Manipulation de Trame

Durée: 6h

Objectifs:

- Manipuler les méthodes de la librairie '**string**'
- Développer une nouvelle classe C++, **TrameManip**, exploitable pour des applications ultérieures.
- Mise en oeuvre d'un objet de type **TrameManip** dans une application graphique Qt.

But:

- Composer et décomposer des chaînes de caractères intégrant des données.

Remarque: Commentez au maximum votre fichier pour une meilleure compréhension et servez-vous des aides (Qt et pages web) pour utiliser les nouvelles fonctions.

I) Présentation

Il est souvent pratique, en informatique, de transmettre ou de recevoir des données sous forme d'une chaîne de caractères. Cette chaîne sera composée d'une ou plusieurs informations de type texte délimitées par un code définit auparavant.

Exemple d'une trame composée d'informations.

"<ip>192.168.0.13</ip><port>80</port><nbElement>185</nbElement>"

Le but est de récupérer facilement les informations encapsulées dans la trame.

On se propose de définir une classe **TrameManip** qui nous permettra facilement de composer ou de décomposer ce type de trame.

TrameManip
 # trame : string **# headBalise : string** **# endBalise: string**
+ TrameManip (dataTrame : string = "") **+ setTrame (dataTrame : string) : void** **+ setHeadBalise (balise : string) : void** **+ setEndBalise (balise : string) : void** **+ getData (idHead : string = "" , idEnd : string = "" , dataTrame : string = "") : string** **+ setData (data : string, idHead : string = "" , idEnd : string = "" , dataTrame : string = "") : string** **+ modifData (data : string , idHead : string = "" , idEnd : string = "", dataTrame: string = "") : bool** **+ supprimeData (idHead : string = "" , idEnd : string = "" , dataTrame : string = "") : string** **+ getTrame ( ) : string**

Remarque : La classe string fait partie de la bibliothèque STL (Standard Template Library).

- Les strings sont des chaînes de caractères avec une gestion de la mémoire et des méthodes de gestion intégrés.
- Pour l'utiliser, il faut rajouter : #include <string>
  using namespace std;

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;

Voici les principales méthodes de cette classe:

Constructeurs de la classe string (STL)	Notes
string();	
string(const char *s);	A partir d'un char *.
string(const char *s, size_t n);	A partir d'un char * avec au plus n caractères.
string(size_t n, char c);	Repète le caractere 'c' n fois.
string(const string &str, size_t pos=0, size_t n=-1);	A partir d'une sous chaine, avec n caractères à partir de pos.

Méthodes de la classe string (STL)	Notes
const char *c_str() const;	Conversion du string vers un char*.
size_t length() const;	Renvoie la longueur de la chaine.
bool empty() const;	Indique si la chaine est vide. (Vrai = vide).
string &assign(const string &str, size_t pos, size_t n); string &assign(size_t rep, char c);	Affecte une sous chaine ou une succession du même caractère.
size_t copy(char *s, size_t n, size_t pos=0);	Copie n caractères à partir de pos de s.
string substr(size_t pos=0, size_t n=-1) const;	Extrait une sous chaine à partir de pos avec au plus n caractères.
int compare(size_t po1, size_t n1, const string &str, size_t po2, size_t n2)const;	Compare la sous chaine de n1 caractères à partir de po1 avec une sous chaine str composé de n2 caractères à partir de pos2. La valeur retournée est: <0, =0 ou >0.
void swap(string &str);	Permute les deux chaines de caractères.
string &append(const string &str, size_t pos, size_t n); string &append(size_t rep, char c);	Ajoute une sous chaine ou une succession de caractères.
string &insert(size_t pos1, const string &str, size_t pos2, size_t n); string &insert(size_t pos, size_t rep, char c);	Insère une sous chaine dans une chaine.
string &replace(size_t pos1, size_t n1, const string &str, size_t pos2, size_t n2); string &replace(size_t pos, size_t n1, size_t n2, char c);	Replace une sous chaine par une autre sous chaine.
string &erase(size_t pos, size_t n);	Efface n caractères de la chaine à partir de pos.

Méthodes de recherche de la classe string (STL)	**Notes** : Les méthodes de recherches renvoie -1 si l'occurrence n'est pas trouvée.

size_t find(const string &str, size_t pos=0) const;
size_t find(char c, size_t pos=0) const;
size_t find_first_of(const string &str, size_t pos=0) const;
size_t find_first_of(char c, size_t pos=0) const;
	Recherche de la première occurrence de str ou c à partir de pos.
	Renvoie le rang de l'occurrence.

size_t find_first_not_of(const string &str, size_t pos=0) const; size_t find_first_not_of(char c, size_t pos=0) const;	Recherche de la première occurrence d'un caractère NON SPECIFIE de str ou c à partir de pos. Renvoie le rang de l'occurrence.

size_t rfind(const string &str, size_t pos=-1) const;
size_t rfind(char c, size_t pos=-1) const;
size_t find_last_of(const string &str, size_t pos=-1) const;
size_t find_last_of(char c, size_t pos=-1) const;
	Recherche de la dernière occurrence de str ou c à partir de pos.
	Renvoie le rang de l'occurrence.

size_t find_last_not_of(const string &str, size_t pos=-1) const; size_t find_last_not_of(char c, size_t pos=-1) const;	Recherche de la dernière occurrence d'un caractère NON SPECIFIE de str ou c à partir de pos. Renvoie le rang de l'occurrence.

II) Travail demandé

1) Déclarer un nouveau projet, de type Application graphique Qt, ayant pour nom :
"votreNomGestionTrame"

2) Etablir la nouvelle classe **TrameManip** .
- Définir la classe, dans un fichier **trameManip.h**
- Développer toutes les méthodes de la classe, dans un fichier **trameManip.cpp**

3) Développer votre application comme dans l'IHM ci-dessous pour tester votre nouvelle classe **trameManip**