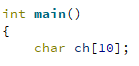
La gestion des chaînes de caractères en C/C++ (et dans d’autres langages) est toujours à la fois un passage obligé et une difficulté souvent mal abordée par les étudiants.

## Rappels :

Une chaîne de caractères est un tableau de caractères. Les caractères dans le tableau sont suivis d’un caractère spécial ‘\0’ de valeur 0 (première valeur de la table ASCII). Attention : tous les caractères de la chaîne ainsi que le caractère spécial ‘\0’ doivent être présents dans le tableau. Par conséquent, la taille du tableau doit être au moins égale au nombre de caractères de la chaîne + 1 (pour l’\0’). Par exemple : si je veux enregistrer la chaîne ˮbonjourˮ, il nous faudra un tableau d’au moins 8 cases (les 7 caractères de ˮbonjourˮ + ‘\0’)

## Déclaration d’une chaîne de caractères en C :

Déclaration de la chaîne *ch* dans la fonction *main()*

Nombre de caractères

Type des données dans le tableau

Nom du tableau

On a l’habitude de représenter le contenu du tableau de caractères en mémoire à l’aide du schéma suivant :

9

8

7

6

5

4

3

2

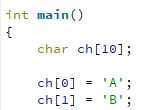
1

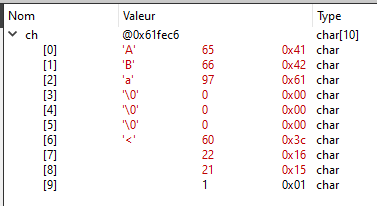
0

ch

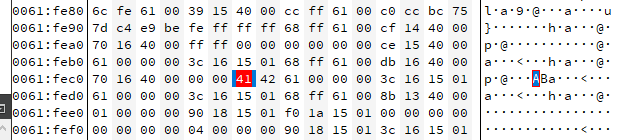
10 caractères possibles dans la chaîne

*ch* est le nom du tableau. C’est ce qu’on appelle un pointeur. La variable *ch* contient l’adresse en mémoire du premier caractère du tableau. Par exemple sur cette capture écran, on a le code source suivant :

Le pointeur *ch* contient l’adresse *0x61fec6*



Si maintenant on observe la mémoire à cette adresse :



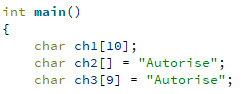
0x41 (caractère ASCII A) est à l’adresse 0X61fec6

## Ecriture et lecture dans un tableau de caractères

Dans l’exemple précédent, on avait écrit les caractères A et B aux emplacements 0 et 1 du tableau. Les emplacements s’appellent les indices (ou index) du tableau. Le premier indice est toujours 0 simplement parce que les indices sont en fait des offsets (décalages) en mémoire par rapport à l’adresse de base du tableau donc ici de *ch* (0x61fec6 dans notre exemple).

### Ecriture dans un tableau de caractères

#### Ecriture à la déclaration



Ici on a réservé 10 cases pour ch1 mais le contenu n’est pas initialisé. ch2 est automatiquement réservée avec le nombre suffisant de cases (9 cases) et initialisée avec la chaîne « Autorise ».

ch3 est réservée avec 9 cases et initialisée avec la chaîne « Autorise »

Remarque : cette dernière écriture est problématique puisqu’il faut que le nombre inscrit entre les crochets soit supérieur ou égal à la taille de la chaîne + 1 (‘\0’). Si on met 8, le compilateur signale une erreur.

En dehors de la déclaration d’une chaîne, les possibilités d’écriture sont plus restreintes.

#### Ecriture de caractères dans le code :



Les caractères doivent être spécifiés entre ‘ ‘ (quote)

#### Ecriture de chaînes dans le code :

IL N’Y A AUCUNE AUTRE POSSIBILITE QUE D’ECRIRE CHAQUE CARACTERE LES UNS APRES LES AUTRES !! ... à l’aide d’une boucle !

Par exemple :

ch1= ˮBonjourˮ ; N’EST PAS POSSIBLE

La fonction *strcpy()* permet cela :

La fonction *strcpy()* copie la chaîne ˮBonjourˮ à l’adresse ch2

source

destination

fonction

### Lecture

On accède à un caractère de la chaîne simplement en insérant l’indice du caractère voulu entre les crochets :



Si on veut lire une chaîne et pas juste un caractère ?

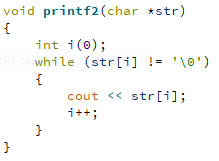
*cout* par exemple effectue une lecture d’une chaîne. *printf()* également*.*

Comment fonctionnent ces fonctions ?

Il faut leur fournir l’adresse de début du tableau donc le nom du tableau. Elles parcourent ce tableau d’une case par une case en partant de la première jusqu’à ce qu’elles atteignent le caractère spécial ‘\0’.

Elles sont donc construites autour d’une boucle et d’un test : typiquement une boucle *while*

Par exemple, on peut imaginer une version très simpliste de la fonction *printf()* (qui serait *printf2()* ):



On l’utiliserait ainsi :



## Travail demandé

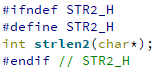
### Création des fichiers .h et .cpp

Ajouter un fichier str2.h qui contiendra les prototypes de nos fonctions.

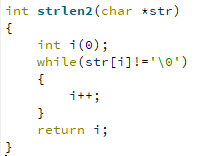
Ajouter le fichier str2.cpp qui contiendra le code source (les implémentations) de nos fonctions

### Recréer la fonction strlen() => strlen2()

Ajouter le prototype de la fonction strcpy2() dans str2.h :



Ajouter l’implémentation de la fonction *strlen2()* dans le fichier str2.cpp :



* Ajouter les lignes dans la fonction main() :

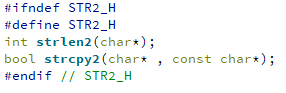


Il faut veiller à ajouter la ligne #include ˮstr2.hˮ

* Mettre un point d’arrêt et exécuter en pas à pas la fonction *strlen2()*

### Recréer la fonction strcpy() => strcpy2()

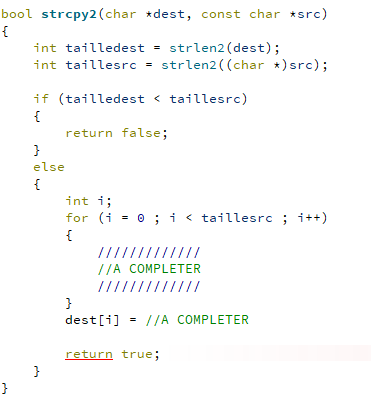
Ajouter le prototype de la fonction strcpy2() dans str2.h :



//prototypes

Remarque : il est très possible que vous n’ayez pas exactement les lignes commençant par #. ça n’a aucune importance. Ne les modifiez pas !

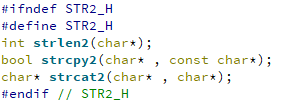
Ajouter l’implémentation de la fonction *strcpy2()* dans le fichier str2.cpp :



Compléter le code ci-dessus de façon à copier la chaîne source dans la chaîne destination.

### Recréer la fonction strcat() => strcat2()

Ajouter le prototype de la fonction *strcat2()*



Ajouter l’implémentation de la fonction *strcat2()*  dans le fichier str2.cpp :

