Introduction à la programmation
Travaux pratiques: séance 14
INFO0201-1

X. Baumans
(xavier.baumans@ulg.ac.be)

29/04/2014

Programme de la séance

- Rappels chaînes de caractères
  $\rightarrow$ Gestion de texte avec un tableau de caractères (**char**)
  - Manipulations d'une suite de caractères contigus
  - Entrée-sortie de texte

Programme de la séance

- Rappels chaînes de caractères
  $\rightarrow$ Gestion de texte avec un tableau de caractères (**char**)
  - Manipulations d'une suite de caractères contigus
  - Entrée-sortie de texte
- Cryptage - décryptage simple

Déclarer et initialiser des chaînes de caractères

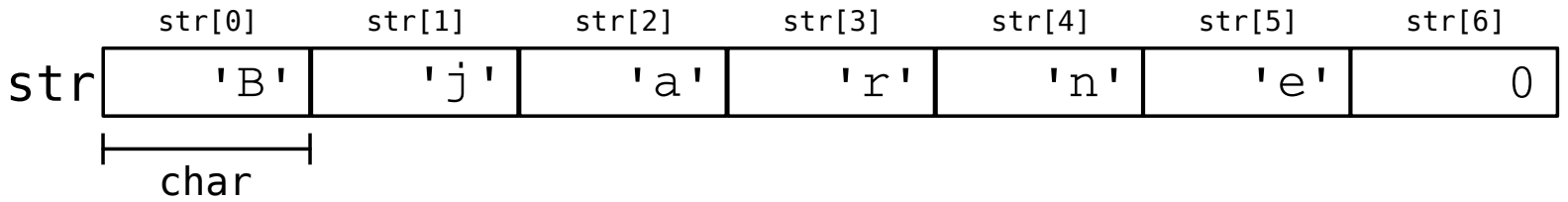
Pour déclarer et initialiser une chaîne de caractères, on déclare un tableau de type **char** et on l'initialise en écrivant la chaîne de caractères entre **guillemets doubles**.
Cela signale qu'il s'agit d'une chaîne de caractère et correspond à initialiser le tableau avec des chacun des caractères de la chaîne en terminant par le caractère **null** (code ASCII = 0)

```
1 char str[] = "Bjarne";
```

est équivalent à

```
1 char str[7] = {'B', 'j', 'a', 'r', 'n', 'e', '\0'};
```

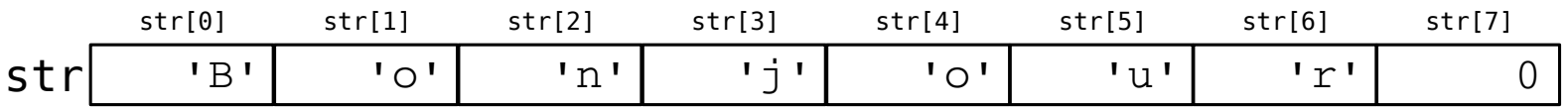
Représentation du tableau dans la mémoire :



Afficher des chaînes de caractères

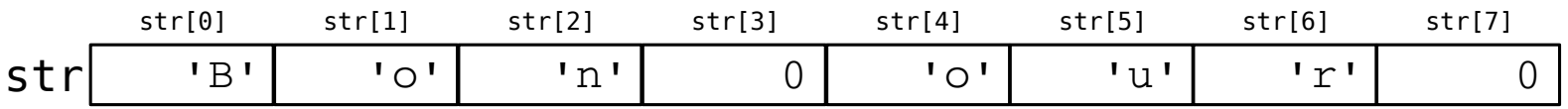
On ne peut normalement pas afficher un tableau entier de manière immédiate avec l'objet cout.
Les chaînes de caractères sont une exception : on peut afficher une chaîne de caractère en passant directement le nom du tableau qui la contient à l'objet cout :

```
1 char str[] = "Bonjour";
2 cout << str; // Affiche Bonjour
```



Le système affiche tous les caractères de la chaîne jusqu'à ce que le caractère **null** soit rencontré.

```
1 str[3] = 0;
2 cout << str; // Affiche Bon
```



Saisir des chaînes de caractères au clavier

L'objet cin permet lui aussi de lire directement des chaînes de caractères complètes. Pour cela, il suffit de signaler à cin qu'il doit placer la saisie clavier dans un tableau d'éléments de type char.

```
1 char buffer[28];
2 cin >> buffer; // ATTENTION: max 27 caractères !!!
```

Le système va copier tous les caractères saisis au clavier dans le tableau de caractères jusqu'à ce qu'un espace soit rencontré. Il placera alors automatiquement un caractère nul à la fin de la chaîne.

ATTENTION : le tableau doit être de taille suffisante pour pouvoir accueillir toute la chaîne de caractères (caractère null de fin y compris). Si ce n'est pas le cas, le système tentera d'écrire dans des éléments situés derrière le tableau déclaré, qui n'existent pas, et cela ne fonctionnera pas !

Chaînes de caractères dans des fichiers

- Écrire dans un fichier : l'écriture d'une chaîne de caractères dans un fichier fonctionne de la même manière qu'avec cout :

```
1 char buffer[28] = "Hello";
2 mon_fichier << buffer;
```

Chaînes de caractères dans des fichiers

- Écrire dans un fichier : l'écriture d'une chaîne de caractères dans un fichier fonctionne de la même manière qu'avec cout :

```
1 char buffer[28] = "Hello";
2 mon_fichier << buffer;
```

- Lire dans un fichier : la lecture d'une chaîne de caractères fonctionne de la même manière qu'avec cin :

```
1 char buffer[256];
2 mon_fichier >> buffer;
```

Comme avec cin, la lecture s'interrompt dès qu'un caractère 'espace' est rencontré.

Opérations sur les chaînes de caractères

Avec **tous** les tableaux, donc les chaînes de caractères aussi, il est impossible de copier directement un tableau dans l'autre :

```
1 char buffer[256] = "Bonjour";
2 char buffer2[256] = "Hello";
3 buffer_2 = buffer; // NON !!!
```

Il n'est pas permis de copier un tableau dans un autre en utilisant l'opérateur d'affectation (explication à la prochaine séance).
Il faut donc réaliser les opérations "à la main", caractère par caractère.

Fonctions de caractères standards

Quelques fonctions de chaînes de caractères standard :

- strlen(char[] str) : retourne la longueur de la chaîne de caractères ;

Fonctions de caractères standards

Quelques fonctions de chaînes de caractères standard :

- strlen(char[] str) : retourne la longueur de la chaîne de caractères ;
- strcpy(char[] destination, char[] source) : copie la chaîne *source* dans la chaîne *destination* ;

Fonctions de caractères standards

Quelques fonctions de chaînes de caractères standard :

- strlen(char[] str) : retourne la longueur de la chaîne de caractères ;
- strcpy(char[] destination, char[] source) : copie la chaîne *source* dans la chaîne *destination* ;
- strcat(char[] destination, char[] source) : ajoute une copie de *source* à la suite de la chaîne *destination* ;

Fonctions de caractères standards

Quelques fonctions de chaînes de caractères standard :

- strlen(char[] str) : retourne la longueur de la chaîne de caractères ;
- strcpy(char[] destination, char[] source) : copie la chaîne *source* dans la chaîne *destination* ;
- strcat(char[] destination, char[] source) : ajoute une copie de *source* à la suite de la chaîne *destination* ;
- strcmp(char[] str1, char[] str2) : compare les chaînes *str1* et *str2* et renvoit :
  - 0 : si les chaînes sont identiques ;
  - > 0 : si le premier caractère différent a une valeur plus grande dans *str1* que dans *str2* ;
  - < 0 : dans le cas contraire.

Exercices

1. Décryptage d'un code de César.
   - A partir du mot de passe (clef) :
     - A partir des fonctions précédemment écrites, écrire une fonction qui permet de décrypter une chaîne de caractères à partir de la clé du code de César correspondant (attention à l'intervalle $[32, 127]$ des caractères imprimables) ;
     - Charger le fichier crypté à la séance précédente et le décrypter à partir du mot de passe. Afficher le message décodé et l'enregistrer dans un fichier appelé message_decode.txt.
   - Sans la clef :
     - Charger le fichier message_crypte.txt disponible à l'adresse `http://mate06.phys.ulg.ac.be/` et tenter de le décrypter sans l'aide de la clef.
     - Constater qu'un code César est relativement facile à casser...