TP 5 Programmation – DUT 1 – Utilisation des structs (sans pointeur)

P.Courtieu

Résumé

On utilise des **`struct`** pour manipuler des coordonnées dans la grille de jeu, à la place de 2 entiers.

1 Rappels

`inout.c/h` : Maintenant vous savez utiliser `scanf` et `printf` (**`#include`**`<stdio.h>`), donc vous pouvez vous passer de `inout.c/h`. Néanmoins il est autorisé de s'en servir encore :

```
wget "http://deptinfo.cnam.fr/~courtiep/inout/inout.h"
wget "http://deptinfo.cnam.fr/~courtiep/inout/inout.c"
```

`BOOL` : Continuez à utiliser un pseudo type `BOOL` à la place de **`int`** :

```
#define BOOL int
#define TRUE 1
#define FALSE 0
```

`BOOL` toujours : Attention Ne faites ***jamais*** **`if`** `(xxx == TRUE)` mais **`if`** `(XXX)`. Symétriquement ne faites pas **`if`** `(xxx==FALSE)` mais **`if`** `(!XXX)`.

2 Othello (Reversi) utilisant un tableau à double entrée et `struct coordonnees`

Utilisez le squelette.

Le **`typedef`** suivant est défini :

```
typedef struct {
  int x;
  int y;
 } coordonnees ;
```

Programmez le jeu Othello (règles) avec un tableau de **`char`** à deux dimensions (8×8) en utilisant le type `coordonnees` partout où vous manipulez les positions dans la grille.

Avant de programmer le jeu proprement dit suivez les étapes et principe suivants :

— Choisissez si votre tableau à deux dimension représente un tableau de colonnes ou un tableau de lignes. Écrivez ce choix dans un commentaire au début de votre fichier.

— Dans votre programme les coordonnées démarrent à 0, seule la communication avec le joueur (messages et lecture au clavier) utilisent la numérotation à partir de 1.

— Définissez les fonctions suivantes

— `BOOL coordOK(coordonnees coord)` qui retourne `TRUE` si la coordonnées `coord` existe dans la grille othello, et `FALSE` sinon.

— **char** `get(char t [][8],coordonnees coord)` qui retourne le char de la case de `t` à la coordonnée `coord`. Attention il faut appeler cette fonction sur une coordonnées qui existe. Dans le cas contraire la fonction retourne n'importe quoi et peut même faire planter le programme.
— `BOOL estChar(char t [][8],coordonnees coord,char c)` retourne `TRUE` si la case `coord` (supposée valide comme pour `get`) du tableau `t` contient un caractère `c`.
— `coordonnees demandeCaseVide(char t[][8])` qui demande les coordonnées d'une case de `t` vide à l'utilisateur. Si l'utilisateur donne une case non vide ou inexistante, on la lui redemande.
ATTENTION : l'utilisateur utilise une numérotation des lignes et colonnes commençant à 1 mais la coordonnées retournée doit avoir été traduite dans la numérotation démarrant à 0.
— **void** `set(chat t [][8, coordonnees coord, char c)` qui remplace le **char** de la case `coord` du tableau `t` par le caractère `c`.
— `coordonnees voisinG(coordonnees coord)` qui retourne la coordonnées voisine à gauche de `coord`.
— idem `voisinD`, `voisinH`, `voisinB` pour les coordonnées voisines droite, haut et bas.

Les fonctions de voisinage sont en fait des fonction d'incrément. Elles vous permettrons d'écrire simplement des boucles utiles pour othello, par exemple :

```
coordonnee c = ...;
// Effectue un traitement sur chaque case à gauche de c qui contient
// un 'X', s'arrête à la première case différente ou inexistante:
while (coordOK(c) && estChar(t,c,'X')) {
 ... // traitement
 c = voisinG(c)); // incrément
}
```