





TP12 – Premières classes

 L'objectif de ce TP très scolaire est de vous familiariser avec la syntaxe du C++ pour la création de classes.

Exercice 0 Mise en place du TP

-  Créez une nouvelle solution Visual Studio.
-  Vous utiliserez la même solution pour tout le TP.






Exercice 1 Votre première classe et vos premiers objets

 Nous allons implémenter dans cet exercice une première classe **CEtudiant**.


1.1 Présentation de la classe CEtudiant

Cette classe gèrera un étudiant. Chaque étudiant se voit doté d'un nom, un prénom et d'une date de naissance.

La classe aura donc en données membres *privées* le nom, le prénom et la date de naissance de l'étudiant stockés dans des objets de type **std::string**.

-  Écrivez la déclaration de cette classe dans un nouveau fichier que vous nommerez **etudiant.h**. Pour ajouter un nouveau fichier, faites un clic droit sur l'icône de votre projet dans l'explorateur de solution puis choisissez **Ajouter / Nouvel élément...** puis **Visual C++ / Code / Fichier d'en-tête (.h)**
 -  Par défaut, le nom de l'étudiant sera « **Sans nom** », son prénom sera vide, et sa date de naissance sera « **inconnue** ».
-  Ajoutez à cette classe une méthode publique **AfficherTexte()**
 -  Vous devez déclarer cette fonction dans le fichier **Etudiant.h**
 -  Vous devez définir cette fonction dans le fichier **Etudiant.cpp** de façon qu'elle affiche à l'écran de la console :

```
Nom : nom de l'étudiant
Prénom : prénom de l'étudiant
Date de naissance : date de naissance de l'étudiant
```

-  Dans votre programme principal, créez un objet de type **CEtudiant** puis affichez-le à la console. Vérifiez que les valeurs par défaut sont correctes.

1.2 Amélioration de la classe

Il n'est pas pratique que tous les étudiants s'appellent « **Sans nom** », nous allons alors modifier la classe **CEtudiant**.

- ☞ Munissez la classe **CEtudiant** d'un constructeur à trois paramètres, correspondant respectivement au nom, prénom et date de naissance de l'étudiant.
- ☞ Testez dans votre programme principal ce constructeur en utilisant vos nom, prénom et date de naissance.
 - ➡ Utilisez la méthode **AfficherTexte()** pour valider le contenu de la classe.

Exercice 2 Une promotion

- ☞ Ecrivez un programme principal qui présente le menu suivant :

```
'a' : Ajouter étudiant
's' : Supprimer étudiant
'p' : Affiche la promotion
'q' : Quitter
Votre choix : _
```

- ➡ L'appuie sur la touche 'a' ajoute un étudiant dans la promotion après avoir demandé le nom, prénom et date de naissance de l'étudiant.
 - ➡ La promotion peut être un **std::vector** de **CEtudiant**
- ➡ L'appuie sur la touche 's' supprime un étudiant de la promotion après avoir demandé son nom.
- ➡ L'appuie sur la touche 'p' affiche la liste des étudiants de la promotion
- ➡ L'appuie sur la touche 'q' quitte le programme.

Exercice 3 Une meilleure gestion de la date

Pour le moment, la date de naissance de l'étudiant est stockée dans une chaîne de caractères. Cette solution est sujette aux erreurs de saisie. Afin de pallier ce problème, nous allons construire un ensemble de classes permettant de mieux gérer les dates.

3.1 Gestion des mois

Les numéros des mois sont strictement compris entre 1 et 12. Nous allons créer une classe qui s'en assurera.

- ☞ Créez une classe **CMonth** munie de :
 - ➡ Une donnée membre numérique privée permettant de stocker le numéro du mois. Choisissez le type de donnée de ce membre sagement.
 - ➡ Un constructeur avec un paramètre dont l'objectif sera d'initialiser cette dernière donnée membre. Dans le cas où le nombre passé en paramètre ne serait pas valide, la donnée membre sera initialisée à la valeur valide la plus proche.
 - ➡ Une fonction membre publique **getNum()** retournant le numéro du mois.
 - ➡ Une fonction membre publique **getNbDays()** retournant le nombre de jours maximum du mois (c'est-à-dire 29 pour février).
 - ➡ Une fonction membre publique **toString()** retournant le nom du mois.
- ☞ Testez rapidement cette classe dans le programme principal.

3.2 Gestion des années

Les années sont des nombres entiers strictement positifs. Voici comment créer une classe de gestion d'une année.

- ☞ Créez une classe **CYear** munie de :
 - ☞ Une donnée membre numérique privée permettant de stocker l'année. Choisissez le type de donnée de ce membre sagement.
 - ☞ Un constructeur avec un paramètre dont l'objectif sera d'initialiser cette dernière donnée membre.
 - ☞ Une fonction membre publique **getNum()** retournant le numéro de l'année.
 - ☞ Une fonction membre publique **isLeap()** retournant **true** si l'année est bissextile.
Rappel : une année est bissextile si elle est multiple de 4 à l'exception des années multiples de 100, non multiples de 400. Ainsi 2008 est bissextile, 2000, 1600 sont également bissextiles, mais 2001, 1900, 1800 ne sont pas bissextiles.
 - ☞ Il serait sage également que le caractère bissextile de l'année ne soit déterminé qu'une seule fois au moment de la création de l'instance de **CYear**.
 - ☞ Une fonction membre publique **toString()** retournant l'année sous forme de chaîne de caractères.
- ☞ Testez rapidement cette classe dans le programme principal.

3.3 Gestion d'une date

Grâce aux deux classes précédentes, il est plus aisé de créer une classe de gestion de date.

- ☞ Créez une classe **CDate** munie de :
 - ☞ Une donnée membre numérique privée pour stocker le numéro du jour. Choisissez le type de donnée de ce membre sagement.
 - ☞ Une donnée membre privée de type **CMonth** pour stocker le mois.
 - ☞ Une donnée membre privée de type **CYear** pour stocker le mois.
 - ☞ Un constructeur par défaut initialisant l'année au 1/01/1970.
 - ☞ Un constructeur avec trois paramètres numériques correspondant respectivement au jour, mois et année de la date.
 - ☞ Le constructeur devra valider le jour en vérifiant qu'il est bien compris entre 1 et 28, 29, 30 ou 31 selon le mois et l'année spécifiés. Dans le cas où le jour est invalide, il devra être défini à la valeur valide la plus proche.
 - ☞ Une fonction membre publique **getDay()** retournant le numéro du jour.
 - ☞ Une fonction membre publique **getMonth()** retournant l'objet du mois.
 - ☞ Une fonction membre publique **getYear()** retournant l'objet mois de l'année.
 - ☞ Une fonction membre publique **toString()** retournant la date sous forme de chaîne de caractères.
- ☞ Testez rapidement cette classe dans le programme principal.

3.4 Intégration de la classe CDate dans le code

- ☞ Modifiez la classe **CEtudiant** pour intégrer la classe **CDate** pour gérer la date de naissance des étudiants.
- ☞ Apportez les modifications nécessaires à ce changement dans le reste du code du programme principal.

Exercice 4 Tri de la promotion

La bibliothèque standard du C++ apporte un ensemble d'algorithmes classiques via la bibliothèque `<algorithm>`. Parmi ceux-ci, la fonction `std::sort()` permet de réaliser un tri dans un vecteur.

- 📖 En examinant l'[aide en ligne de cette fonction](#), faites en sorte que l'affichage de la promotion se fasse de façon triée selon les noms des étudiants.