Travaux Dirigés n°6

Chaîne de caractères

Une chaîne de caractères est une suite de caractères, c'est-à-dire un ensemble de symboles faisant partie du jeu de caractères, défini par le code ASCII. En langage C, une chaîne de caractères est un tableau, comportant plusieurs données de type char, dont le dernier élément est le caractère nul '\0', c'est-à-dire le premier caractère du code ASCII (dont la valeur est 0). Ce caractère est un caractère de contrôle (donc non affichable) qui permet d'indiquer une fin de chaîne de caractères. Ainsi une chaîne composée de n éléments sera en fait un tableau de $n+1$ éléments de type char. On peut par exemple représenter la chaîne Bonjour de la manière suivante :
B o n j o u r \0
Pour définir une chaîne de caractères en langage C, il suffit de définir un tableau de caractères. Le nombre maximum de caractères que comportera la chaîne sera égal au nombre d'éléments du tableau moins un (réservé au caractère de fin de chaîne).

```
char Nom_du_tableau[Nombre_d_elements]
```

où `Nombre_d_elements`est égal au nombre maximal de caractères de la chaîne + 1.
On peut intialiser une chaîne de caractère en l'affectant à une chaîne de caractère lors de sa délaration. On peut comme pour un tableau préciser ou pas sa taille mais veillez si on la précise que la taille soit suffisamment grande. Par exemple

```
char a[]="Bonjour";
char a[15]="Bonjour";
```

En revanche, comme pour les tableaux, dans le programme on ne peut pas effectuer l'affectation directe $a = b$ où a, b sont deux chaîne de caractères. Pour cela, on utilise la bibliothèque `<string.h>`où quelques fonctions sont énumérées ci-dessous

```
char *strcpy(char *toHere, const char *fromHere);  copie une
chaîne de caractères  à une autre
int strcmp(const char *, const char *);  compare
deux chaînes en utilisant l'ordre lexicographique
size_t strlen(const char *);  retourne la longueur d'une chaîne caractères
char *strncpy(char *toHere, const char *fromHere, size_t);  copie
au plus size_t caractères d'une chaîne à une autre
int strncmp(const char *, const char *, size_t);  compare les size_t premiers
caractères au plus de deux chaînes en utilisant l'ordre lexicographique
char *strncat(char *dest, const char *src, size_t);  concatène au plus size_t caractè
de la chaîne src à la suite de dest
```

Exercice 1

Ecrire l'entête d'une fonction `manip_chaine` qui prend en paramètre une chaine de caractères.

Exercice 2

Ecrire l'entête d'une fonction `modif_chaine` qui prend en paramètre une chaine de caractères et retourne un entier. Cette fonction doit pouvoir modifier le contenu de la chaine de caractères.

Exercice 3

Ecrire une fonction `longueur` qui prend en paramètre une chaine de caractère et retourne la longueur de cette chaine. Vous ne devez pas utiliser la fonction `strlen` dans cet exercice.

Exercice 4

Ecrire une fonction qui étant donné un verbe régulier en "er" en affiche sa conjugaison au présent de l'indicatif.

Solution

```
conjuguer(char verbe[20]){
int taille;
char racine[20];
char tmp[20];
char term[3];
taille = strlen(verbe);
term[0] = verbe[taille-2];
term[1] = verbe[taille-1];
term[2] = '\0';
if(strcmp(term,"er") == 0)
{
strncpy(racine,verbe,taille-2);
racine[taille-2]='\0';
printf("je %s\n",strcat(strcpy(tmp,racine),"e"));
printf("tu %s\n",strcat(strcpy(tmp,racine),"es"));
printf("il %s\n",strcat(strcpy(tmp,racine),"e"));
printf("nous %s\n",strcat(strcpy(tmp,racine),"ons"));
printf("vous %s\n",strcat(strcpy(tmp,racine),"ez"));
printf("ils %s\n",strcat(strcpy(tmp,racine),"ent"));
  }
else
printf("%s n'est pas un verbe en er\n",verbe);
}
```

Exercice 5

Ecrire une fonction qui, étant donné une chaîne de caractères, convertit les majuscules en minuscules et les minuscules en majuscules, remplace les signes d'espacements par des virgules et laisse les autres caractères inchangés. Cette chaîne de caractères résultat sera placée dans le

deuxième paramètre de la fonction.
Exemple : "BONNE nuit les PETITS enfants !" → "bonne,NUIT,LES,petits, ENFANTS !"

Solution

```
transformer(char chaine_ini[], char chaine_res[]){
int i, taille;
taille = strlen(chaine_ini);
for(i=0;i<=taille;i++)
{
if(isupper(chaine_ini[i]))
chaine_res[i] = tolower(chaine_ini[i]);
else if(islower(chaine_ini[i]))
chaine_res[i] = toupper(chaine_ini[i]);
  else if(isspace(chaine_ini[i]))
chaine_res[i] = ',';
    else
chaine_res[i] = chaine_ini[i];
}
}
```

Matrices

Dans les exercices suivants, nous considérons les matrices entières carrée NxN avec N constante.

Exercice 6

Écrire une fonction `init_zero` qui prend comme argument d'une matrice NxN A et initialise tous les éléments de A à 0.

Exercice 7

Écrire une fonction `affiche_mat` qui prend comme argument d'une matrice NxN A et qui l'affiche ligne par ligne

Solution

```
void affiche_mat(int m[N][N]){
     int i,j;
     for(i=0;i<N;i++){
         for(j=0;j<N;j++)
            printf("%d ",m[i][j]);
         printf("\n");
     }
}
```

Exercice 8

Écrire une fonction `matrice_plus` qui prend comme argument 3 matrices NxN A, B et C et qui calcule la somme $C = A + B$.

Solution

```
void matrice_plus(int mat1[N][N], int mat2[N][N],
int mat_res[N][N]){
   int i, j, k, s;
     for(i=0;i<N;i++)
         for(j=0;j<N;j++){

            mat_res[i][j] = mat1[i][j]+mat2[i][j];
         }
}
```