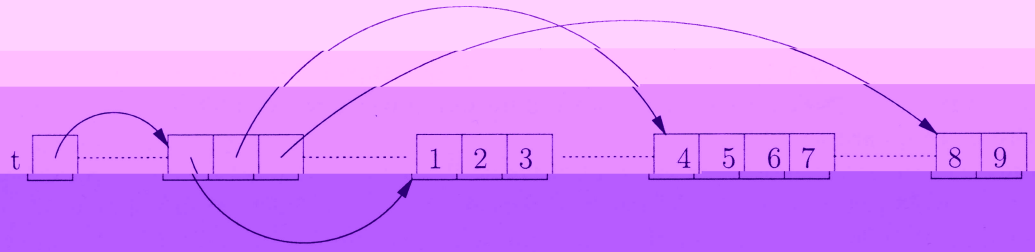
Université Paris Diderot – Paris 7

Langage C (LC4) - Examen du 21 mai 2014 - Durée : 3 heures

Exercice 1. Quelques petites questions en guise d'échauffement

1.1. On souhaite disposer en mémoire de données organisées selon le schéma suivant :



1.1.1. En supposant que les nombres sont des entiers, comment s'écrit en C la définition de la variable `t` ?

1.1.2. Donner une séquence d'instructions qui initialise complètement la zone mémoire correspondant au schéma (les points de suspension indiquent que les zones ne sont pas nécessairement contiguës en mémoire).

1.1.3. Écrire une **instruction unique** utilisant `scanf` qui remplace la valeur 6 qui a été écrite en mémoire par une valeur lue au clavier :

a) en utilisant la notation de tableau (avec des crochets) ;

b) en n'utilisant pas la notation de tableau.

1.2. Écriture de définitions de type

1.2.1. Écrire la définition du type `T1` « pointeur sur fonction renvoyant un pointeur sur caractère et ayant en premier paramètre un tableau de pointeurs sur caractère et en second paramètre un entier ».

1.2.2. Écrire la définition du type `T2` « pointeur sur fonction renvoyant un pointeur sur fonction de type `T1` et ayant en paramètres deux pointeurs sur fonctions de type `T1` ».

1.3. On considère les définitions suivantes :

```
int t[3][4] = {{1,2,3,4}, {5,6,7,8}, {9,10,11,12}};
int *p = &t[1][1], *q = &t[2][2];
int *x = t[1], *y = t[2];
int (*a)[4] = &t[1], (*b)[4] = &t[2];
```

et on suppose que, après leur réalisation, l'instruction

```
printf("%ld %ld %p\n", sizeof(void *), sizeof(int), t);
```

affiche `4 4 0x2020`.

Pour chacune des expressions suivantes, dire si elle est ou non correcte et, si elle est correcte, quel est son type et quelle est sa valeur (si elle est connue).

1.3.1. `*(*(t+2)+1)`

1.3.2. `**t+1+1`

1.3.3. `q - p`

1.3.4. `b - a`

1.3.5. `(int *)b - (int *)a`

1.3.6. `a + 2`

1.3.7. `a - x`

1.3.8. `(int *)a - x`

1.3.9. `*a + 2`

1.3.10. `*t + 1`

Exercice 2. Autour du jeu de morpion

Dans cet exercice, on s'intéresse à la mise en place d'une représentation en mémoire du jeu de morpion : il n'est pas question de programmer complètement le jeu, mais simplement de mettre en place un certain nombre d'éléments constitutifs d'un programme complet.

Rappelons que dans ce jeu, deux joueurs doivent aligner (un maximum de fois) sur une grille 5 ronds pour l'un et 5 croix pour l'autre, et ceci sur une ligne, une colonne, ou une diagonale.

Les grandes lignes d'une représentation possible du jeu, que l'on va implanter, sont les suivantes :

- un tableau contenant toutes les suites de 5 cases contiguës (appelées **quintuples**) sur la grille. On admettra ici qu'il y a $Q = 4 \times (N-4) \times (N-2)$ quintuples. Les éléments du tableau seront de type `quintuple_t` défini en 2.1. L'indice d'un quintuple dans ce tableau sera utilisé comme numéro de ce quintuple ;
- la fonction `initialiserQuintuples` est fournie et il n'est pas demandé de l'écrire. Cette fonction ne renvoie pas de valeur. Elle doit être appelée avec en paramètres un pointeur sur le type `quintuple_t` et un entier : le pointeur est supposé correspondre à un tableau de la bonne taille. La fonction construit et initialise à cette adresse le tableau de quintuples d'une grille de côté égal à l'entier ;
- la grille utilisée (que nous supposerons ici carrée) sera représentée par un tableau `grille` de `N` lignes et `N` colonnes (lignes et colonnes numérotées comme en C de 0 à $N-1$). Les éléments du tableau seront de type `point_t` défini en 2.1.

2.1. Le fichier d'interface `morpion.h` : les constantes et types

Écrire les lignes du fichier d'interface correspondant à :

- la macro définition de `N` à la valeur 40, de `VIDE` à 7 et de `Q` (nombre de quintuples) à la bonne

appartient) ;

- le prototype de la fonction `initialiserQuintuples`.

2.2. Définir les tableaux utilisés

Donner les définitions en C du tableau des quintuples et du tableau correspondant à la grille telles qu'elles devraient apparaître dans la fonction `main`.

2.3. Initialiser la grille

grille, les coordonnées d'un point et un numéro de quintuple et qui ajoute ce numéro de quintuple à la liste des numéros des quintuples contenant le point de coordonnées transmises en paramètre (le numéro du quintuple pourra être ajouté n'importe où dans la liste).

2.3.2. Écrire une fonction `creerListe` qui, pour chacun de $N \times N$ points de la grille, initialise la liste des numéros de quintuples auxquels il appartient. Cette fonction ne renverra pas de valeurs et recevra comme paramètres la grille et le tableau de quintuples.

2.3.3. Écrire une fonction `valeurPoint` qui calcule et renvoie la valeur d'un point de la grille vu par un joueur. Cette fonction aura comme paramètres la grille, les coordonnées du point et un caractère (O ou X identifiant le joueur) dont on veut calculer la valeur pour le joueur utilisant ce caractère (la valeur d'un point pour un joueur est la somme des valeurs pour ce joueur des quintuples auxquels ce point appartient).

2.3.4. Écrire une fonction `initialiser` ne renvoyant pas de valeur et qui réalise l'initialisation complète du tableau de quintuples et d'une grille vide : la fonction recevra en paramètre la grille et le tableau de quintuples.

2.4. Analyser le grille

2.4.1. Choisir un point

Écrire le code d'une fonction `choisirPoint` recevant en paramètre le tableau `grille` et un caractère (O ou X) et renvoyant les coordonnées du point de la grille non occupé ayant la plus grande valeur pour le joueur utilisant le caractère donné (si plusieurs points ont cette valeur, la fonction renverra le premier rencontré au cours de la recherche).

2.4.2. Mettre à jour la grille

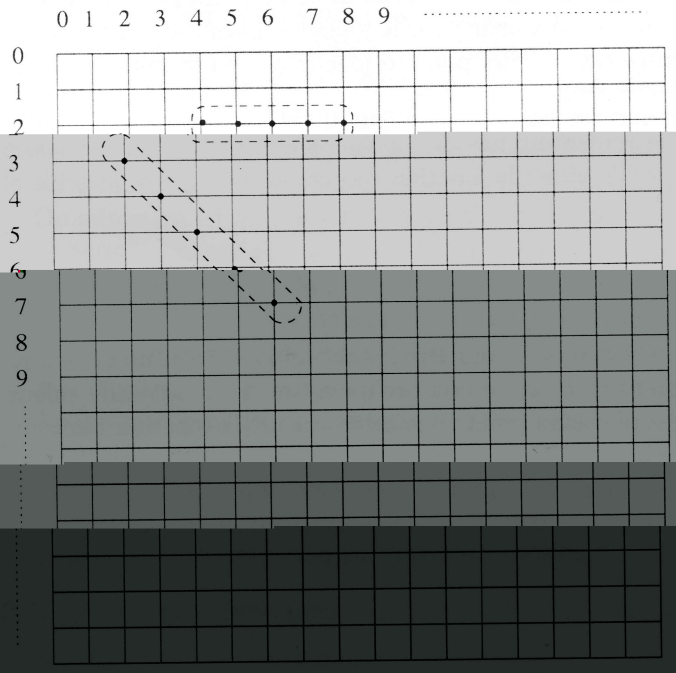
Écrire le code d'une fonction `jouer` renvoyant un entier et recevant en paramètre
- la grille et le tableau de quintuples,
- les coordonnées d'un point,
- un caractère,
- un pointeur `valeurQuintuple` sur une fonction ne renvoyant pas de valeur et ayant en parametres la grille et un pointeur sur un quintuple (l'effet de cette fonction est de recalculer et mettre à jour la valeur du quintuple vu par les deux joueurs),

La fonction `jouer` ne fait rien que renvoyer `0` si le point de coordonnées transmises n'est pas dans l'état `VIDE` et dans le cas contraire elle renvoie `1` avec les effets suivants :
- affecter à l'état du point de coordonnées données le caractère transmis ;
- pour chaque quintuple auquel le point appartient, mettre à jour sa valeur vue par chacun des joueurs) ;
- recalculer la valeur de tous les points pour chacun des joueurs.

2.4.3. Tester si un quintuple est gagnant

Écrire une fonction [illegible] en paramètre la grille et un pointeur sur

Annexe : la grille et deux exemples de quintuples



Quintuplet horizontal : ((2,4)(2,5),(2,6),(2,7),(2,8))

Quintuplet diagonal : ((3,2),(4,3),(5,4),(6,5),(7,6))