Algo/JavaScript – USAL34

TP4 Passage de paramètres et fonctions

Objectifs du TP

- Apprendre à découper un programme en plusieurs fonctions ou procédures.
- Apprendre à manipuler les tableaux de caractères à une dimension
- Apprendre à manipuler les tableaux de caractères à deux dimensions
- Apprendre à utiliser les structures de contrôles itératives (boucles `while`)

1 Rappels : bibliothèque et compilation

La bibliothèque d'entrée-sortie : inout.h et inout.c mettez les dans un répertoire `tp4`. Pour cela vous pouvez faire les commandes suivantes (une seule fois) dans un terminal (menu pricipal : `terminal/Konsole`) :

```
mkdir tp4
cd tp4
wget "https://cedric.cnam.fr/~lamberta/enseignements/DSP/C/sources/inout.h"
wget "https://cedric.cnam.fr/~lamberta/enseignements/DSP/C/sources/inout.c"
```

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` : on test le résultat obtenu par rapport au résultat attendu et on affiche un message d'erreur si ils ne sont pas égaux.

Exercice 1 — *carré magique*

Un *carré magique* est une matrice carrée de taille $n \times n$ telle que la somme de chaque rangée, de chaque colonne et de chaque diagonale ait la même valeur. Un carré magique est dit *normal* s'il contient chaque entier compris entre 1 et n^2 exactement une fois. Par exemple, le tableau suivant est un carré magique normal :

$$\begin{bmatrix} 6 & 7 & 2 \\ 1 & 5 & 9 \\ 8 & 3 & 4 \end{bmatrix}$$

1. Écrivez deux fonctions `int ligne(int t[3][3], int i)` et `int colonne(int t[3][3], int j)` qui retournent la somme de la `i`-ème ligne (resp. de la `j`-ème colonne) du tableau `t` passé en paramètre. Écrivez une fonction `main` qui vous permette de tester ces fonctions.
2. Écrivez deux fonctions `int diagonale1(int t[3][3], int i)` et `int diagonale2(int t[3][3], int i)` qui retournent la somme de la diagonale majeure (resp. de la diagonale mineure) du tableau passé en paramètre. Écrivez une fonction `main` qui vous permette de tester ces fonctions.
3. Écrivez une fonction `BOOL magique(int t[3][3])` qui retourne `TRUE` si le tableau `t` est magique et `FALSE` sinon.
4. Écrivez une fonction `BOOL normal(int t[3][3])` qui retourne `TRUE` si le tableau `t` est normal et `FALSE` sinon.

2 Les tableaux de chaînes de caractères.

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);
```

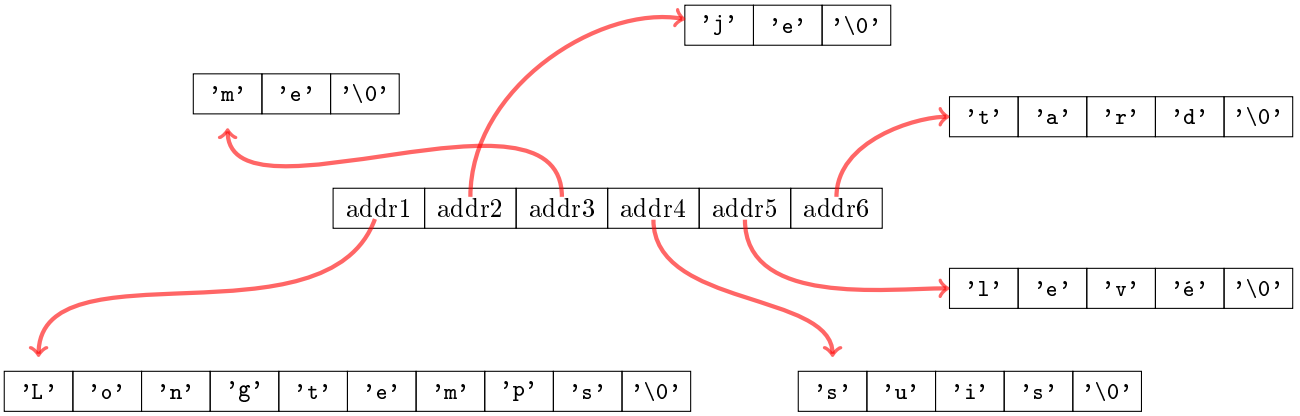
Après cet appel, `t` est un tableau contenant tous les mots du fichier `"toto.txt"`, et `x` contient la taille de ce tableau.

Vous pouvez télécharger un exemple de fichier `"toto.txt"` avec la commande suivante :

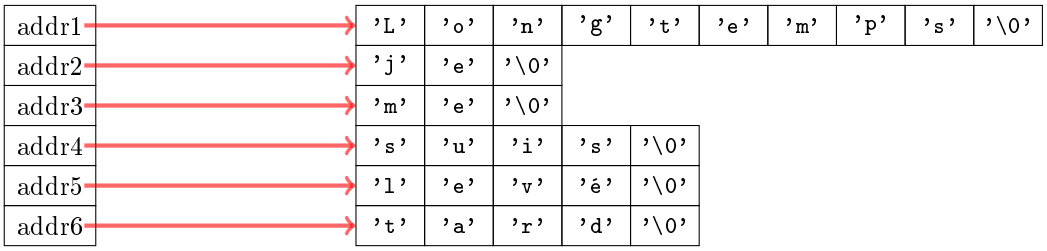
```
wget "https://cedric.cnam.fr/~lamberta/enseignements/DSP/C/sources/toto.txt"
```

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.

Exercice 2 — *Manipulation des chaînes de caractères* Implantez les différentes fonctions du squelette fourni sur le site du cours :

1. `tailleMax(char * t[], int taille)` qui retourne la taille du plus grand mot du tableau de mot `t`.
2. `affichageFiltreNbLettre(char *t[], int taille, int nbLettres)` qui affiche les mots de `t` dont la longueur est supérieure ou égale à `nbLettres`.
3. `affichageFiltreChar(char *t[], int taille, char c)` qui affiche les mots de `t` en remplaçant toutes les occurrences du caractère `c` par des espaces (' '). Bien entendu il est interdit de modifier les mots du tableau.
4. `compteLettre(char * t[], int taille , char c)` qui retourne le nombre d'occurrences du caractère `c` dans les mots de `t`.
5. `compteMotContenant(char * t[], int taille, char c)` qui retourne le nombre de chaines contenant le caractère `c` dans le tableau `t`.
6. `effaceChar(char *t[], int taille, char c)` qui modifie le tableau `t` en remplaçant toutes les occurrences du caractère `c` par un espace. Testez en affichant le tableau avant et après.
7. `renverseMots(char *t[], int taille)` qui modifie le tableau `t` en inversant l'ordre des lettres de chaque mots (ex : "levé" devient "ével").

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 La boucle `while`

Lorsqu'on veut répéter une séquence d'instructions plusieurs fois, on utilise une *boucle*. La boucle `while` ressemble à un `if` sans `else` :

```
while (cond) {
 ... // Corps de la boucle: séquence d'instruction à
     // effectuer ET À RECOMMENCER TANT QUE cond est vraie
}
```

Cette instruction s'exécute de la façon suivante :

1. la condition (`cond`) est évaluée
2. Si elle est fause, on sort immédiatement de l'instruction `while` (le corps de la boucle n'est pas exécutée)
3. Si elle est vraie on exécute le corps de la boucle

4. Puis on revient à l'étape 1. ci-dessus.

Notez que si le corps de la boucle ne modifie par la valeur de `cond` le programme va boucler indéfiniment.

Exemple :
Le programme suivant lit au clavier des nombres jusqu'à ce qu'un nombre négatif soit tapé.

```
int x = 0;
while(x>=0) {
  x = lireInt();
  ecrireString("vous avez tapé: ");
  ecrireInt(x);
  ecriresautDeLigne();
}
```

Exercice 3 — *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 "https://cedric.cnam.fr/~lamberta/enseignements/DSP/C/sources/dico_pendu.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!
```

Pour programmer le jeu du pendu on utilise dans la fonction `main` un tableau d'entier `estTrouve`, qui a la même taille que le mot cherché et qui contient dans le case `i`, la valeur `1` si la lettre `i` du mot à trouver a été trouvée et la valeur `0` sinon. Il faudra bien initialiser ce tableau avec des `0` au début du jeu.
On déclarera également un `BOOL jouer` qui est initialisé à `TRUE`, et qui sera mis à `FALSE` quand

le joueur aura gagné, pour pouvoir arrêter la partie.

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

1. `int nbOccurence(char t[], int a[], char c)` qui retourne le nombre d'occurences du caractère `c` du tableau de mot `t`. Cette fonction met à jour le tableau `a`, qui a la même taille que le tableau de mot `t` et qui contient dans le case `i`, la valeur `1` si la lettre `i` du mot à trouver a été trouvée et la valeur `0` sinon.
2. `void affichageMotACherche(char t[], int a[])` qui, en fonction des valeurs du tableau `a` affiche le mot `t` : si `a[i]=1`, elle affiche la lettre de `t`, sinon le caractère `-`.
3. `int main()` qui contient la suite d'instructions du jeu.