UE Algorithmes et programmation
L1/S2 – TD/TP n^o 6
chaînes de caractères, tri de tableau et structures
Année 2015

Vous pouvez récupérer du code ici :
`http ://lab-sticc.univ-brest.fr/~rodin/FTP/Enseignements/L1/AlgoEtProg`

Sans indication particulière, les fonctions ne doivent pas être écrites sous forme récursives. La récursivité ne sera utilisée que lorsque cela sera indiqué explicitement dans le sujet.

TD

1 TD Exercice 1 : petites fonctions de traitements de chaînes de caractères

Rappels sur les chaînes de caractères :

- En C, pour utiliser les fonctions sur les chaînes de caractères, il faut inclure le fichier `string.h`.
- `strlen` est une fonction renvoyant le nombre de caractères de la chaîne passée en argument.
- Pour obtenir ou modifier le caractère de la chaîne `str` se trouvant en une certaine position `i` (le (i+1)ème caractère), on utilise tout simplement `str[i]`.

```
#include <stdio.h>
#include <string.h> /* Pour strlen */

int main(void)
{
 char str[256] = "bonjour";
 int  lg = strlen(str);

 printf("%s a %d caracteres\n",str,lg); /* bonjour a 7 caracteres */
 str[0]='B';
 printf("%s a %d caracteres\n",str,lg); /* Bonjour a 7 caracteres */

 return 0;
}
```

Rappels sur les caractères : il est possible de tester si un caractère est égal à un autre (`==`), est différent d'un autre ! =, supérieur à un autre >, ou encore inférieur à un autre <. La relation d'ordre s'applique sur les codes ASCII des caractères.

Code ASCII	Type	Exemples
de 0 à 31, et 127	caractères non imprimables	'\n' :10
de 48 à 57	chiffres	'0' :48, '1' :49, ..., '9' :57
de 65 à 90	lettres majuscules	'A' :65, 'B' :66, ..., 'Z' :90
de 97 à 122	lettres minuscules	'a' :97, 'b' :98, ..., 'z' :122
autres	symboles	'*' :42, '<' :60, ...

1. Ecrire une fonction `palindrome` qui vérifie si une chaîne de caractères passée en paramètre est un palindrome... Cette fonction retournera `1` (vrai) si la chaîne est un palindrome, sinon, elle retournera `0` (faux). Rappel : pas de récursivité !

   Définition : un palindrome est une chaîne qui peut se lire aussi bien de droite à gauche que de gauche à droite... Ex : `hannah`
2. Ecrire une fonction `myStrcmp` qui compare deux chaînes passées en arguments, `s1` et `s2`. Cette fonction `myStrcmp` retourne un entier en fonction du résultat de la comparaison.
   Cet entier est égal à 0 si les chaînes `s1` et `s2` sont identiques.
   Cet entier est positif si la 1ère chaîne (`s1`) est supérieure à la 2ème chaîne (`s2`).
   Cet entier est négatif si la 1ère chaîne (`s1`) est inférieure à la 2ème chaîne (`s2`).

   ```
   myStrcmp("hannah","toto"); ---> -12                    ... 'h'-'t'--> -12
   myStrcmp("toto","hannah"); --->  12                    ... 't'-'h'-->  12
   myStrcmp("toto","toto");   --->   0                    ... 't'-'t'-->   0
   ```

   Rappel : pas de récursivité !... et **impossibilité** d'utiliser l'opérateur `==` sur les chaînes ⇒ comparaison sur des `char`.

2 TD Exercice 2 : tri de tableau par sélection

La méthode de tri par sélection d'un tableau, par exemple un tableau d'entiers, `int t[`n`] =` $\{e_0$; e_1 ; ... $e_{n-1}\}$ consiste à déterminer e le plus petit élément du tableau t et à le placer en tête du tableau. On recommence cette opération sur le tableau privé du premier élément, puis sur le tableau privé des deux premiers éléments, et ainsi de suite...

1. Ecrire une fonction `posMin` qui retourne la position du plus petit élément d'un tableau.

   Cette fonction prend comme paramètre un tableau `t` et la position `indiceDebutRecherche` à partir de laquelle il faut chercher.
2. Ecrire une fonction `triSelection` permettant de trier un tableau `t` passer en paramètre via la méthode du tri par sélection. Utiliser la fonction `posMin` et la fonction `echange` vue en cours.
3. Dérouler cet algorithme pout `int t[9] = {29, 6, 5, 1, 9, 7, 1, 0, 5} ;`
4. Quel est l'ordre de complexité de ce tri ?

1 TP Exercice 1 : tri rapide

L'algorithme de tri rapide, "quick sort" en anglais, est algorithme de type dichotomique. Son principe consiste à séparer l'ensemble des éléments en deux parties. La différence par rapport au tri fusion, vu en cours, est que la séparation des différentes valeurs ne s'effectue pas n'importe comment. Pour effectuer la séparation, une valeur pivot est choisie. Les valeurs sont réparties en deux ensembles suivant qu'elles sont plus grandes ou plus petites que le pivot. Ensuite, les deux ensembles sont triés séparément, suivant la même méthode. L'algorithme, tout comme le tri fusion, est récursif, mais cette fois, il n'est pas nécessaire de fusionner les deux ensembles. Le résultat du tri est égal au tri de l'ensemble dont les valeurs sont inférieures au pivot concaténé à l'ensemble des valeurs supérieures au pivot, ce dernier étant pris en sandwich entre les deux ensembles.

Le choix du pivot est le problème central de cet algorithme. En effet, l'idéal serait de pouvoir répartir les deux ensembles en deux parties de taille à peu près égales. Cependant, la recherche du pivot qui permettrait une partition parfaite de l'ensemble en deux parties égales aurait un coût trop important. C'est pour cela que le pivot est choisit de façon aléatoire parmi les valeurs de l'ensemble. Dans la pratique, le pivot est le premier ou le dernier élément de l'ensemble à fractionner.

1. Ecrire une fonction `partition` prenant 3 paramètres : un tableau `tab`, un indice de `depart` d'un sous-tableau de `tab` et un indice d'`arrivee` de ce même sous-tableau de `tab`. Cette fonction doit retourner un entier.

   Cette fonction `partition` place le 1er élément (le pivot) du sous-tableau à sa place dans le sous-tableau. C'est-à-dire qu'après la partition, tous les éléments à gauche du pivot sont plus petits que le pivot et, tous les éléments à droite du pivot sont plus grands que le pivot. Utiliser la fonction `echange` vue en cours.

   L'entier retourné par la fonction `partition` correspond à la place où a été placé le pivot.

   Rappel :
   ```
   void echange(int tab[], int size, int i, int j)
   {
    int temp;

    if (i<0 || i>=size) { printf("Parametre i invalide\n"); exit(1); }
    if (j<0 || j>=size) { printf("Parametre j invalide\n"); exit(1); }

    temp   = tab[i];
    tab[i] = tab[j];
    tab[j] = temp;
   }
   ```

2. Ecrire une fonction récursive `triRecRapide` permettant de trier un sous–tableau allant de l'indice `depart` à l'indice `arrivee` du tableau `tab`. Utiliser la fonction `partition`.

   L'écriture de la fonction `triRapide` sera alors très simple :

   ```
   void triRapide(int tab[],int size)
   {
    triRecRapide(tab,size,0,size-1);
   }
   ```

3. Quel est l'ordre de complexité de ce tri ?

2 TP Exercice 2 : Tri et structures

1. En vous aidant du cours, écrire un nouveau type de structure (`struct personne`) composé d'un nom (`char nom[256] ;`), d'un prénom (`char prenom[256] ;`) et d'un age (`int age ;`).
2. Toujours en vous aidant du cours, écrire une fonction `printPersonne` et une fonction `readPersonne` permettant respectivement d'afficher et de saisir une personne.
   En interne, la fonction `readPersonne` contiendra 3 `scanf` !
3. Ecrire un programme principal comprenant la déclaration d'un tableau de 5 personnes, la lecture (avec `readPersonne`) de chacune des cases du tableau et l'affichage de chacune des cases du tableau (avec `printPersonne`).
4. Soit la fonction suivante (tri à bulle) permettant de trier un tableau d'entiers... et la fonction `echange`. Récupérer le fichier `page124-125_triBulle.c` se trouvant :

`http ://lab-sticc.univ-brest.fr/~rodin/FTP/Enseignements/L1/AlgoEtProg`

```
void echange(int tab[], int size, int i, int j)
{
 int temp;
 if (i<0 || i>=size) { printf("Parametre i invalide\n"); exit(1); }
 if (j<0 || j>=size) { printf("Parametre j invalide\n"); exit(1); }

 temp   = tab[i];
 tab[i] = tab[j];
 tab[j] = temp;
}

void triBulle(int tab[], int size)
{
 int i,j;
 int n = size;

 for(i=0; i<n ; i=i+1)              /* Pour la deuxieme boucle :   */
 {                                  /* n: comme d'habitude         */
  for(j=0; j<n-i-1 ; j=j+1)         /* -i, car les i derniers sont */
  {                                 /* deja tries, -1: pour le j+1 */
   if (tab[j]>tab[j+1])
   {
    echange(tab,size,j,j+1);
   }
  }
 }
}
```

Modifier ces fonctions (`echange` et `triBulle`) de façon à trier le tableau de (`struct personne`) en fonction de l'âge.

Utiliser cette nouvelle fonction dans le programme principal, et afficher les éléments de tableau après le tri.

3 TP Exercice 3 : une fonction de traitement de chaînes de caractères

Ecrire une fonction `remplace` permettant de modifier une chaîne de caractère passée en paramètre. La modification se fait par remplacement de toutes les occurences d'un caractère donné, remplacement par un autre caractère.
Exemple :

```
#include <stdio.h>
/* #include <string.h> ... pas utile ici car, strlen pas utile ! */

/* La fonction remplace ... a faire */

int main(void)
{
 char str[256] = "toto";

 remplace(str,'o','a');

 printf("str=%s\n",str);  /* str=tata */
 return 0;
}
```

Remarquons qu'ici,
la chaîne est directement modifiée par la fonction (i.e. sans utilisation de `&`).