Programmation C++ (débutant)/La classe string

Le cours du chapitre 12 : La classe string

Présentation de la classe string

- Il s'agit d'une classe standard qui permet de représenter une chaîne de caractères.
- Pour l'utiliser, il faut rajouter #include <string>
- Cette classe encapsule des données pour pouvoir effectuer toutes les opérations de base sur les chaînes.
- Ces opérations sont assez complexes notamment la gestion de la mémoire : l'encapsulation permet de masquer à l'utilisateur de la classe toutes les difficultés techniques.

Différentes opérations sur la classe string

- **Déclaration et initialisation :** string s1; string s2= "BONJOUR";
- **Affichage et saisie :** cout<<s2; cin>>s1;
- **Concaténation :** string s3=s2+s1;

Exemple 1 : la classe string

Fichier main.cpp

```
#include <iostream>
#include <string>

using namespace std;

int main (void)
{
  string s1, s2, s3;

  cout << "Tapez une chaine : "; cin >> s1;
  cout << "Tapez une chaine : "; cin >> s2;
  s3 = s1 + s2;
  cout << "Voici la concatenation des 2 chaines :" << endl;
  cout << s3 << endl;
  return 0;
}
```

Explications

- Dans cet exemple, nous étudions l'utilisation de la classe string.
- On peut saisir le contenu d'un chaîne en utilisant cin.
- On peut concaténer 2 chaînes grâce à l'opérateur +.
- On peut afficher une chaîne grâce à cout.
- Dans cet exemple, on demande à l'utilisateur de saisir 2 chaînes de caractères s1 et s2 et on affiche s3 la concaténation de s1 et de s2.

Exécution

Lorsqu'on exécute ce programme, il s'affiche à l'écran :
Tapez une chaîne : **AZERTY**
Tapez une chaîne : **QSDFGH**
Voici la concaténation des deux chaînes :
AZERTYQSDFGH

Séparateurs

- Par défaut, lorsqu'on saisit une chaîne de caractères en utilisant cin, le séparateur est l'espace : cela empêche de saisir une chaîne de caractères comportant une espace.
- La fonction getline(iostream &,string) permet de saisir une chaîne de caractères en utilisant le passage à la ligne comme séparateur : notre chaîne de caractères peut alors comporter des espaces.

Exemple 2 : string avec des espaces

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

int main (void)
{
  string s1, s2, s3;

  cout << "Tapez une chaine : "; getline (cin, s1);
  cout << "Tapez une chaine : "; getline (cin, s2);
  s3 = s1 + s2;
  cout << "Voici la concatenation des 2 chaines :" << endl;
  cout << s3 << endl;
  return 0;
}
```

Explications

- Dans cet exemple, la chaîne de caractères s1 est saisie grâce à l'instruction getline(cin,s1) : cela permet de saisir au clavier la chaîne s1, la fin de la chaîne est alors indiquée lorsqu'on tape sur ENTREE.
- On saisit ensuite la chaîne s2 et on affiche la concaténation des deux chaînes.

Exécution

Lorsqu'on exécute ce programme, il s'affiche à l'écran :
Tapez une chaîne : **AZ ERTY**
Tapez une chaîne : **QS DFGH**
Voici la concaténation des deux chaînes :
AZ ERTYQS DFGH

Analyse de chaînes

- **Nombre de caractères d'une chaîne :** size() est une méthode de la classe string qui renvoie le nombre de caractères utiles.
- **Récupération du i-ième caractère :** la méthode const char at(int i) permet de récupérer le i-1ième caractère. (0 = 1er)

Exemple 3 : analyse de chaînes

```
#include <iostream>
#include<string>

using namespace std;

int main (void)
{
  string s= "BONJOUR";
  int i, taille = s.size ();

  cout << "La chaine comporte " << taille << " caracteres." << endl;

  for (i = 0 ; i < taille ; i++)
    cout << "caractère " << i << " = " << s.at(i) << endl;
  return 0;
}
```

Explications

- La méthode size() sur un string permet de connaître la taille d'une chaîne de caractères.
- Si i est un entier s.at[i] permet de connaître le (i+1)-ième caractère de la chaîne (s.at[0] étant le premier caractère).
- Dans ce programme, on initialise la chaîne s à "BONJOUR" : on affiche ensuite la taille de la chaîne et on affiche ensuite un à un chaque caractère.

Exécution

Lorsqu'on exécute ce programme, il s'affiche à l'écran :

```
La chaîne comporte 7 caractères
caractère 0 = B
caractère 1 = O
caractère 2 = N
caractère 3 = J
caractère 4 = O
caractère 5 = U
caractère 6 = R
```

Compatibilité avec les char * et les tableaux de char

- **Transformation de chaîne de type C en string :** on peut utiliser le constructeur string(char *) ou l'affectation grâce au symbole = d'un char * vers une string.
- **Transformation d'une string en chaîne de type C :** il suffit d'utiliser la méthode : c_str() qui renvoie un char * qui est une chaîne de type C.

Exemple 4 : compatibilité avec les tableaux de char et les char *

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

int main (void)
{
  string s1, s2;
  char c1 []= "BONJOUR";
  const char * c2;

  s1 = c1;
  cout << s1 << endl;
  s2 = "AU REVOIR";
  c2 = s2.c_str();
  cout << c2 << endl;
  return 0;
}
```

Explications

- Dans cet exemple, c1 est un tableau de 8 char contenant la chaîne "BONJOUR" (n'oubliez pas le caractère de fin de chaîne '\0').
- Le pointeur c2 est un pointeur vers un tableau non modifiable de char.
- Les variables s1 et s2 sont des string.
- On peut affecter directement s1=c1 : le tableau de char sera transformé en string.

Dans c2, on peut récupérer une chaîne « de type C » identique à notre string en écrivant c2=s2.c_str().

- On peut transformer aisément une string en tableau de char et inversement.

Exécution

Lorsqu'on exécute ce programme, il s'affiche à l'écran :
BONJOUR
AU REVOIR

Transformation d'une chaîne en int ou double

- Pour transformer une chaîne en double ou en int, il faut transformer la chaîne en flot de sortie caractères : il s'agit d'un istringstream.
- Ensuite, nous pourrons lire ce flot de caractères en utilisant les opérateurs usuels >>.
- La méthode eof() sur un istringstream permet de savoir si la fin de la chaîne a été atteinte.
- Chaque lecture sur ce flot grâce à l'opérateur >> renvoie un booléen qui nous indique d'éventuelles erreurs.

Exemple 5 : transformation de string en int

```
#include <iostream>
#include <sstream>
#include <string>
using namespace std;
int main (void)
{
  string s;
  cout << "Tapez une chaine : "; getline (cin, s);
  istringstream istr(s);
  int i;

  if (istr >> i) cout << "VOUS AVEZ TAPE L'ENTIER " << i << endl;
  else cout << "VALEUR INCORRECTE" << endl;
  return 0;
}
```

Explications

- Dans cet exemple, s est une chaîne : on saisit une chaîne au clavier en utilisant getline(cin,s).
- On crée ensuite un istringstream appelé istr et construit à partir de s.
- On peut lire un entier i à partir de istr en utilisant : istr>>i .
- (istr>>i) renvoie true si un entier valide a pu être lu et renvoie false sinon.
- De la même manière, on pourrait lire des données d'autres types, double par exemple.

Exécution 1

Lorsqu'on exécute ce programme, il s'affiche à l'écran :
Tapez une chaîne : **12345**
VOUS AVEZ TAPE L'ENTIER 12345

Exécution 2

Lorsqu'on exécute ce programme, il s'affiche à l'écran :
Tapez une chaîne : **BJ9UYU**
VALEUR INCORRECTE

EXERCICES

EXERCICE 1

On veut écrire une classe Note qui est représentée par 3 données membres : 2 chaînes de caractères nom et prénom et un entier v qui est la valeur de la note. Grâce au constructeur par défaut la note est initialisée ainsi : le nom et le prénom sont à la chaîne vide et la valeur note vaut 0.
Pour pouvoir accéder à la valeur du nom, prénom et de la note il y a un accesseur. Un unique mutateur permet de modifier le nom, le prénom et la valeur de la note mais il doit respecter les contraintes suivantes :
Le nom et le prénom ne peuvent pas être la chaîne vide.
La note est comprise entre 0 et 20 bornes incluses
Le mutateur renvoie un booléen true si tout s'est bien passé false sinon.
On peut afficher la note grâce à l'opérateur << et en saisir une grâce à l'opérateur >>. On écrira une classe menu qui gérer notre une note de la manière suivante :

1. Afficher la note
2. Modifier la note
0. Quitter

Écrire le programme principal qui crée notre note et qui appelle notre menu.