Centre Informatique pour les Lettres et les Sciences Humaines

Apprendre C++ avec QtCreator
Etape 6 : Secouons nos chaînes !

Le propos de ce TD est de poursuivre notre entraînement à la création et à l'utilisation de fonctions communiquant entre-elles à l'aide des différents moyens proposés par C++.

1 - A propos des `std::string`

Le type `std::string` n'est pas un type prédéfini, mais une adjonction proposée par la librairie standard (d'où son nom). Les variables de ce type peuvent contenir du texte, c'est à dire des séquences ordonnées de caractères, qu'on appelle communément des chaînes.

Lorsqu'une chaîne apparaît littéralement dans le code source, elle doit figurer entre guillemets :

```
std::string exemple("Une chaîne figurant littéralement dans le code source");
```

Une chaîne peut ne comporter qu'un seul caractère, mais cela ne doit pas conduire à la confondre avec une variable de type `char` qui contiendrait le même caractère (les deux types sont différents).

```
1 std::string uneChaine("x");
2 char unCaractere('x');
```

Les valeurs de type `char` qui apparaissent littéralement dans le code doivent être entre apostrophes.

Déterminer la longueur d'une chaîne

On peut obtenir le nombre de caractères composant une chaîne en appelant la fonction `length()`. Cet appel utilise une syntaxe particulière : au lieu de passer à la fonction la chaîne dont nous souhaitons connaître la longueur, nous invoquons la fonction "au titre de" la variable qui contient cette chaîne :

```
1 std::string duTexte("blabla");
2 int longueurDuTexte(duTexte.length());
```

Une conséquence désagréable de cette façon de procéder est qu'il n'est pas possible de demander à `length()` la longueur d'une chaîne de caractères qui n'est pas contenue dans une `std::string` :

```
3 int x("blablabla".length()); //ERREUR : "blablabla" n'est pas une std::string
```

Le nombre de caractères d'une chaîne ne pouvant en aucun cas être négatif, `length()` renvoie une valeur de type `unsigned` plutôt qu'un `int` ordinaire. Si vous souhaitez évitez de déclencher des messages d'avertissement en provenance du compilateur, vous pouvez faire en sorte que les variables qui seront comparées à la longueur d'une chaîne soient elles-mêmes des `unsigned` et non des `int`.

Isoler l'un des caractères d'une chaîne

Beaucoup de traitements impliquent d'accéder individuellement aux différents caractères que comporte une chaîne. Cet accès est rendu facile par l'utilisation d'une numérotation implicite des caractères : le premier porte le numéro 0, le second le numéro 1, et ainsi de suite.

Un caractère donné est désigné par le nom de la chaîne dont il fait partie, suivi de son numéro d'ordre (on dit aussi son *index*) placé entre crochets :

```
1 std::string histoire("Il était une fois un petit chaperon rouge.");
2 char uneLettre(histoire[0]);
```

La variable `uneLettre` est ici initialisée avec le premier caractère de l'`histoire`, un i majuscule.

Remarquez que, du fait que le premier caractère porte le numéro 0,

Si `uneChaine` comporte `N` caractères, `uneChaine[N]` n'existe pas.

Cette notation peut aussi bien être utilisée pour copier (lire) un caractère de la chaîne que pour modifier (écrire) le contenu de celle-ci :

```
3 histoire[histoire.length() - 1] = '!'; //remplace le point par un point d'exclamation
```

Concaténation

Deux chaînes peuvent être mises bout à bout à l'aide de l'opérateur `+`, qui perd ici sont sens "addition" pour adopter un sens voisin : aboutage (ou, plus communément, concaténation). La variante `+=`, qui permet d'ajouter du texte à la fin d'une chaîne, est également possible, et le texte ajouté à une `std::string` peut aussi n'être qu'un simple `char`.

```
4 histoire += "Et le loup la mangea.";
```

2 – Exercices

(Les deux premiers exercices ont été traités en cours)

a) Écrivez une fonction `debut()` qui renvoie les premiers caractères d'une chaîne (on lui passe la chaîne et le nombre de caractères souhaités).

Exemple : `debut("un deux trois", 7);` renvoie "un deux"

b) Écrivez une fonction `fin()` qui renvoie les derniers caractères d'une chaîne (on lui passe la chaîne et le nombre de caractères souhaités).

Exemple : `fin("un deux trois", 10);` renvoie "deux trois"

c) Écrivez une fonction `milieu()` qui renvoie un extrait d'une chaîne (on lui passe la chaine et les positions du premier et du dernier caractère devant figurer dans l'extrait).

Exemple : `milieu("un deux trois", 3, 6);` renvoie "deux"
Vous pouvez utiliser `debut()` et `fin()` pour définir `milieu()`

d) Écrivez une fonction `remplir()` qui remplit une chaîne en y répétant un même caractère (on lui passe la chaîne à remplir, le caractère à répéter et la longueur souhaitée).

Exemple : après `remplir(uneChaine, "#", 7);` la variable `uneChaine` contient "#######"

e) Écrivez une fonction `trouve()` qui renvoie la première position à laquelle une séquence de caractères est rencontrée dans une chaine (on lui passe la chaine et la séquence à rechercher). Si le fragment recherché n'apparait pas dans la chaîne, la fonction doit renvoyer -1.

Exemple : `trouve("un deux trois", "de");` renvoie 3

f) Écrivez une fonction `remplace()` qui remplace dans une chaine toutes les occurrences d'une séquence de caractères donnée par une autre séquence de caractères (on lui passe la chaîne, la séquence à remplacer et la séquence de remplacement).

Exemple : si `uneChaine` contient "un lapin et un chapeau", après `remplace(uneChaine, "un", "deux");` `uneChaine` contiendra "deux lapin et deux chapeau"

3 - Problème

L'exécution du programme suivant s'est soldée par l'affichage de
`NKEBFGBF_RGYJYRGJKYRGMFKJY`
Quel était le message saisi par l'utilisateur ?

```
5  int main(int argc, char *argv[])
6  {
7  QCoreApplication a(argc, argv);
8  std::string alpha("ABCDEFGH_IJKLMNOPQRSTUVWXYZ");
9  std::string beta("ENIOYMAXGPZWVCLFQHKRJ_BUSDT");
10 std::string texte;
11 std::cout << "Quel est votre message ?\n";
12 std::cin >> texte;
13 traduit(alpha, beta, texte);
14 std::cout << texte;
15 return a.exec();
16 }
   //************************************************************************
17 void traduit(std::string C1, std::string C2, std::string & texte)
18 {
19 unsigned position(0);
20 do {
21   std::string aTraduire;
22   aTraduire += texte[position];
23   int p = trouve(C1, aTraduire); //appel de la fonction de l'exercice e)
24   if (p != -1)
25      texte[position] = C2[p];
26   ++position;
27   } while(position < texte.length());
28 }
```