TP 3 Programmation – DUT 1 – Tableaux et boucles

P.Courtieu

Septembre 2017

Résumé

On écrit des boucle **`while`** et on utilise des tableaux pour programmer le morpion de manière plus simple et plus souple.

1 Rappels

La bibliothèque d'entrée-sortie : inout.h et inout.c et mettez les dans un répertoire `tp2`. Pour cela vous pouvez faire les commandes suivantes (une seule fois) dans un terminal (menu pricipal : `terminal/Konsole`) :

```
mkdir tp2 # si tp2 pas deja cree
cd tp2
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)`.

Programmez et tester ces fonctions *une par une* (testez une fonction dès que vous pensez qu'elle est finie). Pour tester une fonction `f` on programme un ou plusieurs appels à cette procédure dans une procédure `test_f` de la manière vue en cours : on test le résultat obtenu par rapport au résultat attendu et on affiche un message d'erreur si ils ne sont pas égaux.

2 Fin TP 2

Ceux qui n'ont pas fini le TP 2 (le morpion sans test de victoire), finissez le rapidement.

3 Morpion avec un tableau fixe de taille 9

Dans un nouveau fichier, reprogrammez le morpion en changeant votre structure de donnée : enlevez les 9 variable `c1`...`c9` et remplacez les par un tableau de 9 caractères.

```
char t[] = { '_' , '_' , '_' , '_' , '_' , '_' , '_' , '_' , '_' };
```

Cette ligne crée un tableau de 9 caractère et initialise le tableau avec les caractère écrit entre les accolades. Le nombre de caractères donnés détermine la taille du tableau.

Remarquez que si vous

1. remplacez `c1` par `c[0]`...`c9` par `c[8]` dans le programme principal (fonction `main`) uniquement. Votre programme devrait fonctionner à nouveau.
2. Remplacez la fonction`afficheGrille(char a1, char a2 ..., char a9)` par la fonction qui prend seulement un tableau de caractères en paramètre :

```
void afficheGrille(char tc[]){
  ...
}
```

3. Simplifiez le code de votre fonction `main`, en effet les 9 cas traités séparément peuvent maintenant être traité en un seul **`if`** sur le caractère tapé par l'utilisateur. Il faut tester si l'entier correspond bien à une case et si cette case est vide.

4 Morpion à taille variable

Il s'agit maintenant de faire évoluer votre code (dans le `main` et dans `afficheGrille`) pour supporter plusieurs tailles de grille. La taille de la grille sera fixée avant de compiler en modifiant la valeur d'une variable (`largeurGrille` voir ci-dessous) au début du main.

1. Changez la déclaration du tableau `t` de la manière suivante (il se peut que l'option `-std=c99` soit nécessaire) :

```
const int largeurGrille = 4;
char t[largeurGrille*largeurGrille];
```

   L'idée est que pour changer de taille de grille il vous suffit de modifier le « `4` » ci-dessous par une autre taille pour que tout marche correctement. Il n'est pas demandé de programmer le test de victoire (mais vous pouvez essayé si tout le reste est fini).
2. Reprogrammez l'initialisation de la grille (toutes les cases mises à `'_'`) pour qu'elle s'adapte à la largeur `largeurGrille`.
3. Ajouter un argument **`int`** `larg` à la procédure `afficheGrille`. Cet argument représente le nombre de lignes (et de colonnes) de la grille. Adaptez le code de la procédure pour qu'elle s'adapte à la largeur passée en paramètre.
   Remarques : Voir dans l'appendice comment relier la position dans la grille et le numéro de la case dans le tableau.

4.1 Amélioration

1. Définissez une fonction `lireCoupMorpion` de lecture d'un entier au clavier qui redemande un coup tant que celui-ci est invalide. Quels arguments doit prendre cette fonction ? Que doit-elle retourner ?
   Pour redemander la lecture vous pouvez soit utiliser une boucle **`while`** (voir ci-dessous) soit simplement rappeler la fonction `lireCoupMorpion`.
2. Programmez la fonction `BOOL testpasGagne(char tc[], int larg){ .. }` qui retouren `FALSE` si le tableau `tc`, de taille `larg`×`larg`, contient une grille de morpion dans laquelle un joueur a gagné (3 cases aligné avec la même coche, et `TRUE` sinon.

4.2 Exercice Supplémentaire : Puissance 4

Si, et seulement si, vous avez programmé toutes les améliorations ci-dessus (morpion à taille variable + test de victoire, protection contre les coups invalides), programmez le puissance 4 en utilisant le même genre de technique (tableau de taille 42 puis variable) ou un tableau à 2 dimensions si vous connaissez déjà cette technique.

A Traduction position dans la grill (x,y) $\Rightarrow$ position dans le tableau (i)

Considérons la grille 4x4 (`larg=4`) suivante, numérotée pour l'utilisateur (à partir de 1) et la case (4,3) (4^e colonne, 3^e ligne) :

2 lignes complètes

	1	2	3	4
1	'_'	'x'	'_'	'o'
2	'x'	'o'	'_'	'_'
3	'_'	'x'	'_'	'o'
4	'x'	'o'	'_'	'_'

Cette grille est représentée en mémoire par un tableau à 16 cases :

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
_	'x'	'_'	'o'	'x'	'o'	'_'	'_'	'_'	'x'	'_'	'o'	'x'	'o'	'_'	'_'

la 4^e case de la 3^e ligne se situe dans le tableau à la case $(2 \times \texttt{larg} + 3)$. Plus généralement la case (n, m) (colonne n , ligne m) se situe à la case $(n-1) \times \texttt{larg} + m - 1$.