

# INSTITUT UNIVERSITAIRE DE TECHNOLOGIE D'AIX EN PROVENCE

Projet tuteuré C++

## ACRONYME

Définition d'une famille de classes permettant la gestion des personnes  
Utilisation pour la mise à jour d'un fichier de personnes

P. Sébilo en Février 2005

---

### MISE EN ŒUVRE

L'objectif de ce projet est de réaliser un travail de synthèse et de révision d'un maximum de notions élémentaires que nous avons abordées en cours de manière dispersée. La plupart des algorithmes mis en œuvre ont déjà été résolus en cours.

Ce projet pourra être réalisé en solo ou en binôme. Bien entendu, je serai plus exigeant pour les étudiants qui travaillent en binôme.

Il est fortement recommandé de vous mettre au travail le plus rapidement possible et de ne pas attendre le dernier moment pour rendre votre dossier.

Ce projet comporte trois parties qui devront être traitées successivement :

- . Définition des classes dont les déclarations sont imposées.
- . Utilisation de ces classes dans 3 programmes de mise à jour d'un fichier.
- . Complémentation des classes sans directives précises

Vous remarquerez que pour simplifier, les exceptions ne sont pas volontairement traitées. On vous demandera de les ajouter dans un deuxième temps. Même remarque pour les fichiers inclus.

Il est fortement conseillé de tester les classes avant de les utiliser dans les exercices de la fin. Pour cela, vous devez écrire pour chacune des classes un programme qui teste l'ensemble des fonctions avec un ensemble de paramètres choisis de la manière la plus exhaustive possible. A titre d'exemple, je vous fournis mon programme de test de la classe CNom que vous pouvez utiliser pour tester votre propre classe (fichier TNom.cxx) .

Ce travail débouchera sur un dossier papier et un dossier informatique :

- . Le dossier papier comportera les listings demandés ainsi que les explications nécessaires à la compréhension des programmes. Il devra être relié et comporter sur la première page toutes les informations indispensables pour pouvoir identifier les auteurs et la nature du sujet traité. L'analyse de ce dossier, pourra être éventuellement complété par un oral où vous serez amenés à me fournir toutes explications supplémentaires sur les méthodes choisies ainsi que sur votre travail personnel.

. Le dossier informatique devra comporter ces mêmes programmes stockés dans un répertoire nommé cppps créé directement dans votre répertoire racine. Ces programmes devront pouvoir être testés avec mes propres programmes de test. Pour que cela soit possible, vous ne devez pas modifier les fichiers de déclarations qui vous sont fournis.

Les deux dossiers devront être rendus au plus tard le **Vendredi 19 Mars 2005** au secrétariat. Tout retard sera sanctionné au niveau de la note.

Pensez à bien documenter vos listings et à soigner la présentation, j'en tiendrai le plus grand compte dans la note.

Des informations supplémentaires pourront être fournies par mail.

Bien entendu, vous pouvez me consulter pour toute demande d'explications supplémentaires.

# PREMIERE PARTIE : LES CLASSES

## 1. Classe CString

### Listings à fournir : CString.h et CString.cxx

Nous avons déjà étudié en cours une version réduite de la classe CString qui dérive de la classe std::string. Nous devons compléter cette classe pour traiter les questions qui vont suivre :

#### **Question 1 :**

Ajouter la fonction non membre :  
string UnsignedToString (const unsigned i) ;  
qui transforme un unsigned decimal en string.

#### **Question 2 :**

Ajouter la fonction non membre :  
unsigned StringToUnsigned (const string & Str) ;  
qui transforme une chaîne supposée contenir l'image d'un unsigned décimal ;  
(utiliser l'algorithme d'Horner)

#### **Question 3 :**

Procédure de lecture contrôlée :  
En cours, nous avons réalisé une procédure de lecture 'à la volée'.  
Nous allons compléter cette procédure en contrôlant le type des caractères saisis.

Pour cela, nous devons définir une procédure où la fonction qui valide les caractères saisis est passée en paramètre. Lorsqu'on passe une fonction en paramètre, on passe en réalité un pointeur sur la fonction dont l'identificateur est le nom de cette fonction. Le problème c'est qu'il n'existe pas de type de pointeur standard sur les fonctions. Le type du pointeur dépend du type de la fonction, du nombre et du type de ses paramètres. On vous propose donc les déclarations suivantes :

```
typedef bool (* FctValidChar_t) (const char) ;  
// FctValidChar_t est donc le type pointeur de fonction de type booleen (ou prédicat)  
qui ont un paramètre donné de type char.
```

```
Void LireString ( FctValidChar_t isGood, const bool AvecEcho = true) ;  
// Fonction membre de la classe CString  
// Les caractères non conformes sont refusés (avec sonnerie)  
// Possibilité de retour en arrière mais pas au - delà de la première position.  
// Fin de saisie exclusivement sur <cr>
```

Exemple d'utilisation (pour lire une chaîne en majuscules) :

```
CString MaStr ;  
MaStr.LireString (isupper) ;
```

## 2. Classe CNom

### Listing à fournir : CNom.cxx

#### Question 4 : Définition de la classe CNom

Nous allons définir une classe Nom (voir fichier CNom.h) :

Pour simplifier, pour le moment, un nom est constitué d'un seul mot.

Ce mot est une chaîne alphabétique.

Ce Nom est NORMALISER (voir classe CDuree) à la construction en une chaîne en majuscules.

La procédure de lecture doit utiliser la procédure LireString précédente en n'acceptant que les lettres (majuscules ou minuscules). Le nom sera normalisé ensuite.

Les opérateurs de relation sont surchargés comme d'habitude.

La surcharge de l'injecteur est réalisée avec une fonction 'friend' (non membre de la classe) qui renvoie le flux modifié. La définition de cette fonction est très simple :

```
ostream & operator << (ostream & flux, const CNom & Nom)
{ return flux << Nom.m_Nom ; }
```

La fonction de validation (non membre) EstStrNom contrôle si la chaîne passée en paramètre est alphabétique.

## 3. Classe CPrenom

### Listing à fournir : CPrenom.cxx

#### Question 5 : Définition de la classe CPrenom

Nous allons définir une classe Prenom (voir fichier CPrenom.h) :

La classe CPrenom a la même structure que la classe CNom sauf que le prénom doit être en minuscules.

## 4. Classe CIdent

### Listing à fournir : CIdent.cxx

#### Question 6 : Définition de la classe CIdent

Nous allons définir la classe CIdent (voir fichier CIdent.h)

La classe CIdent est constituée d'un nom et d'un prénom plus un ACRONYME qui sera calculé à la construction.

On peut utiliser plusieurs images de l'identité.

Les relations d'ordre concernent les acronymes.

## 5. Classe CCpt

### **Listing à fournir : CCpt.h CCpt.cxx**

#### **Question 7 : Déclaration et définition de la classe CCpt**

Il nous faut définir maintenant une classe compteur qui sera utilisée plus loin.  
On vous demande de concevoir entièrement cette classe afin qu'elle puisse répondre le plus simplement possible aux besoins suivants :

- . Une donnée membre m\_Compteur qui puisse être incrémenté ou décrémenté.
- . Une décrémentation sur un compteur nul sera sans effet.

#### 6. Classe CPerso

### **Listing à fournir : CPerso.cxx**

#### **Question 8 : Définition de la classe CPerso**

Nous allons définir la classe CPerso (voir fichier CPerso.h)

La classe CPerso dérive de la classe CIdent qui est la classe mère de toute les classes qui ont besoin de gérer une identité. En plus de l'identité, la classe CPerso est constituée d'un compteur (qui pourrait représenter une note, par exemple) et d'un classement fonction du compteur.

Les modifieurs permettent d'incrémenter le compteur et de changer le classement (voir plus loin).

La relation d'ordre se fait par rapport au classement. Vous pouvez ajouter les relations absentes.

La surcharge de l'injecteur fournit les données dans une présentation agréable.

La fonction non membre StringToPerso extrait les différentes données de la chaine Str. Elle pourra être utilisée par l'injecteur.

## **DEUXIEME PARTIE : LE FICHIER**

Il s'agit maintenant de gérer un fichier de personnes en utilisant les classes ci-dessus. L'ensemble des personnes est mémorisé dans le fichier texte PersoIn à raison d'une ligne par personne. A l'intérieur de la ligne, les données sont séparées par au moins un espace (on pourra utiliser l'extracteur). On trouvera successivement : Nom, Prénom, Compteur, Classement. Les lignes sont triées dans l'ordre alphabétique des noms puis des prénoms (en fait, selon les acronymes).

### **1. Mise à jour du classement**

#### **Listing à fournir : Acro1.cxx**

Ce programme consiste à saisir l'acronyme d'une personne, à incrémenter son compteur et à visualiser son classement. Bien entendu, à chaque modification du compteur, le classement doit être mis à jour.

Pour bien comprendre ce que vous devez réaliser, je vous propose d'exécuter mon programme Acronyme1.

On vous demande de réaliser les opérations dans l'ordre où elles sont présentées.

#### **Question 8 : Mémorisation en mémoire dynamique**

Lire le fichier d'entrée PersoIn ligne par ligne et le stocker en mémoire dynamique tout en créant un vector de pointeurs comme cela a été réalisé en TP. Ce premier vector permet donc d'accéder aux personnes dans l'ordre alphabétique puisque le fichier est trié sur les acronymes.

Afficher les personnes dans l'ordre alphabétique pour contrôler.

#### **Question 9 : Recherche dichotomique**

Faire une boucle de saisie sur l'acronyme. Rechercher la personne en faisant une recherche dichotomique dans le vector de pointeurs. La version de la recherche dichotomique étudiée en cours devra être modifiée pour tenir compte des pointeurs.

En cas de succès, afficher la personne, en cas d'échec, afficher un message d'erreur.

#### **Question 10 : Vector de classement**

Ajouter un deuxième vector de pointeurs qui permet d'accéder aux personnes dans l'ordre du classement. Il y a donc deux vectors de pointeurs pour accéder aux données qui elles ne sont mémorisées qu'une seule fois.

Afficher les personnes dans l'ordre du classement pour contrôler.

#### **Question 11 : Mise à jour**

Il s'agit maintenant de modifier le compteur en mettant à jour le classement.

Revenir à la question 9 et incrémenter le compteur de la personne lorsque celle-ci a été trouvée. Mettre à jour alors son classement et visualiser les données de la personne

modifiée. Attention, cette opération nécessite la mise à jour du vecteur de classement pour pouvoir accéder aux personnes dans l'ordre du classement.

### **Question 12 : Accès au classement**

Faire une boucle de lecture des classements. Contrôler la validité et afficher la personne correspondante.

### **Question 13 : Menu**

Regrouper les opérations précédentes dans un menu qui permettra de :

- . Visualiser une personne d'un acronyme donné.
- . Incrémenter son compteur.
- . Visualiser une personne d'un classement donné.

### **Question 14 : Sauvegarde**

.Ajouter un fichier Journal qui mémorise l'ensemble des opérations réalisées comme on l'a fait en TD pour l'application éditeur.

- . Sauvegarder les données dans l'ordre alphabétique dans le fichier PersoOut.

## **2. Ajout de personnes**

### **Listing à fournir : Acro2.cxx**

### **Question 15 : Ajout de personnes**

Faire une boucle de lecture qui saisit le nom et le prénom d'une nouvelle personne.

Par recherche dichotomique, vérifier son acronyme.

Si l'acronyme existe déjà, refuser l'ajout.

Insérer la nouvelle personne en respectant l'ordre alphabétique.

Bien entendu, le compteur des nouvelles personnes est nul.

Faire attention au classement.

Faire un journal et sauvegarder.

Exécuter de nouveau le programme Acro1 pour contrôler.

## **3. Suppression de personnes**

### **Listing à fournir : Acro3.cxx**

### **Question 16 : Suppression de personnes**

Saisir le nom des personnes à supprimer et mettre à jour.

Pensez à mettre à jour le classement avant la sauvegarde.

## **TROISIEME PARTIE : COMPLEMENTS**

Cette partie ne pourra être envisagée que lorsque tout ce qui précède sera terminé.  
Elle est facultative et ne sera comptée qu'en bonus (3 points au maximum) .  
A la différence des questions précédentes, l'énoncé est assez libre.

Pour rendre ce programme exploitable, vous voyez bien qu'il faut compléter les classes.

Vous pouvez donc ajouter (dans l'ordre d'importance) :

1. Traiter les exceptions.
2. Compléter la classe CNom en gérant des noms composés.
3. Compléter la classe CPrenom en gérant plusieurs prénoms (3 au maximum).
4. Ajouter une classe CAdresse.
5. Autres améliorations à votre convenance.