UNIVERSITE PARIS DAUPHINE

2003 - 2004

Université Paris Dauphine
IUP Génie Mathématique et Informatique
IUP 2ème année

MISE A NIVEAU EN INFORMATIQUE :

EXEMPLE D'ALGORITHMES ET DE PROGRAMMES MANIPULANT LES CHAINES DE CARACTERES

Maude Manouvrier

1. **La fonction `Egalite_Chaines` compare les deux chaînes de caractères passées en paramètres. La fonction retourne `1` si les deux chaînes sont égales et `0` sinon.**

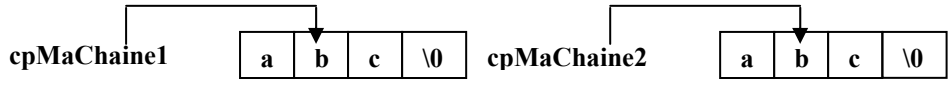
```
/* La fonction prend en paramètres deux chaînes de caractères
   gérées par des pointeurs */

int Egalite_Chaines (char *cpMaChaine1, char *cpMaChaine2)
{
      /* Tant que l'on a pas atteint la fin d'au moins une des
         deux chaînes de caractères */

      while( (*cpMaChaine1!='\0') && (*cpMaChaine2!='\0') )
      {
            /* Si le 1er caractère est ≠ dans les 2 chaînes */

            if (*cpMaChaine1 != *cpMaChaine2)
            {
               /* Les chaînes sont différentes, on quitte la
                  fonction en retournant la valeur 0*/

               return 0 ;
            }

            /* Si le caractère est le même dans les 2 chaînes,
               on passe au caractère suivant en avançant les
               2 pointeurs */

            cpMaChaine1++ ;
            cpMaChaine2++ ;

      } /* Fin du tant que ⇒ on a atteint la fin d'au moins une
           chaîne de caractères */

      /* Si on a atteint la fin des deux chaînes */
      if ( (*cpMaChaine1 =='\0') && (*cpMaChaine2 =='\0') )
      {
         /* Alors les chaînes sont égales, on retourne 1*/
         return 1 ;
      }

      /* Sinon on a atteint la fin d'une seule chaîne */
      else
      {
         /* Les chaînes sont différentes, on retourne 0*/
         return 0 ;
      }
}
```

Si on simule l'appel de `Egalite_Chaines("abc","abc")`, il se passe les actions suivantes :

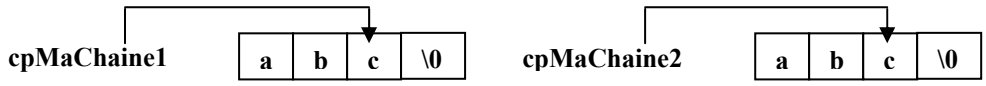
- **Au début de la fonction :** **cpMaChaine1** → | **a** | **b** | **c** | **\0** | **cpMaChaine2** → | **a** | **b** | **c** | **\0** |

- **Les caractères (de type `char`) `*cpMaChaine1` et `*cpMaChaine2` ont la même valeur `'a'`, ce qui est différent du caractère de fin de chaîne `'\0'`, on rentre donc dans la boucle `while`.**

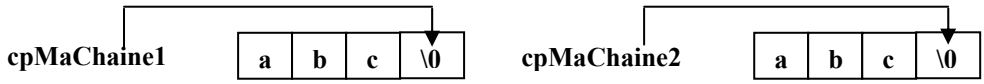
  - **Les caractères `*cpMaChaine1` et `*cpMaChaine2` ne sont pas différents, par conséquent on ne retourne pas 0, mais on incrémente les pointeurs :**



- **Aucun des deux caractères `*cpMaChaine1` et `*cpMaChaine2` ne correspond au caractère de fin de chaîne (leur valeur est `'b'`), on recommence donc à exécuter le corps de la boucle `while`. Les caractères `*cpMaChaine1` et `*cpMaChaine2` n'étant pas différents, on incrémente les pointeurs :**



- **Aucun des deux caractères `*cpMaChaine1` et `*cpMaChaine2` ne correspond au caractère de fin de chaîne (leur valeur est `'c'`), on recommence donc à exécuter le corps de la boucle `while`. Les caractères `*cpMaChaine1` et `*cpMaChaine2` n'étant pas différents, par conséquent on ne retourne pas 0, mais on incrémente les pointeurs :**



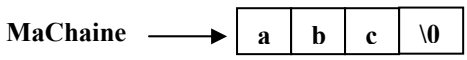
- **Les deux caractères `*cpMaChaine1` et `*cpMaChaine2` ont pour valeur le caractère de fin de chaîne, on sort donc de la boucle `while`.**

- **Après le `while`, comme les deux caractères `*cpMaChaine1` et `*cpMaChaine2` sont égaux au caractère de fin de chaîne, on retourne 1 (les chaînes sont identiques).**

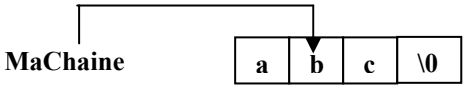
2. **La fonction `Afficher_Inverse` n'affiche pas la chaîne à l'envers car on effectue l'affichage (appel de `printf`) avant de faire l'appel récursif.**

```
/* La fonction prend en paramètre une chaîne de caractères gérée
   par un pointeur */
void Afficher_Inverse(char *MaChaine)
{
   char cara;
   /* Si la chaîne ne contient qu'un caractère */
   if(strlen(MaChaine)==1)
   {
        /* On affiche le caractère unique de la chaîne */
        printf("%c\n",*MaChaine);
   }
   /* Si la chaîne contient plus d'un caractère */
   else if(strlen(MaChaine)>1)
        {
             /* On copie le premier caractère dans la variable
                locale cara */
             cara = *MaChaine;
             /* On affiche le caractère cara qui est le 1er
                caractère de la chaîne passée en paramètre */
             printf("%c\n",cara);
             /* On appelle à nouveau la fonction avec la chaîne
                sans le premier caractère (le pointeur est
                déplacé d'un caractère il pointe sur le 2ème)*/
             Afficher_Inverse(MaChaine+1);
        }
}
```

Simulons l'appel `Afficher_Inverse("abcd")`. Au début de l'appel, on a :



- **`strlen(MaChaine)=3`, par conséquent, on passe dans le corps du deuxième `if`. `*MaChaine='a'`. Cette valeur est copiée dans `cara`, qui est affiché, et la fonction est à nouveau appelée avec en paramètre l'adresse du deuxième caractère :**



- **Au deuxième appel, le caractère 'b' est affiché etc.**

La fonction ainsi écrite affiche donc la chaîne à l'endroit.

Pour qu'elle l'affiche à l'envers, il faut inverser le `printf` et l'appel récursif. Ainsi, les caractères de la chaîne vont être empilés dans l'ordre dans la pile d'exécution et vont donc être affiché dans l'ordre inverse.

3. **L'algorithme d'une fonction `egale_inverse` qui prend en paramètre deux chaînes de caractères, `Chaine1` et `Chaine2`, et retourne 1 si `Chaine1` correspond à `Chaine2` à l'envers ou 0 sinon, est le suivant :**

```
Données en entrée :     Chaine1 chaîne de caractères
                        Chaine2 chaîne de caractères

Variables locales :     Compteur1 entier
                        Compteur2 entier

/* Si les chaînes n'ont pas la même longueur, elles sont ≠ */
SI longueur(Chaine1)≠longueur(Chaine2) ALORS retourner 0

/* Si les chaînes ont même longueur */
SINON
   /* Compteur1 = l'indice du premier caractère de Chaine1 */
   Compteur1 = 1
   /* Compteur2 = l'indice du dernier caractère de Chaine2 */
   Compteur2 = longueur(Chaine2)

/* On avance à l'endroit dans Chaine1 et à l'envers de Chaine2 */
   TANT QUE Chaine1[Compteur1] == Chaine2[Compteur2] FAIRE
         Compteur1 = Compteur1 + 1
         Compteur2 = Compteur2 - 1
   FIN TANT QUE

   /* Si on a parcouru toute la chaîne Chaine2 à l'envers */
   SI Compteur2 == 0 ALORS retourner 1
   /* Sinon, on s'est arrêté avant de parcourir toute la chaîne
      Chaine2 à l'envers */
   SINON Retourner 0
   FIN SI
 FIN SI
```