Tableaux et chaînes de caractères en C++
Méthode utilisant la librairie STL (C++ seulement)

1. Manipulation des chaînes de caractères en utilisant la classe string de la bibliothèque STL

Sans rentrer dans les détails de la bibliothèque standard du C++ (STL) que nous reverrons en détail plus tard, je vais néanmoins vous présenter une classe très pratique qui est la classe string. Grâce à elle, la manipulation des chaînes de caractères devient un jeu d'enfants.

1.1. Voyons déjà un premier exemple :

```
#include <iostream>
#include <string>
#include <conio.h>
#include <sstream>

using namespace std;

int main()
{
        string str1="Super " , str2="la STL et la classe string" , str3;
        str3 = str1 + str2 ;
        cout << "\n str1= "<< str1 << "\n str2= "<< str2 << "\n str3= "<< str3 << endl << flush


        cout <<"3eme lettre de str1 = "<< str1[2] << endl << flush;


        char chaine[100+1];
        strcpy(chaine , str1.c_str() );
        cout << "La chaine C equivalente est "<< chaine <<endl << flush;

        char chainebis[]="Facile";
        string str4(chainebis);
        cout <<"La string equivalente est "<< str4 << endl << flush ;


        int Longueur = str3.length();


        cout <<"\n str3 a "<< Longueur << " caracteres "<< endl << flush;

        str1.swap(str2);      // inversion de str1 avec str2
        cout <<"\nApres inversion de str1 et str2 ";
        cout << "\n str1= "<< str1 << "\n str2= "<< str2 << endl << flush;



        _getch();
        return 0;
}
```

Concaténation très simple par le signe +

Accès à un caractère particulier de la chaîne

Conversion d'une string en chaîne C classique

Conversion d'une chaîne C classique en string (passage en paramètre dans le constructeur)

Appel d'une méthode de la classe string pour trouver la longueur

1.2. Tableau résumé

Accès aux prototypes	exemple d'utilisation	entier renvoyé ou nouvelle valeur de ch
exemple de déclaration	`string ch("Bonjour les SNIR");`	`Bonjour les SNIR`
longueur d'une chaîne	`ch.length()`	16
comparaison de chaînes	`ch.compare("cd")`	-1
modification de l'objet		
concaténation (ajout en fin de chaîne)	`ch += " DE CHARLES DE FOUCAULD"`	`Bonjour les IRIS DE CHARLES DE FOUCAULD`
insertion	`ch.insert(12,"BTS ")`	`Bonjour les BTS IRIS DE CHARLES DE FOUCAULD` (insertion à partir de l'indice 12)
suppression de sous-chaine	`ch.erase(8,4)`	`Bonjour BTS IRIS DE CHARLES DE FOUCAULD` (effacement à partir de la position 8 de 4 caractères)
remplacement d'un caractère	`ch[8]='S'`	`Bonjour STS IRIS DE CHARLES DE FOUCAULD` (utilisation directe des [])
remplacement de sous-chaine	`ch.replace(12,4,"Info et Reseaux")`	`Bonjour STS Info et Reseaux DE CHARLES DE FOUCAULD` (remplacement, à partir de l'indice 12, de 4 caractères)

lecture d'une chaîne
fonction externe à la classe
caractère séparateur \n par défaut
	`string s;`
	`getline(cin,s);`
	`getline(cin,s,'!');`
		on tape au clavier: `toto ! titi`
		→ s : `toto ! titi`
		→ s : `toto`

recherche : Chaine de référence = `"Bonjour STS Info et Reseaux DE CHARLES DE FOUCAULD"`	`01234567890123456789012345678901234567890123456789`	
d'une sous-chaîne	`ch.find("STS")`	8
d'une sous-chaîne	`ch.find("zzz")`	string::npos **(valable pour toutes les fonctions de recherche)** (soit zzz n'est pas une sous-chaîne)
d'une sous-chaîne, depuis la fin	`ch.rfind("CH")`	31
d'un caractère	`ch.find_first_of('o')`	1
d'un caractère, depuis la fin	`ch.find_last_of('o')`	15 ('o' de Info)
d'un caractère d'un ensemble	`ch.find_first_of ("yiaueo")`	1 (soit la première voyelle en minuscule)
d'un caractère d'un ensemble, depuis la fin	`ch.find_last_of ("yiaueo")`	25 ('u' de Reseaux) (soit la dernière voyelle en minuscule)
d'un caractère hors ensemble	`ch.find_first_not_of ("Aabb")`	1 (soit le premier 'o')
d'un caractère hors ensemble, à partir de la fin	`ch.find_last_not_of ("CAULD")`	44 (1er 'U' de FOUCAULD)
obtention		
d'une sous-chaîne	`ch.substr(8,3)`	`STS` (soit une sous-chaîne de 3 caractères)
d'un caractère	`ch[3]`	`j`
de la référence en mémoire	`ch.c_str()`	`0x18a00c48` `Conversion en chaine compatible C`

Les fonctions de recherche qui échouent retournent string ::npos

On peut ajouter un 2ème argument pour indiquer à partir de quelle position on cherche.

Exercices :

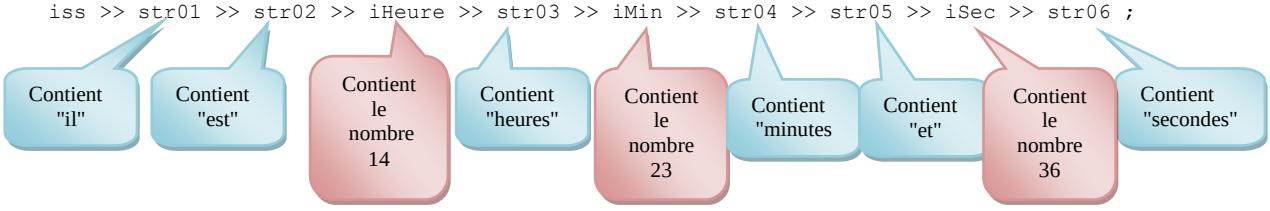
Pour faire ces exercices, inspirez-vous du tableau précédent ou consultez la documentation de la classe string ou basic_string

- Dans une chaîne de caractères saisie au clavier (type string), cherchez la position du premier caractère qui est une voyelle.
- Extraire les caractères 5 à 9 d'une chaîne de caractères saisie au clavier, les stocker dans une autre chaîne, puis les afficher.
- Dans une phrase saisie au clavier comportant plusieurs fois le mot "titi", remplacez tous les "titi" par "toto".

1.3. Conversion d'un nombre en chaîne ou vis-versa

Pour cela, il faut utiliser un gestionnaire de flux (un objet de la classe ostringstream pour créer un chaîne à partir d'un nombre ou un objet de la classe istringstream pour créer un nombre à partir d'une chaîne).

```
#include <sstream>     // à ajouter au début du programme

 ...

//========== Conversion d'un nombre en chaîne ====================
ostringstream oss;
double Nombre=3.14159265 ;

// écrire un nombre dans le flux
oss << Nombre;

// récupérer une chaîne de caractères
string strPi = oss.str();

cout << "\nPi vaut "<< strPi << endl << flush ;

int PosVirgule;
PosVirgule = strPi.find("."); // Recherche de la position de la virgule
cout << "\nLe 4eme chiffre apres la virgule est "<< strPi[PosVirgule + 4];


//============= Extraction de nombres contenus dans une chaîne ========
string str01 , str02, str03, str04 , str05 , str06;
int iHeure, iMin , iSec ;
string strHeure="Il est 14 heures 23 minutes et 36 secondes ";

istringstream iss(strHeure);
iss >> str01 >> str02 >> iHeure >> str03 >> iMin >> str04 >> str05 >> iSec >> str06 ;
```




```
unsigned NbTotalSecondes;
NbTotalSecondes=iHeure*3600 + iMin*60 + iSec ;
       cout <<"\nNombre total de secondes = "<< NbTotalSecondes << endl <<flush;
```

2. Gestion des tableaux en utilisant la classe vector de la bibliothèque STL

Nous venons de voir une classe nommée string qui permet de simplifier énormément l'utilisation des chaînes de caractères. Dans ce chapitre, nous allons aborder une autre classe nommée vector qui permet de simplifier l'utilisation des tableaux.

Les problèmes posés par les tableaux classiques :

- La taille des tableaux doit être définie au départ. En conservant les tableaux classiques, Il existe une solution nommée "allocation dynamique de mémoire" que nous aborderons plus tard qui permet de choisir la taille d'un tableau lors de l'exécution. Mais en utilisant la classe ***vector***, la taille est vraiment dynamique, c'est-à-dire que le tableau pourra être agrandi même après que sa taille ait été définie.

- La suppression ou l'ajout d'éléments dans un tableau nécessite de nombreux décalages (à droite pour agrandir ou à gauche pour supprimer) qui sont des opérations fastidieuses. Avec la classe vector, la suppression ou l'ajout d'un élément dans un tableau se fait en une ligne.

Plutôt que de longs discours, voici quelques exemples qui montrent comment utiliser simplement cette classe[1].

[1] la bibliothèque STL permet de parcourir les éléments d'un tableau (ou d'un conteneur, de façon plus générale en utilisant des itérateurs. Nous reverrons ceci dans un chapitre spécifique sur la bibliothèque STL).

Exemple 1 : un tableau d'entiers

```
#include <iostream>
#include <conio.h>
#include <vector>


using namespace std ;

int main( )
{
        unsigned int i;

        // Création d'un tableau de 5 entiers
        vector<int> Tab(5);

        // Affichage du tableau
        cout <<"Affichage du tableau apres sa creation  " << endl;
        for (i=0 ; i<5 ; i++)
                cout << Tab[i] <<" \t" << flush;

        // Remplissage du tableau à votre guise
        Tab[0]=7; Tab[1]=11 ; Tab[2]=3; Tab[3]=4 ; Tab[4]=26;

        // Affichage du tableau
        cout <<"\nAffichage du tableau apres remplissage " << endl;
        for (i=0 ; i<5 ; i++)
                cout << Tab[i] <<" \t" << flush;

        // Agrandissement du tableau
        // Ajout d'un élément à la fin
        Tab.push_back(49) ;

        // Ajout d'un élément au début
        Tab.insert(Tab.begin( ) , 37);

        // Ajout d'un élément en 3ème position
        Tab.insert(Tab.begin( ) + 2  ,  24);

        // Affichage du tableau
        cout <<"\nAffichage du tableau apres ajout " << endl;
        for (i=0 ; i<Tab.size() ; i++)
                cout << Tab[i] <<" \t" << flush;

        // Suppression du 4eme élément
        Tab.erase(Tab.begin( ) + 3 );

        // Affichage du tableau apres suppression du 4eme élément
        cout <<"\nAffichage du tableau apres suppression du 4eme element " << endl;
        for (i=0 ; i<Tab.size() ; i++)
                cout << Tab[i] <<" \t" << flush;


        // Effacement complet du tableau
        Tab.clear( );
        cout <<"\nAffichage du tableau apres effacement total" << endl;
        for (i=0 ; i<Tab.size() ; i++)
                cout << Tab[i] <<" \t" << flush;


        _getch();
        return 0 ;
}
```

Attention !
5 est entre **parenthèses,** pas entre **crochets !**

Contrairement aux tableaux classiques, les tableaux de type vector sont initialisés à 0 à la création

Tab.begin() fait référence au 1er élément du tableau

La méthode size() permet de récupérer le nombre d'éléments du tableau

Tab.begin() + 3 fait référence au 4ème élément du tableau

```
c:\mes documents\travail\cours_07_08\stsi1\tp\stl_vector\debug\STL_vector.exe
Affichage du tableau apres sa creation
0       0       0       0       0

Affichage du tableau apres remplissage
7       11      3       4       26

Affichage du tableau apres ajout
37      7       24      11      3       4       26      49

Affichage du tableau apres suppression du 4eme element
37      7       24      3       4       26      49

Affichage du tableau apres effacement total
```

Exemple 2 : un tableau de chaînes de caractères

```
#include <iostream>
#include <conio.h>
#include <vector>
#include <string>

using namespace std ;

int main( )
{
        unsigned int i;

        // Création d'un tableau de chaines de caractères
        vector<string> TabEtudiants ; // Tableau de chaines vide au départ

        // Ajout des étudiants
        TabEtudiants.push_back("Toto");
        TabEtudiants.push_back("Titi");
        TabEtudiants.push_back("Tutu");

        // Affichage des étudiants
        cout <<"Affichage des etudiants " << endl;
        for (i=0 ; i<TabEtudiants.size() ; i++)
                cout << TabEtudiants[i] <<" \t" << flush;

        // Comme il y a des homonymes, on accole au bout de chaque nom d'étudiant le
        // nom de la classe. Toto devient Toto_STSI1, Titi devient Titi_STSI1, etc.
        for (i=0 ; i<TabEtudiants.size() ; i++)
                TabEtudiants[i].insert(TabEtudiants[i].size() , "_STSI1");

        // Affichage des étudiants
        cout <<"\n\nAffichage des etudiants apres ajout de la classe" << endl;
        for (i=0 ; i<TabEtudiants.size() ; i++)
                cout << TabEtudiants[i] <<" \t" << flush;

        // Max est un nouvel etudiant en STSI2 et Titi démissionne
        TabEtudiants.push_back("Max_STSI2");
        TabEtudiants.erase(TabEtudiants.begin( ) + 1);

...
}
```

C'est un tableau de **string**

```
c:\mes documents\travail\cours_07_08\stsi1\tp\stl_vector\debug\STL_vector.exe
Affichage des etudiants
Toto    Titi    Tutu

Affichage des etudiants apres ajout de la classe
Toto_STSI1      Titi_STSI1      Tutu_STSI1

Affichage des etudiants apres arrivee de Max et depart de Titi
Toto_STSI1      Tutu_STSI1      Max_STSI2

Affichage des etudiants de STSI1 seulement
Toto_STSI1      Tutu_STSI1
```

Appel de la méthode **insert** de la classe string

push_back stocke à la fin