TP 4 Programmation – DUT 1 – boucles imbriquées et tableau à double entrées

P.Courtieu

Septembre-octobre 2017

Résumé

On écrit des boucles **`while`** imbriquées et on utilise des tableaux de chaînes de caractères

1 Rappels

`inout.c/h :`

```
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 Programmer le jeu de puissance 4 avec un tableau à double entrée

Si vous ne l'avez pas déjà fait, programmez le morpion avec une grille 10x10 représentée en mémoire par un tableau de caractères à double entrée. Si vous l'avez déjà fait ou si vous préférez vous pouvez faire les exercices de la section suivante.

3 Exercices sur les tableaux de chaînes de caractères.

Pour cet exercice il vous faut télécharger la dernière version de `inout.h` et `inout.c`.

3.1 La fonction de lecture dans un fichier, documentation

Vous disposez de la fonctions

```
char ** lireFichierParMots(char * nomFicher, int* nombreMots);
```

Cette fonction se comporte de la manière suivante :

— Elle prend en paramètre

— un nom de fichier (chaîne de caractère) `nomFicher`
— *l'adresse* `nombreMots` d'un entier . Vous pouvez lui passer un tableau d'entier de taille 1, ou bien vous pouvez lui passer une variable entière en faisant précéder son nom de l'opérateur `&` qui signifie « adresse de ».

— Elle effectue deux choses :
— Elle retourne un tableau de chaînes de caractères (**`char`** `**`) qui contient tous les mots contenus dans le fichiers.
— Elle *modifie* l'entier passé en paramètre (plus exactement : dont l'adresse est passé en paramètre) afin qu'il contienne la taille du tableau retourné.

Autrement dit un appel possible à la fonction pourrait être (en utilisant un tableau d'entier de taille 1 pour le deuxième paramètre) :

```
int x[1];
char ** t = lireFichierParMots("toto.txt",x);
```

Après cet appel, `t` est un tableau contenant tous les mots du fichier `"toto.txt"`, et `x[0]` contient la taille de ce tableau.

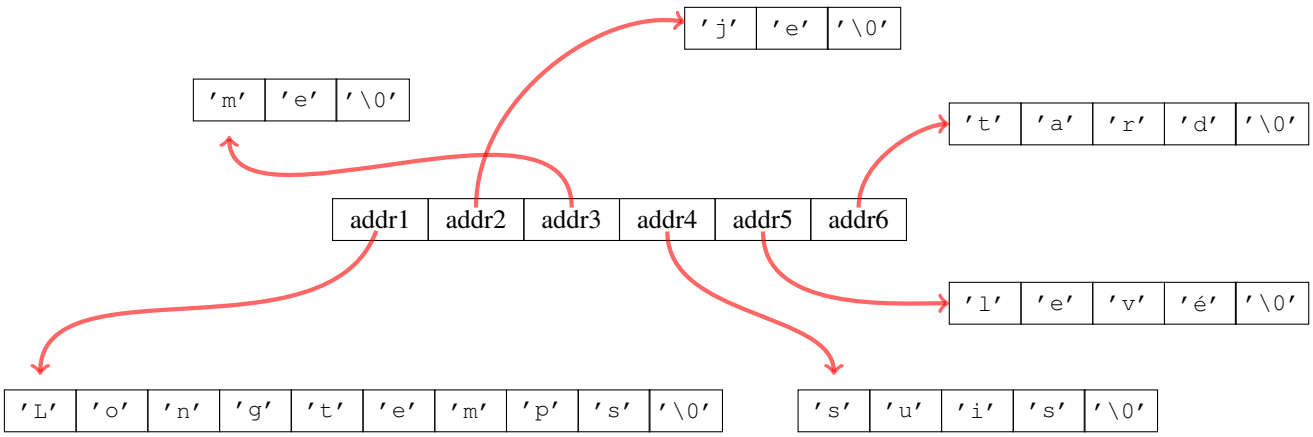
Autre exemple en utilisant l'opérateur `&x`, on déclare un entier `x` cette fois :

```
int x;
char ** t = lireFichierParMots("toto.txt",&x);
```

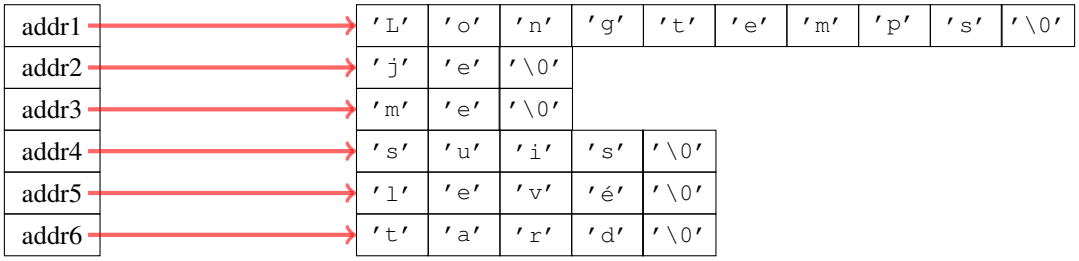
Après cet appel, `t` est un tableau contenant tous les mots du fichier `"toto.txt"`, et `x` contient la taille de ce tableau.

3.2 Les tableaux de chaînes

En mémoire un tableau de chaine de caractère (**`char`** `**t` ou **`char`** `*t[]`) est un tableau dont chaque case est l'adresse d'un tableau de caractères (chacun ayant un dernier caractère `'\0'`).



Pour se simplifier les idées on se le représente bien « rangé » :



Pour accéder aux lettres des mots on utilise la même syntaxe que pour les tableaux à double entrée (mais il doit être évident maintenant que ça ne fait pas la même chose) :

```
char c = t[3][1];  // donne le caractère u de "suis"
```

Cette fois les deux indice doivent être compris comme ceci : `t[3]` est un tableau de caractère, on peut donc demander sa case n^{o}1.

3.3 Exercices

Implantez les différentes fonctions du squelette fourni sur le site du cours.

Le squelette permet de passer un nom de fichier en paramètre du programme comme ceci (en admettant que vous avez appelé l'exécutable `tp4`) :

```
tp4 toto.txt
```

3.4 Le jeu du pendu

En utilisant ce que vous savez sur les chaîne, programmez le jeu du pendu.

Le jeu du pendu consiste à deviner un mot (dont on connaît la taille au départ) avec le moins de *coups* possible. Une tentative consiste à demander si une lettre apparaît dans le mot à deviner. Si oui l'arbitre (ici le programme) doit donner toutes les positions auxquelles la lettre apparaît. Après chaque *coup* le joueur peut proposer un mot. Attention si la proposition est fausse il perd instantanément.

Le mot à deviner devra être tiré au hasard dans un fichier `Liste_mots.txt` obtenu avec cette commande :

```
wget "http://deptinfo.cnam.fr/Enseignement/CycleA/APA/nfa031/docs/dico_nfa031.txt"
```

Les mots n'ont pas d'accent et sont en majuscule, pour simplifier le jeu se joue donc entièrement en majuscule. Voici un exemple de déroulé d'un partie que doit permettre votre programme :

```
------
Quelles lettre proposez vous?
E
E---E-
Quelles lettre proposez vous?
L
E---E-
Quelles lettre proposez vous?
p
e---e-
Quelles lettre proposez vous?
M
E--ME-
Quelles lettre proposez vous?
X
EX-ME-
Quelles lettre proposez vous?
N
EX-MEN
Quelles lettre proposez vous?
EXAMEN
Gagné en 6 coups!
```

Voici un squelette de programme qui commence par tirer un mot au hasard dans le fichiers `dico_nfa031.txt` (que vous aurez préalablement téléchargé (voir ci-dessus) :

```
#include"inout.h"
#include<time.h>
#include<stdlib.h>
#define BOOL int
#define TRUE 1
#define FALSE 0
int main() {
  int nbMots;
  int nombre_aleatoire = 0;
  srand(time(NULL)); // initialisation de rand
  char **mot = lireFichierParMots("dico_nfa031.txt",&nbMots);
  printf("mot 0 : %s\n",mot[0]);
  printf("mot 1 : %s\n",mot[99957]); /* le dernier */
  nombre_aleatoire = rand()%99958; /* entre 0 et 99957 */
  // Plus généralement pour tirer dans l'intervale [a,b[ (b exclu
  // donc) on fait:
  // nombre_aleatoire = rand()%(99958-a)+a; /* entre a et b-1 */
  printf("mot %d : %s\n",nombre_aleatoire,mot[nombre_aleatoire]); /* le dernier */
  // Voilà le mot est choisi vous pouvez programmer le pendu ci-dessous.

}
```