UNIVERSITÉ BORDEAUX 1 Sciences Technologies

Année 2005–2006 — Session de Septembre 2006

Parcours : INF2, MAT2 **Date :** 5 Septembre 2006
Code UE : INF101 **Durée :** 1h30
Épreuve de : Initiation à la programmation

Épreuve de G. Bianchi, F. Pereyrol, M. Zeitoun

Notes de cours, TD, TP autorisées. Autres documents interdits.
Le barème comporte 20 points et est donné à titre indicatif.
On pourra utiliser la fonction `lire_entier()` du cours, et les fonctions de la bibliothèque standard.
Sauf mention contraire, on pourra utiliser les fonctions des questions précédentes, même si elles n'ont pas été écrites.

Exercice 1 (4pts) Répondre par vrai ou faux aux questions suivantes. Il n'est pas demandé de justification. Ici, vrai signifie « toujours vrai » et faux signifie « pas toujours vrai ».
Barème : $+0{,}5$ pour une réponse juste, $-0{,}5$ pour une réponse fausse.

1. Si `a`, `j` et `i` sont déclarées de type `int`, l'instruction `a = 2*j-i;` provoque un effet de bord.
2. Si `t` est un pointeur, l'expression `*(t+j+1)` est équivalente à l'expression `t[j+1]`.
3. Si `u` et `v` sont deux tableaux de même type et même taille l'instruction `u = v;` recopie le contenu de chaque case de `v` dans la case correspondante de `u`.
4. La séquence suivante est incorrecte en C :

```
char c;
char p[10], *q;
p = q = &c;
```

5. Dans un tableau de `char`, il est impossible d'écrire au delà de la première case contenant `'\0'`.
6. L'expression `((++i) + (i--))` s'évalue en 4 lorsque `i` vaut initialement 1.
7. Pour pouvoir créer un fichier exécutable, il faut qu'au moins un fichier d'en-tête contienne une définition de la fonction `main`.
8. La fonction

```
long calcul_suite(int a, int n)
{
   if (n == 0)
      return a;
   return 2 * calcul_suite(a, n+1);
}
```

calcule le n-ième terme de la suite de terme général $u_n = a.2^n$.

Exercice 2 (4pts) 1. Écrire une fonction `void supprime_dernier_car(char *s)` qui supprime le dernier caractère d'une chaîne de caractères `s` ayant longueur non nulle.

Exemple : si `s` vaut `"abcd"`, elle vaudra `"abc"` après l'appel `supprime_dernier_car(s)`.

2. Écrire une fonction `void supprime_car(int pos, char *s)` qui supprime le caractère à la position `pos` $\geqslant 0$ d'une chaîne de caractères `s`, et décale d'une position vers la gauche les caractères qui le suivent. Si `pos` est supérieur ou égal à la longueur de `s`, la fonction ne fera rien.

Exemple : si `s` vaut `"dessous"`, elle vaudra `"dessus"` après l'appel `supprime_car(4,s)`.

3. Écrire une fonction `int position_car(char c, char *s)` qui retourne la position de la première occurrence du caractère `c` dans la chaîne `s`. Si `c` n'apparaît pas dans `s`, la fonction retourne -1.

Exemple : si `s` est la chaîne `"toto"` et `c` le caractère `'o'`, la fonction retournera 1.

4. Écrire une fonction `void supprime(char c, char *s)` qui, en utilisant les fonctions précédentes, supprime toute occurrence du caractère `c` dans la chaîne `s`.

Exercice 3 (3pts) On considère les deux suites définies par

$$\begin{aligned} u_0 &= 1, \quad u_n = 2u_{n-1} + v_{n-1} \quad \text{si } n \geqslant 1, \\ v_0 &= 2, \quad v_n = u_{n-1} - v_{n-1} \quad \text{si } n \geqslant 1. \end{aligned}$$

1. Écrire deux fonctions mutuellement récursives calculant le n-ième terme de chaque suite.
2. Sans utiliser les fonctions précédentes, écrire un programme sans argument sur la ligne de commande, et qui
   a. demande à l'utilisateur un entier positif n,
   b. alloue un tableau `u` de taille n et y range les n premiers termes de la suite $(u_k)_{k\geqslant 0}$,
   c. alloue un tableau `v` de taille n et y range les n premiers termes de la suite $(v_k)_{k\geqslant 0}$,
   d. affiche les n premiers termes de chaque suite.

Exercice 4 (4pts) Écrire une fonction prenant comme arguments deux tableaux d'entiers `T1` et `T2` et deux entiers `n1` et `n2`, et qui renvoie le nombre de valeurs des `n1` premières cases de `T1` qui sont aussi présentes dans au moins une des `n2` premières cases de `T2`. La fonction supposera, sans le vérifier, que `T1` a au moins `n1` cases, que `T2` a au moins `n2` cases, et qu'aucune valeur n'apparaît deux fois dans `T1`.

Exemple : si `T1` vaut `{0, 2, 8, 7, 4, 9}`, `n1` vaut `5`, `T2` vaut `{0, 4, 3, 0, 0, 1, 9, 4}`, `n2` vaut `8`, la fonction retournera 2. Les 2 valeurs communes sont 0 et 4 (pas 9 car il apparaît en 6ème position dans `T1` et `n1` vaut seulement 5).

Exercice 5 (2pt) On dit qu'un mot de longueur `2d` est un *carré* s'il est obtenu en répétant deux fois le même mot de longueur `d` (ici, un mot est simplement une suite de lettres).

Exemple : Le mot *toto* est un carré obtenu en répétant deux fois le mot *to* ; le mot *bonbon* est un carré obtenu en répétant deux fois le mot *bon*. Le mot vide, correspondant en C à la chaîne `""` de longueur 0, est un carré obtenu en répétant deux fois le mot vide.

Écrire une fonction `bool est_carre(char *s);` retournant `true` si `s` est un carré, et `false` sinon.

Exercice 6 (3pts) Détailler précisément le comportement du programme suivant.

```
#include <stdio.h>

int *x1[1];

int main(void)
{
   int **p, *s, t;

   t = 5;
   p = x1;
   x1[0] = &t;
   *x1[0] = 1;
   s = *p;

   printf("%d %d %d\n", **p, *s, t);

   return EXIT_SUCCESS;
}
```

Dessiner et expliquer l'évolution de l'état de la mémoire jusqu'à l'appel à `printf()`, en précisant l'évolution de leurs valeurs, et en particulier les liens entre variables de type pointeur et emplacements pointés par ces variables.