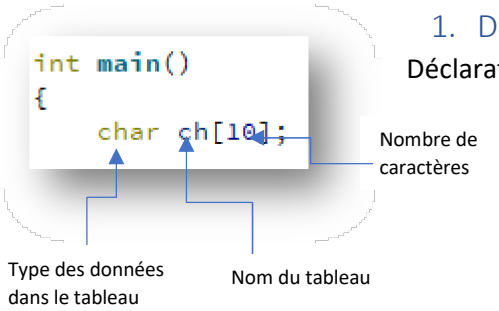
TP 7 : Les chaînes de caractères

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 :

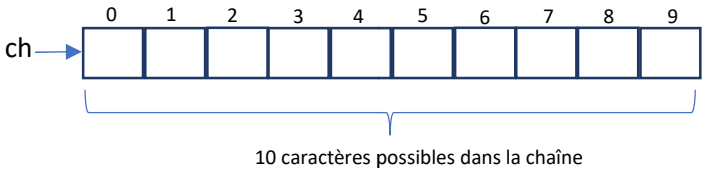
Une chaîne de caractères est un tableau de caractères. Les caractères dans le tableau sont suivis d'un caractère spécial '\0' de valeur 0 (première valeur de la table ASCII). Attention : tous les caractères de la chaîne ainsi que le caractère spécial '\0' doivent être présents dans le tableau. Par conséquent, 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 "bonjour", il nous faudra un tableau d'au moins 8 cases (les 7 caractères de "bonjour" + '\0')



1. Déclaration d'une chaîne de caractères en C :

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

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



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 :

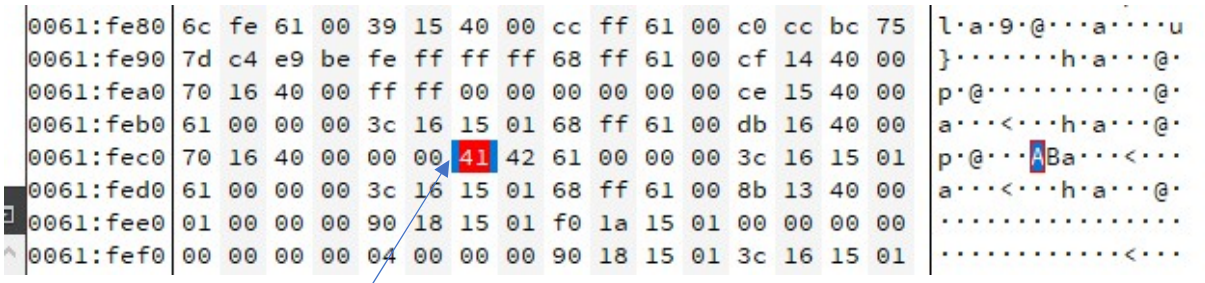
```
int main()
{
    char ch[10];

    ch[0] = 'A';
    ch[1] = 'B';
```

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

Nom	Valeur			Type
ch	@0x61fec6			char[10]
[0]	'A'	65	0x41	char
[1]	'B'	66	0x42	char
[2]	'a'	97	0x61	char
[3]	'\0'	0	0x00	char
[4]	'\0'	0	0x00	char
[5]	'\0'	0	0x00	char
[6]	'<'	60	0x3c	char
[7]		22	0x16	char
[8]		21	0x15	char
[9]		1	0x01	char

TP 7 : Les chaînes de caractères

Si maintenant on observe la mémoire à cette adresse :



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

2. 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).

2.1 Ecriture dans un tableau de caractères

Ecriture à la déclaration

```
int main()
{
    char ch1[10];
    char ch2[] = "Autorise";
    char ch3[9] = "Autorise";
```

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 :

```
ch1[0] = 'A';
ch1[1] = 'B';
```

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= "Bonjour" ; N'EST PAS POSSIBLE

La fonction *strcpy()* permet cela :

```
strcpy(ch2, "Bonjour");
```

fonction destination source

La fonction *strcpy()* copie la chaîne "Bonjour" à l'adresse ch2

TP 7 : Les chaînes de caractères

2.2 Lecture

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

```
cout << ch1[0];
```

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()*):

```
void printf2(char *str)
{
    int i(0);
    while (str[i] != '\0')
    {
        cout << str[i];
        i++;
    }
}
```

On l'utiliserait ainsi :

```
strcpy(ch2, "Bonjour");

printf2(ch2);
```

3. Travail demandé

3.1 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

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

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

```
#ifndef STR2_H
#define STR2_H
int strlen2(char*);
#endif // STR2_H
```

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

TP 7 : Les chaînes de caractères

```
int strlen2(char *str)
{
    int i(0);
    while(str[i]!='\0')
    {
        i++;
    }
    return i;
}
```

⇨ Ajouter les lignes dans la fonction main() :

```
char ch2[] = "Autorise";
cout << strlen2(ch2);
```

Il faut veiller à ajouter la ligne #include ”str2.h”

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

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

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

```
#ifndef STR2_H
#define STR2_H
int strlen2(char*);
bool strcpy2(char* , const char*);
#endif // 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 :

```
bool strcpy2(char *dest, const char *src)
{
    int tailledest = strlen2(dest);
    int taillesrc = strlen2((char *)src);

    if (tailledest < taillesrc)
    {
        return false;
    }
    else
    {
        int i;
        for (i = 0 ; i < taillesrc ; i++)
        {
            //////////////
            //A COMPLETER
            //////////////
        }
        dest[i] = //A COMPLETER

        return true;
    }
}
```

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

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

Ajouter le prototype de la fonction *strcat2()*

```
#ifndef STR2_H
#define STR2_H
int strlen2(char*);
bool strcpy2(char* , const char*);
char* strcat2(char* , char*);
#endif // STR2_H
```

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

```
char *strcat2(char *str1, char *str2)
{
      int taillestr1 = strlen2(str1);
      int taillestr2 = strlen2(str2);

      int taillestrf = taillestr1+taillestr2;

      char *str = new char[taillestrf+1];

      //A COMPLETER
      //INDICATIONS :
      //Il faut creer une premiere boucle en parcourant la
      //premiere chaine str1
      //Chaque caractere de str1 doit etre copie dans str
      //Une fois le parcours de str1 termine, il faut une
      //seconde boucle pour parcourir str2.
      //Chaque caractere de str2 doit etre copie dans str
      //apres le dernier caractere de str1 precedemment copie dans str
      //Ne pas oublier de mettre le caractere de fin de chaine '\0'

      return str;
}
```