulier, vous devrez récrire vous-mêmes certaines fonctions de cette bibliothèque.

Vous implémenterez les fonctions suivantes.

Exercice 6 `int strlen(const char *)` renvoie le nombre de caractères d'une chaîne. Le `const char *` spécifie que le contenu de la chaîne ne sera pas modifié par la fonction.

Exercice 7 `int strcmp(const char *s1, const char *s2)` compare deux chaînes `s1` et `s2` : elle renvoie un nombre négatif si la chaîne `s1` est avant `s2` (pour l'ordre lexicographique, l'ordre d'un dictionnaire usuel), 0 si elles sont égales, et un nombre positif sinon.

Exercice 8 `int palindrome(const char *)` (qui n'est pas dans `<string.h>`) teste si une chaîne est un palindrome, c'est-à-dire une chaîne identique qu'on la lise de la gauche vers la droite ou de la droite vers la gauche (par exemple les mots *laval* ou *subitomotibus*).

Exercice 9 `char *strchr(const char *s, int c)` renvoie l'adresse de la première occurrence du caractère `c` dans la chaîne `s` en partant du début de la chaîne.

Exercice 10 `char *strsep(char **stringp, const char *separateurs)` coupe une chaîne en deux à la première occurrence d'un caractère séparateur.

La chaîne à couper est `*stringp`, la chaîne `separateurs` contient tous les caractères séparateurs. Le premier caractère séparateur trouvé est remplacé par `'\0'` dans `*stringp` et la valeur de `*stringp` est modifiée (d'où l'intérêt de passer l'adresse de la chaîne) pour pointer sur le caractère suivant. L'ancienne valeur de `*stringp` est renvoyée.