

TP2 C++ – M2 ISIFAR

Pointeurs, tableaux, ...

M. de Falco [defalco@pps.jussieu.fr] – Site de Chevaleret, Bureau 5A16

6 octobre 2009

Préliminaires

Rappels sur les tableaux

On rappelle qu'un tableau d'entiers de 10 éléments se déclare ainsi :

```
int tableau[10];
```

On accède au *i*ème élément avec `tableau[i]`.

Pour déclarer une fonction prenant un tel tableau en argument on écrira

```
retType fonction(int tableau[10])
```

Le nombre d'élément est alors optionnel et on peut écrire juste `tableau[]`, d'ailleurs il n'y aucune erreur si la taille ne correspond pas lorsque l'on passe un tableau. Les éléments du tableau sont alors modifiables (on a passé un pointeur).

On peut définir des tableaux multi-dimensionnels. Par exemple,

```
int matrice[3][4];
```

représente une matrice d'entiers à 3 lignes et 4 colonnes. Pour obtenir l'élément situé sur la *i*-ème ligne et *j*-ème colonne on écrit `matrice[i][j]`.

Pour passer un tel tableau en argument on utilisera

```
retType fonction(int matrice[3][4])
```

là c'est le premier nombre de dimension, 3, qui est optionnel et seulement celui-ci ! L'exemple suivant produit une erreur à la compilation :

```
void f(int t[][2]) {}  
int t[2][3];  
f(t);
```

Nombres aléatoires

Pour pouvoir tirer un petit nombre aléatoirement on inclue le `<stdlib.h>` et on utilise `int rand()` qui renvoie un nombre aléatoire dans la plage de définition du type `int` (*on verra dans les tps suivant qu'il faut également initialiser les générateurs aléatoires et que `rand` n'est pas le meilleur d'entre eux.*)

1 Jeux de plateau

1.1 Introduction

Dans ce TP on se propose de réaliser un programme pour jouer à des jeux de plateau. Dans un premier temps on s'intéressera au jeu du morpion, encore appelé Tic-Tac-Toe puis on adaptera notre programme pour le jeu Othello. Ce TP sera poursuivi dans un TP ultérieur, pensez à sauvegarder vos fichiers ou à me demander de le faire.

Un plateau de $n \times n$ cases sera représenté par une variable globale, i.e. définie avant les fonctions au début du fichier,

```
int board[n][n];
```

Les éléments sur le plateau ne seront que de trois types : case vide, point du premier joueur et point du second joueur. On définit alors trois variables globales

```
int NO_PLAYER = 0;
int PLAYER_1 = 1;
int PLAYER_2 = 2;
```

1.2 Tic-Tac-Toe

Tic-Tac-Toe se joue sur un plateau de 3×3 cases. Chaque joueur à tour de rôle pose un pion sur une case vide, le premier joueur à avoir trois pions alignés horizontalement, verticalement ou en diagonale gagne.

Si vous avez une imprécision sur l'écriture des fonctions suivantes, reportez-vous à l'exemple de session à la section 1.2.1.

1. Ecrivez une fonction

```
void initBoard();
```

qui remplit le plateau de cases vides (NO_PLAYER).

2. Ecrivez des fonctions

```
bool sameValueOnRow(int r, int *value);
bool sameValueOnColumn(int c, int *value);
bool sameValueFirstDiagonal(int *value);
bool sameValueSecondDiagonal(int *value);
```

qui vérifie si le plateau contient trois éléments identiques sur la ligne *r*, la colonne *c*, la première ou la deuxième diagonale. La fonction renvoie `true` si c'est le cas et met la valeur à l'adresse pointée par *value*, `false` sinon.

3. Ecrivez une fonction

```
bool isWinning(int *winner,
               int currentPlayer);
```

qui renvoie `true` si le plateau est gagnant alors que c'est au joueur `currentPlayer` de jouer. Le joueur gagnant est alors mis à l'adresse pointée par `winner`.

Lorsqu'il y a égalité on considère que la partie est gagnante et que le gagnant est NO_PLAYER.

4. Ecrivez une fonction

```
void validMoves(bool moves[][3],
                int currentPlayer);
```

qui met `moves[i][j]` à `true` si le joueur `currentPlayer` peut jouer à cette position, `false` sinon.

5. Ecrivez une fonction

```
void playMove(int i, int j, int forPlayer);
```

qui joue le coup en (i,j) pour le joueur `forPlayer`. On supposera le coup valide.

6. Ecrivez des fonctions

```
int numberOfMoves(bool moves[][3]);
void getNthMove(bool moves[][3],
                int n, int *i, int *j);
```

qui renvoie respectivement le nombre de coups valides et le *n*-ème coup valide (dont les coordonnées sont passées dans *i* et *j*).

7. Ecrivez des fonctions

```
void displayBoard();
void displayBoardWithMoves(bool moves[][3]);
```

qui affichent le plateau avec un espace pour une case vide, un X pour le premier joueur et un O pour le second. Dans le second cas on affichera également le numéro *i* pour le *i*ème coup valide.

8. Ecrivez une fonction

```
void requestMove(bool moves[3][3],
                 int *i, int *j, int forPlayer);
```

qui demande à l'utilisateur jouant le joueur `forPlayer` d'entrer un coup à jouer, les coordonnées du coup sont placées en *i* et *j*. On utilisera la fonction d'affichage du plateau précédente pour proposer les coups valides.

9. Ecrivez une fonction

```
void randomMove(bool moves[][3],
                int *i, int *j);
```

qui choisit un coup aléatoire parmi les coups possibles.

10. On va maintenant recoller tous les morceaux pour faire le programme complet. A titre d'indication un graphe de flot correspondant au coeur du programme est donné à la figure 1.

11. Inspirez vous du programme précédent pour réaliser un programme qui demande un nombre de parties à simuler et renvoie la proportion de parties gagnées par le premier joueur, le second ou finissant sur une égalité.

1.2.1 Exemple de session

```
Welcome to TicTacToe
Should Player 1 be human ? (0 for no, 1 for yes) |0|1|2|
1
Should Player 2 be human ? (0 for no, 1 for yes) 0| | | |
0
|0|1|2|
-----
0| | | |
-----
1| | | |
-----
2| | | |
-----
Player 1 please select a move between
|0|1|2|
-----
0|0|1|2|
-----
1|3|4|5|
-----
2|6|7|8|
-----
Number of selected move: 4
|0|1|2|
-----
0| | | |
-----
1| |X| |
-----
2| | | |

-----
1| |X| |
-----
2| | | |

-----
1| |X| |
-----
2| | | |

-----
Player 1 please select a move between
|0|1|2|
-----
0|0|1|2|
-----
1|3|X|4|
-----
2|5|6|0|
-----
Number of selected move:
[...]
|0|1|2|
-----
0|X|X|X|
-----
1|0|X| |
-----
2| |0|0|
-----
Player 1 has won
```

1.3 Othello

Othello est un jeu qui se joue sur un plateau 8×8 . Comme au Tic-Tac-Toe chaque joueur pose un pion à tour de rôle. Les coups admissibles sont ceux qui placent un pion de telle sorte qu'il encadrent un nombre non nul de pions de l'adversaire dans au moins un des huit directions autour de lui.

Par exemple si une partie de la configuration est `XXX0` le coup consistant à rajouter un pion `0` sur la gauche est admissible. Lorsqu'on effectue ce coup on capture tous les pions ainsi encadrés (dans l'exemple on obtient `OXXX0 -> 00000`).

Lorsqu'un joueur ne peut pas jouer il passe son tour. Lorsque les deux joueurs passent successivement leurs tour, la partie est finie. Le gagnant est le joueur ayant le plus de pions sur le plateau.

Exercice : **Sans effacer votre code précédent** (en ouvrant un nouveau projet par exemple) modifiez-le pour pouvoir jouer à Othello.

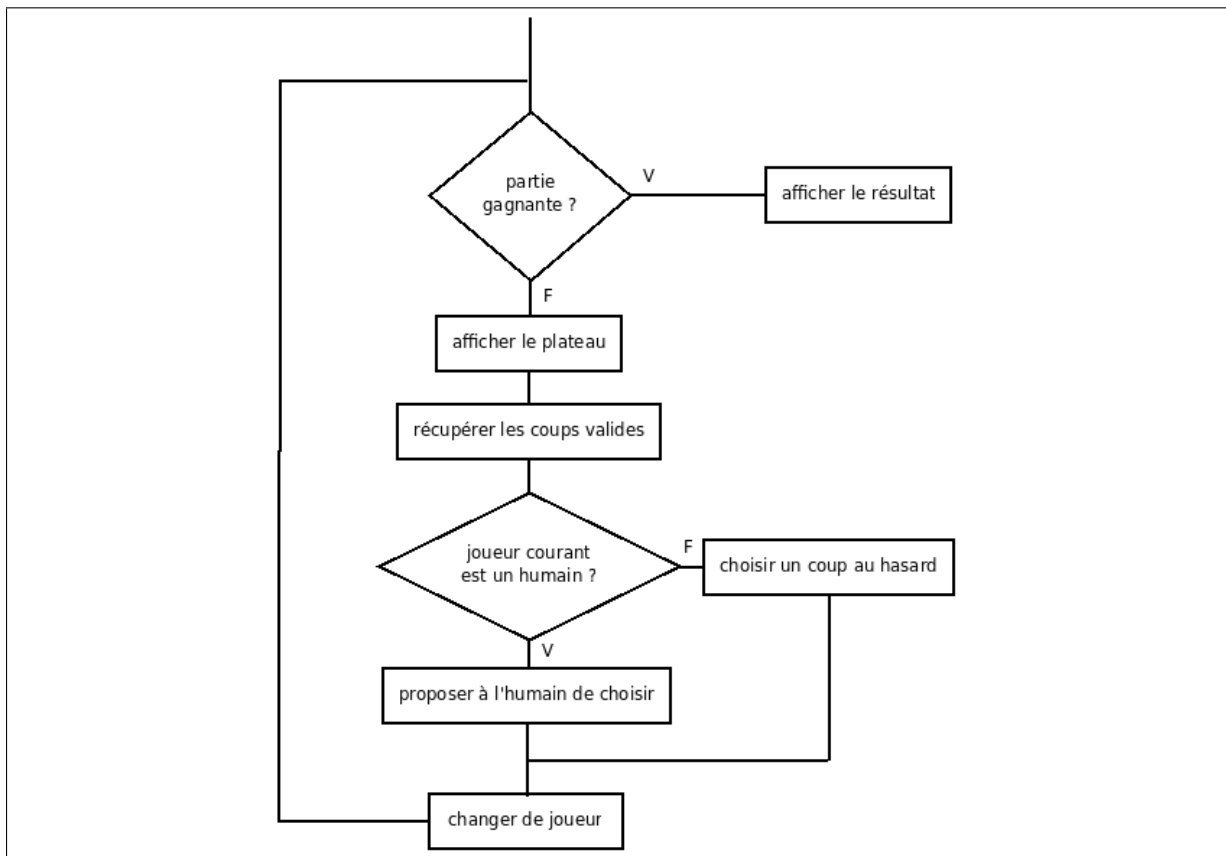


Figure 1: Graphe de flot du programme TicTacToe