Novembre 2004

Examen de Décembre 2003 – Puissance 4

Adaptation par Laure Gonnord
`http://www-verimag.imag.fr/~gonnord/`

À tout moment, toute procédure ou fonction d'une question précédente, pourra être utilisée même si elle n'a pas été écrite.

On désire ici réaliser un programme qui permet à l'utilisateur de jouer au jeu puissance 4. Ce jeu est un jeu à deux joueurs et est décrit ci-dessous. L'un des deux joueurs sera l'ordinateur, l'autre sera l'utilisateur du programme.

Règles du jeu : Le jeu puissance 4 est constitué de 7 tiges verticales alignées et de même taille. Sur chaque tige, on peut enfiler au maximum 5 boules. Ce jeu se joue à 2. Une personne prend les boules blanches et l'autre les boules noires. Chacun son tour, chaque joueur enfile une boule de sa couleur sur une tige. Toute boule insérée descend le long de la tige jusqu'en bas ou sur une autre boule placée avant. Le but de chaque joueur est d'aligner 4 boules de sa couleur, soit horizontalement, soit verticalement, soit diagonalement, sans qu'il n'y ait ni vide, ni boule d'une autre couleur intercalée. Le jeu s'arrête dès qu'un alignement d'au moins quatre boules a été réalisé, ou lorsque toutes les cases sont occupées par une boule.

Pendant toute la suite du problème, on considère que l'ordinateur joue avec les boules de couleur blanche et l'utilisateur joue avec les noires.

Voici les structures de données utilisées par ce programme au niveau global. Ces constantes et variables sont donc accessibles depuis toutes les fonctions et procédures :

```
# define L 5            /* nombre de lignes */
# define C 7            /* nombre de colonnes */
int Jeu[L][C];
int LigneJouable[C];
int Gagnant;
```

- `int Jeu[L][C];` On utilise un tableau d'entiers de 2 dimensions pour représenter ce jeu. La ligne 0 de ce tableau représente la ligne basse du jeu, la ligne 1 la ligne immédiatement supérieure, ... , la ligne 4 la ligne au sommet du jeu. Les 7 colonnes (0, 1, ... , 6) représentent les 7 tiges du jeu de gauche à droite. On fait le choix de mettre une case de ce tableau à 0 si la position correspondante du jeu ne contient pas de boule, de mettre 1 si la position correspondante du jeu contient une boule blanche et 2 si elle contient une boule noire. Au début du jeu, chaque case de ce tableau contient la valeur 0.
- `int LigneJouable[C];` La valeur de la case i de ce tableau contient le numéro de la ligne qui serait occupée, si l'on enfilait une boule en tige i (où $i \in \{0, ..., 6\}$). La valeur 5 signifie que la tige ne peut être jouée. Au début du jeu, chaque case de ce tableau contient la valeur 0. À chaque coup joué, ce tableau est actualisé.
- `int Gagnant;` Par convention, cette variable vaut 2 si l'utilisateur gagne, 1 si l'ordinateur gagne et 0 si personne n'a encore aligné 4 boules.

En début de programme principal, on initialise à 0 toutes les cases du tableau `Jeu`, toutes les cases du tableau `LigneJouable` et la variable `Gagnant`.

Question 1

Pour effectuer un affichage simple, on peut se contenter des valeurs 0, 1, 2 du tableau `Jeu`. Écrire la procédure `Affichage`, sans paramètre, de telle sorte que les valeurs du tableau `Jeu` apparaissent à l'écran sous forme de tableau à deux dimensions. Les lignes sont affichées dans l'ordre décroissant ; la ligne 0 est la plus basse. L'en-tête de cette procédure est : `void Affichage(void);`

Exemple 1 *Quelle situation du jeu représente ce tableau ?*

```
0  0  2  0  0  0  0
0  0  1  0  0  0  0
2  0  1  0  0  2  0
1  0  1  2  1  2  0
1  2  2  1  2  1  2
```

Question 2

La fonction `FinDePartie` retourne `vrai(1)` si toutes les cases du tableau `Jeu` sont remplies et faux sinon. Celle-ci vérifie qu'il existe au moins une colonne jouable avec l'aide du tableau `LigneJouable`. Pour cela, il suffit donc de chercher dans le tableau `LigneJouable` la première case inférieure ou égale à 4. Écrire la fonction `FinDePartie` dont l'en-tête est : `int FinDePartie(void);`

Question 3

Supposons que l'on joue une boule de couleur (désignée par la variable) `couleur` dans la colonne `colonne`, alors cette boule occuperait la case `Jeu[LigneJouable[colonne]][colonne]`. La fonction `AligneVertical` retourne le nombre de boules de couleur couleur figurant dans l'alignement vertical de cette nouvelle boule, incluant cette dernière. Le tableau Jeu ne doit pas être modifié. On suppose de plus que `LigneJouable[colonne]` $< L$.

Écrire cette fonction `AligneVertical` qui a pour paramètre un numéro de colonne et la couleur du joueur (1 si l'ordinateur joue, 2 si l'utilisateur joue). L'en-tête de celle-ci est :
`int AligneVertical (int colonne, int couleur);`

Exemple 2 *Si la situation courante du jeu est représentée par l'ex1, la fonction* `AligneVertical` *appelée pour colonne = 5, couleur = 1, retourne 1. Cette même fonction appelée pour colonne = 5, couleur = 2, retourne 3. Cette même fonction appelée pour colonne = 0, couleur = 1, retourne 1.*

Question 4

Écrire une fonction `AligneHorizontal`. Comme ci-dessus cette fonction a pour paramètre un numéro de colonne et la couleur de la boule, la case désignée est une case jouable. Comme dans la fonction précédente, on suppose que l'on joue une boule de couleur couleur dans la colonne colonne. Cette fonction fournit le nombre de boules de couleur figurant dans l'alignement horizontal de la nouvelle boule, incluant cette dernière. L'en-tête de cette fonction est : `int AligneHorizontal (int colonne, int couleur);`

Question 5

Même fonction que les deux précédentes, mais ici on demande de compter les boules alignées diagonalement avec la case désignée. Il y a deux diagonales, on devrait écrire deux fonctions : `AlignerDiagonal1` et `AlignerDiagonal2`. Ces deux fonctions ont de grandes similarités, on demande d'écrire totalement l'une de ces deux fonctions. On supposera dans la suite que la deuxième fonction est écrite.

Exemple 3 *Si l'on considère que le jeu est dans l'état représenté par l'ex 1, la fonction* `AligneDiagonal1` *appelée pour colonne = 3, couleur = 1, retourne la valeur 4.*

Question 6

Écrire une fonction `AligneMax`. Cette fonction utilise deux paramètres : un numéro de colonne qui permet de désigner la case où l'on envisage de jouer et une couleur qui donne la couleur de la boule que l'on envisage d'y placer. Cette fonction retourne la longueur du plus grand alignement (horizontal, vertical ou diagonal) contenant cette boule.

Question 7

Écrire une fonction `CoupGagnant`. Cette fonction utilise deux paramètres : un numéro de colonne qui permet de désigner la case où l'on envisage de jouer et une couleur qui donne la couleur de la boule que l'on envisage d'y placer. Cette fonction retourne `vrai` (1) si le coup envisagé est gagnant et `faux` (0) sinon.

Question 8

Écrire une procédure `JeuUtilisateur`, d'en-tête : `void JeuUtilisateur(void);` Cette procédure suppose que le joueur (utilisateur) peut jouer. Elle demande au joueur (il joue avec la couleur 2) dans quelle colonne il veut jouer. Elle vérifie qu'il peut effectivement jouer dans la colonne désignée, sinon, elle lui redemande une réponse jusqu'à ce que la réponse soit correcte, elle modifie alors les tableaux `Jeu` et `LigneJouable` en conséquence, elle met éventuellement à jour la variable `Gagnant`.

Question 9

Quand c'est au tour de l'ordinateur de jouer, celui-ci évalue puis donne une note aux différents coups possibles (au plus un par tige, donc au plus 7). Ces notes sont calculées puis stockées dans un tableau `Note`. L'ordinateur jouera ensuite le coup ayant obtenu la meilleure note.

On demande d'écrire une fonction `Notation` respectant l'en-tête : `void Notation(int Note[C]);` qui calcule toutes les valeurs du tableau `Note` passé en paramètre. Cette procédure utilise l'algorithme suivant : Si la colonne col ne peut être jouée sa note `N[col]` est -1. Si la colonne peut être jouée et qu'il s'agit d'un coup gagnant de l'ordinateur (couleur 1) la note attribuée est 100. Si la colonne n'est pas gagnante pour l'ordinateur mais gagnante pour l'utilisateur (couleur 2) sa note est 50. Dans tous les autres cas, la note est la longueur du plus grand alignement de l'ordinateur passant par cette position.

Question 10

Écrire une procédure `JeuOrdinateur`, dont l'en-tête est : `void JeuOrdinateur(void);` Celle-ci calcule le tableau `Note`, qui est local à la procédure. Puis décide de la colonne où jouer, celle qui a la meilleure note. Elle précise la colonne sélectionnée à l'utilisateur, modifie en conséquence les tableaux `Jeu` et `LigneJouable`, et met à jour éventuellement la variable `Gagnant`. On rappelle que l'ordinateur joue avec la couleur 1.

Question 11

Écrire la fonction `main` suivant le modèle ci-dessous. On demande la totalité du programme, mais l'on ne recopiera que les en-têtes des procédures et fonctions.

```
Initialisation des variables.
/* Pour simplifier, on suppose que c'est toujours l'utilisateur qui commence. */
Tant qu'il n'y a pas de gagnant et qu'il est possible de jouer faire
        Si c'est au tour de l'utilisateur le faire jouer sinon faire jouer l'ordinateur.
        Afficher l'état du jeu.
        Changer de joueur.
S'il y a un gagnant alors afficher si c'est l'ordinateur ou
l'utilisateur, sinon dire que la partie est nulle.
```