Révisions

Programmation en C (LC4)

31 mai 2007

1 Pointeurs

Exercice 1 Compléter le tableau en indiquant les valeurs des différentes variables au terme de chaque instruction du programme suivant (on indiquera également 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;`					

Exercice 2 Écrivez une fonction qui échange deux variables entières a et b. Écrire l'appel de cette fonction dans la fonction main().

Exercice 3 Écrire une fonction qui échange deux tableaux d'entiers (on veut donc échanger les pointeurs).

Exercice 4 On considère les instructions suivantes :

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

À quelles valeurs correspondent 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) $*(p + *(p + 8) - a[7])$

2 Chaînes de caractères

Exercice 5 Écrire une fonction **char** *recherche_char(**char** *s,**char** c) qui renvoie un pointeur vers la première occurrence du caractère c (le caractère passé en argument, pas le caractère 'c') dans la chaine s. Si ce caractère n'apparaît pas dans la chaine, la fonction devra renvoyer NULL.

Exercice 6 À l'aide de recherche_char(), écrire une fonction **int** compte_char(**char** *s, **char** c) qui compte le nombre d'occurrences de c dans s.

Exercice 7 Écrire une fonction **char** *strdup(**char** *s) qui crée une copie de la chaîne de caractères qui lui est passée en paramètre : on utilisera les fonctions adéquates de la bibliothèque C standard (<string.h> et <stdlib.h> en particulier) pour calculer la longueur de la chaîne s, allouer l'espace mémoire nécessaire pour la copie et recopier s dans la nouvelle chaîne.

Exercice 8 Écrire une fonction **int** strend(**char** *s, **char** *t) qui indique si la chaîne t est présente à la fin de la chaîne s.

Exercice 9 Écrire une fonction **int** strcasecmp(**char** *s1, **char** *s2) qui compare les deux chaînes de caractères s1 et s2 comme strcmp() mais sans tenir compte de la casse.

3 Listes chaînées

Exercice 10

- Définir une structure **struct** couple_s représentant un couple d'entiers.
- Définir un nouveau type couple_t correspondant à un pointeur sur une **struct** couple_s.
- Écrire une fonction couple_t alloue_couple(**int** x, **int** y) qui crée un couple (x, y).
- Écrire une fonction **void** libere_couple(couple_t c) qui libère la mémoire occupée par un couple.
- Écrire une fonction **void** affiche_couple(couple_t c) qui affiche un couple sous la forme `(x, y)`.

Exercice 11

- Définir une structure **struct** liste_s permettant de représenter des listes chainées de pointeurs vers des couple_t.
- Écrire une fonction **void** affiche_liste (liste_t l) qui affiche une liste.

Exercice 12 Écrire une fonction liste_t concatene(couple_t c, liste_t l) qui ajoute c en tête de la liste et renvoie en résultat un pointeur vers le nouveau maillon créé.

Exercice 13 Écrire une fonction **int** compte_differents(liste_t l) qui compte le nombre de couples (x, y) avec $x \neq y$ contenus dans la liste l.

Exercice 14 Écrire une fonction **int** dedouble(liste_t l) qui parcourt la liste l, et qui, à la suite de chaque couple (x, y), insère le couple (y, x).

4 Entrées-sorties

Exercice 15 En utilisant la fonction :

```
int fgetc(FILE *fp);
```

Écrire une fonction **char** *lire_ligne_clavier(**void**) qui lit une ligne tapée au clavier (jusqu'au caractère de retour de ligne) et le stocke dans un **char** *, puis renvoie ce dernier.

On rappelle les prototypes des fonctions de lecture et d'écriture non formatées de la bibliothèque C standard :

```
#include <stdio.h>

size_t fread(void *tableau, size_t taille, size_t n, FILE *f);
```

Cette fonction tente de lire, dans le fichier f, n blocs de taille octets, et les écrit à l'adresse tableau, qui doit donc pointer vers un tableau suffisamment grand que l'on aura créé au préalable. Elle renvoie le nombre de blocs lus, qui peut être plus petit que n (par exemple si l'on a atteint la fin du fichier).

```
size_t fwrite(void *tableau, size_t taille, size_t n, FILE *f);
```

Cette fonction agit de manière similaire : elle écrit, dans le fichier f un nombre n de blocs de taille taille octets qu'elle lit dans le tableau pointé par tableau. Elle renvoie le nombre de blocs écrits en résultat. S'il est inférieur à n, c'est qu'il y a eu une erreur.

Exercice 16 Écrire une fonction **int** copie_fichier_par_bloc(**const char** *src, **const char** *dest) qui copie le fichier dont le chemin est indiqué dans la chaîne de caractères

Exercice 17 On définit la structure **struct** fiche_s pour stocker des données sur une personne (nom, prénom, âge, code postal, taille) de la manière suivante :

```
struct donnees_s {
    int age;
    int code_postal;
    float taille;
};
typedef struct donnees_s donnees_s;

struct fiche_s {
    char *nom;
    char *prenom;
    donnees_s data;
};
typedef struct fiche_s *fiche_t;
```

Écrire deux fonctions, ecrire_fiche (), lire_fiche () qui écrivent (et lisent) une fiche dans une fichier (FILE *) en utilisant *uniquement* les fonctions fread() et fwrite().

Remarque : les chaînes de caractères nom et prenom sont de taille variable. Une solution peut être de commencer par écrire la taille de la chaîne de caractère avant d'écrire la chaîne elle-même.