UNIVERSITE PAUL SABATIER

L2 - S4 Programmation impérative en langage C

TP n° 4 : Chaînes de caractères.

Ce TP fait suite au TP du semestre dernier consacré aux chaînes de caractères. Cette fois-ci l'allocation est dynamique (malloc, calloc, realloc). Vous pouvez employer les fonctions de la librairie <string.h> en particulier : strlen, strcat, strcpy, strcmp ...

◇ Exercice **1** : La librairie <string.h> contient la fonction strcat de concaténation de chaînes, cette fonction prend 2 chaînes et modifie la première en lui concaténant la seconde. Ecrire dans un main une séquence permettant la concaténation de chaînes (différente de strcat) qui, à partir de deux chaînes de caractères, crée une autre chaîne résultat de la concaténation. Contrairement à strcat, les deux chaînes de départ seront conservées.
Ecrire ensuite une fonction d'en-tête : **char* concat (char* ch1, char* ch2)** permettant d'accomplir le même travail.
Vous pouvez utiliser les fonctions de <string.h>

◇ Exercice **2** : On souhaite optimiser la taille de la mémoire vive allouée lors de la saisie d'une chaîne de caractères. Ecrire une fonction de saisie d'une chaîne de caractères qui crée une variable dynamique de la longueur de la chaîne tapée par l'utilisateur.
- cette chaîne pourra contenir des espaces ou tout autre symbole et se terminera par retour chariot : LF(code ascii 10)
- on lira les divers caractères un à un et on allouera à chaque fois une case supplémentaire au tableau de caractères (realloc).

◇ Exercice **3** : Vous disposez de 2 chaînes de caractères ch1 et ch2 . Ecrire une séquence permettant de les permuter en distinguant le cas de chaînes statiques puis de chaînes dynamiques.

◇ Exercice **4** : On veut écrire un programme qui à partir d'un fichier de mots crée un nouveau fichier contenant la liste des mots triés dans l'ordre alphabétique avec leurs fréquences d'apparitions.
Exemple :

fichier initial
lapin
tortue
canard
tortue
hibou
chat
tortue
chameau
canard
hibou

fichier final
canard 2
chameau 1
chat 1
hibou 2
lapin 1
tortue 3

◇ Exercice **5** : Les arguments de la ligne de commande : quand la fonction main est appelée, deux arguments lui sont transmis : argc (argument count) représente le nombre d'arguments de la ligne de commande qui appelle le programme et argv (argument values) est un pointeur sur un tableau de chaînes de caractères qui contiennent les arguments. Le prototype de la fonction main est alors :

int main (int argc, char *argv[])

Exemple : Si on écrit un programme prog.c et qu'on l'exécute avec la ligne de commande " ./prog bonjour monsieur " argc vaut alors 3 ; argv[0] vaut "prog" ; argv[1] vaut "bonjour" ; argv[2] vaut "monsieur".
Modifier le programme précédent pour que le nom du fichier analysé et le nom du fichier résultat soient transmis en argument.

◇ Exercice **6** :

1. Ecrire une fonction de remplacement de la première occurence, si elle existe, d'une sous-chaîne par une autre sous-chaîne dans une chaîne. La chaîne de départ est conservée sans modification. (Cette fonction ne remplace rien si la sous-chaîne à remplacer est inexistante dans la chaîne de départ.
2. Même question pour toutes les occurences de la sous-chaîne.
3. Reprendre la question 2 mais cette fois-ci en modifiant directement la chaîne de départ.

S'il vous reste du temps :

◇ Exercice **7** : Dans un programme, on a besoin de manipuler les noms des 7 jours de la semaine : "lundi","mardi",... Comment faire pour optimiser la mémoire ?

1. Première solution : essayer d'abord avec un tableau de char* (statique ou dynamique) Penser à allouer et désallouer la mémoire comme il convient.
2. Deuxième solution : essayer directement l'instruction :
   char* jours[7]={"lundi","mardi","mercredi","jeudi","vendredi","samedi","dimanche"} ;
3. On veut mettre en majuscule la première lettrre de chaque jour, comment peut-on faire dans les deux cas ?

◇ Exercice **8** : Soient les deux structures suivantes :

```
struct S1 {
    int a ;
    char * ch ;
} ;
```

et

```
struct S2 {
    int a ;
    char ch[40] ;
} ;
```

On définit les variables suivantes :

```
struct S1 v11,v12 ;
struct S2 v21, v22 ;
```

1. Ecrire un programme qui :
   - définit ces types ,
   - crée deux variables v11 et v12 de type struct S1 et initialise v11 avec la valeur 3 pour a et "Kangourou" pour ch,
   - crée deux variables v21 et v22 de type struct S2 et initialise v21 avec la valeur 5 pour a et "Koala" pour ch.
2. Copier v11 dans v12 et v21 dans v22. Que se passe-t-il en mémoire ?
3. Modification de la chaîne de caractères v11.ch :
   (a) Une modification globale de v11.ch en "Renard" est-elle toujours possible ? entraîne-t-elle celle de v12.ch ?
   (b) et une modification partielle v11.ch[0]='c' ?
4. Modification de la chaîne de caractères v21.ch :
   (a) Une modification globale de v21.ch en "Lapin" est elle possible ?
   (b) Une modification partielle comme v21.ch[0]='c' entraîne-t-elle une modification de v22 ?

◇ Exercice **9** : (***)Ecrire une procédure qui crée un vecteur de mots de type `char **` : le nombre de mots est demandé à l'utilisateur ainsi que les divers mots, l'implantation sera faite dans la procédure. L'en-tête de cette procédure contiendra deux paramètres modifiables : le vecteur de mots et le nombre de mots présents dans le vecteur.

◇ ◇ ◇◇ ◇ ◇◇