Programmation C++ (débutant)/Les tableaux de char

Avant-propos important

Lorsqu'on étudie le C++ faut-il étudier d'abord la classe string ou d'abord les tableaux de char ?
Dans ce cours, nous étudierons en premier les tableaux de char, fidèle à notre pédagogie ascendante : nous partons des entités les plus rudimentaires pour construire petit à petit des entités de haut niveau. Ce n'est que dans plusieurs chapitres que nous présenterons la classe string.
Ce choix pédagogique est largement assumé dans ce livre mais il suscite en général une controverse assez vive.
Qui détient la vérité en matière de pédagogie ? Il serait bon d'ailleurs que ceux qui sont opposés à ce choix proposent un autre livre : "*Le C++ pour débutants avec une pédagogie descendante.*"
Il serait très intéressant de pouvoir comparer les différentes stratégies pédagogiques.

Le cours du chapitre 8 : les tableaux de char

string ou tableaux de char

En C++, il existe plusieurs façons de représenter les chaînes de caractères : on peut utiliser la classe prédéfinie string ou on peut utiliser des tableaux de char. On parle alors parfois de chaîne de caractères de style C.
Pour des raisons pédagogiques qui nous semblent fondamentales, nous allons étudier les chaînes de caractères représentées par des tableaux de char.
Ceci nous permettra de mettre en évidence les insuffisances de cette représentation et pourquoi il est largement préférable d'utiliser la classe string. Il faut avoir en tête que la classe string fait appel à des notions finalement très complexes (l'allocation dynamique de mémoire notamment, ou encore les pointeurs), même si son emploi est très simple.

Chaînes de caractères de style C

Une telle chaîne de caractères est contenue dans un tableau de char. Chaque caractère sera dans une case du tableau. A la fin d'une chaîne de caractères (qui n'est pas forcément à la dernière case du tableau) doit se trouver le caractère spécial noté '\0' qui indique la fin de la chaîne.

Affichage et saisie d'une chaîne

On peut afficher une chaîne de caractères par cout : le tableau de caractères sera alors affiché jusqu'au caractère de fin de chaîne.
On peut saisir une chaîne par cin : le caractère de fin de chaîne est alors rajouté automatiquement.
On peut accéder au caractère numéro i d'une chaîne t en indexant le tableau t[i].

Exemple 1 : affichage et saisie

Dans cet exemple, on déclare un tableau de 20 char noté tt : dans chaque case de 0 à 7 on place une lettre et dans la case 8 on place le caractère de fin de chaîne.

```
#include <iostream>
using namespace std;

int main()
```

```
{
    char tt[20];
    tt[0] = 'B';
    tt[1] = 'O';
    tt[2] = 'N';
    tt[3] = 'J';
    tt[4] = 'O';
    tt[5] = 'U';
    tt[6] = 'R';
    tt[7] = '\0';
    cout << tt;
    return 0;
}
```

- **Explications**

tt contient la chaîne de caractères "BONJOUR". Nous avons construit case par case cette chaîne ! En général, on utilisera des fonctions prédéfinies nous permettant de faire cela ! On peut afficher cette chaîne par un simple cout.
Remarque : dans la pratique, on ne copiera JAMAIS lettre par lettre une chaîne dans un tableau.

- **Exécution**

Lorsqu'on exécute le programme voici ce qu'on obtient à l'écran :
BONJOUR

Exemple 2 : saisie d'une chaîne

```
#include <iostream>
using namespace std;

int main()
{
    char tt[20];
    cout << "Tapez une chaîne SVP : ";
    cin >> tt;
    cout << "Vous avez tapé la chaîne : ";
    cout << tt << endl;
    return 0;
}
```

- **Explications**
  - On déclare un tableau de 20 char noté tt.
  - On saisit une chaîne de caractères grâce à cin.
  - On affiche alors la chaîne de caractères qu'on vient de taper par un cout.
- **Exécution**

Lorsqu'on exécute le programme voici ce qu'on obtient à l'écran :
Tapez une chaîne SVP : **BONJOUR**
Vous avez tapé la chaîne : BONJOUR

Remarque: le tableau tt est limité à 20 caractères. Cependant une entrée de 30 caractères les affiche aussi bien !! Où est l'erreur ?

Manipulation de chaînes

Les chaînes de caractères sont contenues dans des tableaux de char, il existe de nombreuses fonctions prédéfinies dans le fichier cstring qui permettent de manipuler simplement ces tableaux. Ainsi, on peut aisément copier, comparer, transformer… des chaînes de caractères. Nous allons étudier quelques-unes de ces fonctions standard.

Comparaison de 2 chaînes

On ne teste pas l'égalité de 2 chaînes par ==, on utilise strcmp(chaine1,chaine2) qui renvoie :
0 si les 2 chaînes sont égales.
un nombre <0 si chaîne1 est avant chaîne2 dans l'ordre lexicographique.
un nombre >0 si chaîne1 est après chaîne2 dans l'ordre lexicographique.
Pour utiliser strcmp, il faut inclure cstring.

Exemple 3 : comparaison de 2 chaînes

```
#include <iostream>
#include <cstring>

using namespace std;

int main()
{
    char tt[20];
    cout << "Tapez une chaîne : "; cin >> tt;
    if(strcmp(tt, "BONJOUR")==0)
        cout<<"GAGNE"<<endl;
    else cout<<"PERDU"<<endl;
    return 0;
}
```

- **Explications**
  - Dans cet exemple, on commence par demander à l'utilisateur de saisir une chaîne de caractères tt.
  - Grâce à la fonction strcmp, on va comparer cette chaîne à la chaîne "BONJOUR".
  - Si on a tapé la chaîne "BONJOUR", on affichera le message "GAGNE", sinon on affichera le message "PERDU".
- **Exécution 1**

Lorsqu'on exécute le programme voici ce qu'on obtient à l'écran :
Tapez une chaîne : **BONSOIR**
PERDU

- **Exécution 2**

Lorsqu'on exécute le programme voici ce qu'on obtient à l'écran :
Tapez une chaîne : **BONJOUR**
GAGNE

Exemple 4 : ordre lexicographique

```
#include <iostream>
#include <cstring>

using namespace std;

int main()
{
    char a[20];
    cout << "Tapez une chaîne : "; cin >> a;
    if (strcmp(a,"BONJOUR")>0)
        cout << a << " est après BONJOUR" << endl;
    else
        cout << a << " est avant BONJOUR" << endl;
    return 0;
}
```

- **Explications**
  - Dans cet exemple, on demande à l'utilisateur de saisir une chaîne de caractères a.
  - On compare ensuite cette chaîne à la chaîne de caractères "BONJOUR".
  - Si la chaîne tapée se trouve avant "BONJOUR" dans l'ordre lexicographique, on affiche un message indiquant que la chaîne se trouve avant "BONJOUR".
  - Si la chaîne se trouve après "BONJOUR", on affiche un message similaire.
- **Exécution 1**

Lorsqu'on exécute le programme voici ce qu'on obtient à l'écran :
Tapez une chaîne : **AAA**
AAA est avant BONJOUR

- **Exécution 2**

Lorsqu'on exécute le programme voici ce qu'on obtient à l'écran :
Tapez une chaîne : **ZZZ**
ZZZ est après BONJOUR

Copie d'une chaîne

On n'utilise pas l'affectation pour copier une chaîne dans une autre.
On utilise strcpy(chaine1,chaine2) qui copie la deuxième chaîne dans la première.

Exemple 5

```
#include <iostream>
#include <cstring>

using namespace std;

int main()
{
    char a[20], b[20];
    strcpy(a, "BONJOUR");
```

```
    strcpy(b, a);
    cout << "La chaîne b vaut : " << b << endl;
    return 0;
}
```

- **Explications**
  - Dans cet exemple, on déclare 2 chaînes de caractères a et b.
  - On copie la chaîne "BONJOUR" dans a en utilisant strcpy.
  - On copie ensuite la chaîne a dans la chaîne b toujours en utilisant strcpy.
  - On affiche ensuite la chaîne b.
- **Exécution**

Lorsqu'on exécute le programme voici ce qu'on obtient à l'écran :
La chaîne b vaut BONJOUR

Longueur d'une chaîne

- La fonction strlen permet de connaître le nombre de caractères d'une chaîne.
- On ne tient pas compte du caractère de fin de chaîne lorsqu'on compte les caractères.

Exemple 6 : longueur d'une chaîne

```
#include <iostream>
#include <cstring>

using namespace std;

int main()
{
    char a[20];
    int b;
    cout << "Tapez une chaîne : "; cin >> a;
    b = strlen(a);
    cout << "Taille de la chaine = " << b << endl;

    return 0;
}
```

- **Explications**
  - Dans cet exemple on déclare une chaîne de caractères a et un entier b.
  - On demande à l'utilisateur de saisir la chaîne a au clavier.
  - Dans l'entier b, on calcule la longueur de la chaîne a en utilisant la fonction strlen.
  - On affiche par un cout la valeur de b.
- **Exécution**

Lorsqu'on exécute le programme voici ce qu'on obtient à l'écran :
Tapez une chaîne : BONJOUR
Taille de la chaîne=7

Validation d'une donnée saisie

Lorsqu'on saisit une chaîne de caractères, l'utilisateur peut toujours taper autre chose que ce qui était demandé : par exemple, on demande à l'utilisateur de taper une heure sous un certain format. Il faut vérifier que l'utilisateur a bien tapé une donnée correcte et lui demander de la retaper s'il s'est trompé.

Exemple 7 : validation d'une "heure"

```
#include <iostream>

using namespace std;

bool valide(char t[])
{
bool r=true;
int h,m;
     if( t[0]<'0' ||t[0]>'9')
     r=false;
     else if( t[1]<'0' ||t[1]>'9')
          r=false;
          else if( t[3]<'0' ||t[3]>'9')
                r=false;
                else if( t[4]<'0' ||t[4]>'9')
                      r=false;
                      else if(t[2]!='h')
                            r=false;
                            else if(t[5]!='\0')
                                  r=false;
                                  else { h=(t[0]-'0')*10+(t[1]-'0');
       if(h>23)r=false; m=(t[3]-'0')*10+(t[4]-'0');
       if(m>59)r=false; }

return r;
}

int main()
{
    char a[20];
    do {
        cout << "Tapez une heure sous le format ..h.. : ";
        cin >> a;
    } while (!valide(a));
    return 0;
}
```

- **Explications**

La fonction valide a comme paramètre une chaîne de caractères et renvoie le booléen true si la chaîne passée en paramètre est une heure valide et renvoie false sinon.
Pour tester si notre chaîne est valide, il faut :

vérifier que les caractères 0, 1, 3 et 4 sont bien des chiffres.
vérifier que le caractère 2 est bien le caractère 'h'.
vérifier que le caractère 5 est bien le caractère de fin de chaîne '\0'.
calculer dans h le nombre d'heures et vérifier que h est inférieur ou égal à 23.
calculer dans m le nombre de minutes et vérifier que m est bien inférieur ou égal à 59.
Si une de ces conditions n'est pas vérifiée, la chaîne ne sera pas valide.
Dans le programme principal, on saisit une chaîne de caractères au clavier.
Tant que la chaîne tapée ne sera pas une heure qui correspond au format imposé, alors on demandera à l'utilisateur de saisir à nouveau une nouvelle chaîne.

- **Exécution**

Lorsqu'on exécute le programme voici ce qu'on obtient à l'écran :
Tapez une heure sous le format ..h.. : **aze**
Tapez une heure sous le format ..h.. : **12h50h**
Tapez une heure sous le format ..h.. : **25h98**
Tapez une heure sous le format ..h.. : **12h06**

Séparateurs

Par défaut le caractère indiquant la fin d'une chaîne en cours de saisie est l'espace ou le retour chariot. Cela empêche de saisir une chaîne comportant un espace.
cin.getline(…) permet de saisir une chaîne pouvant comporter des espaces : seul le retour chariot –touche «entrée») fait alors office de séparateur. On peut également tronquer la chaîne en indiquant la taille du tableau contenant cette chaîne.

Exemple 8 : saisie d'une chaîne avec espaces

```
#include <iostream>

using namespace std;

int main()
{
    char tt[10];
    cout << "Tapez une chaîne SVP : "; cin.getline(tt,10);
    cout << "Vous avez tapé la chaîne : ";
    cout << tt << endl;
    return 0;
}
```

- **Explication**
  - Dans cet exemple, nous définissons une chaîne de caractères notées tt.
  - On saisit au clavier une chaîne de caractères pouvant contenir des espaces et qui doit être contenue dans un tableau de 10 char (elle comportera 9 caractères utiles au maximum plus le caractère de fin de chaîne) : elle sera automatiquement tronquée en conséquence.
  - On affiche ensuite la chaîne que l'utilisateur vient de taper. Elle sera éventuellement tronquée.
- **Exécution 1**

Lorsqu'on exécute le programme voici ce qu'on obtient à l'écran :
Tapez une chaîne SVP : **azer ty**

Vous avez tapé la chaîne : azer ty

- **Exécution 2**

Lorsqu'on exécute le programme voici ce qu'on obtient à l'écran :
Tapez une chaîne SVP : **aaaaaaaaaaaaa**
Vous avez tapé la chaîne : aaaaaaaaa

Conclusion

Nous avons étudié comment les chaînes de caractères étaient représentées en C++. Nous avons vu quelques fonctions prédéfinies permettant de manipuler ces chaînes. Nous avons vu comment il était possible de vérifier si une chaîne de caractères était sous un certain format. Il nous reste maintenant à effectuer des exercices pour utiliser ces fonctions et pour manipuler des chaînes de caractères en tout genre.

Exercices

EXERCICE 1

Ecrire une fonction qui a en paramètres une chaîne de caractères (paramètre en entrée) et un entier e (paramètre en sortie). Cette fonction renvoie un booléen. La fonction renvoie true si la chaîne de caractères est un entier écrit sous la forme d'une suite de chiffres qui ne commence pas par 0, elle renvoie false sinon. Si la chaîne est correcte, la fonction renvoie dans e la valeur de cet entier.

EXERCICE 2

Ecrire une fonction qui a en paramètre une chaîne de caractères (paramètre en entrée et en sortie) et qui transforme toutes les minuscules de la chaîne en majuscules.

EXERCICE 3

Ecrire une fonction qui a en paramètre une chaîne de caractères (paramètre en entrée et en sortie) et qui supprime toutes les voyelles.

EXERCICE 4

Ecrire une fonction qui a en paramètres deux chaînes de caractères ch1 et ch2 (paramètres en entrée) et renvoie un booléen indiquant si la chaîne ch2 est contenue dans la chaîne ch1.

EXERCICE 5

Ecrire un programme qui demande à l'utilisateur de taper un verbe du premier groupe et qui le conjugue à l'indicatif présent.

EXERCICE 6

Ecrire un programme qui saisit une chaîne pouvant contenir des espaces et qui affiche chaque mot de la chaîne, le séparateur étant l'espace.
Exemple, on tape : **je pense donc je suis**
Le programme affiche :
mot 1 : je
mot 2 : pense
mot 3 : donc
mot 4 : je

mot 5 : suis

EXERCICE 7

Ecrire un programme qui demande à l'utilisateur de taper une chaîne de caractères et qui indique si cette chaîne est un palindrome ou non.

EXERCICE 8

Ecrire un programme qui demande à l'utilisateur de taper une chaîne de caractères et qui affiche la lettre (minuscule ou majuscule) la plus fréquente.