PROJET : Cinémathèque

Les bases de données sont des moyens de stocker de nombreuses informations qui sont en relation les unes avec les autres. Pour créer une base de données, il est très souvent nécessaire d'identifier les différents objets qui seront modélisés et stockés, pour éviter de stocker des informations de manière redondante.

Nous allons simuler une petite **base de données cinématographique contenant des films.**

Découpage et organisation des données :

1) Pour chaque film, il faudra stocker :

- Son titre
- Son année de sortie
- Son réalisateur
- Ses acteurs principaux
- Sa durée en minutes
- Un ou plusieurs genres, parmi : Action, Horreur, Comédie, Documentaire, Policier, Drame, Animation, Science-Fiction (un film a au maximum 2 genres)

2) Pour chaque réalisateur et chaque acteur, il faudra stocker :

- Son nom
- Son prénom
- Sa date de naissance
- Sa nationalité
- Son statut : réalisateur ou bien acteur.

Une telle base de données peut être interrogée ou traitée par ce que l'on appelle des **requêtes**. Par exemple, on peut demander à afficher tous les éléments d'un film, à créer ou supprimer un film, à modifier certaines informations.

Une opération classique consiste, par exemple, à établir toute la filmographie d'un acteur ou d'un réalisateur.

Pour cela, est-il nécessaire de stocker, pour chaque réalisateur ou acteur, l'ensemble des films associés ?

De la même manière, un réalisateur ou un acteur a souvent plusieurs films à son actif.

Par exemple, David Fincher (né le 10 mai 1962 à Denver, aux USA) a réalise Fight Club, Seven et The Game, entre autres.

Est-il nécessaire, pour chacun des films cités, de stocker à chaque fois ces informations sur le réalisateur ? Comment pourrait-on éviter de dupliquer ces informations ?

Ce projet respecte les principes de la **programmation modulaire** *(fichiers d'entête .h et fichiers d'implémentation .c)* et offre un jeu de tests. Les tableaux sont soit alloués statiquement soit dynamiquement avec réallocation dynamique si nécessaire.

Les données sont lues sur l'**entrée standard** ou dans des f**ichiers de données** dont le format sera à préciser.

Pour les différents genres, on utilisera un type Genre défini à l'aide d'une énumération :

```
typedef enum genre
{
   AUCUN,
   ACTION,
   HORREUR,
   COMEDIE,
   DOCUMENTAIRE,
   POLICIER,
   DRAME,
   ANIMATION,
   SCIENCE_FICTION
} Genre;
```

1) Pour un acteur et un réalisateur, pourquoi un seul type **Personne** suffit?
   Définir ce type : le nom, le prénom et la nationalité sont représentés par des tableaux de 64 caractères, la date de naissance par un type **Date**, qu'il convient de définir.
   - Comment définir le statut d'une personne : réalisateur ou acteur ?
2) Définir le type **Film**.
   - Le titre est représenté par un tableau de 64 caractères.
   - Comment représenter le réalisateur ?
   - Comment représenter les acteurs, sachant que **4 au plus** sont mémorisés ?
   - Comment représenter les genres ?
3) Pour le type **Date**, définir les fonctions de saisie, d'affichage d'une date et de comparaison de deux dates *(fichiers date.h, date.c, testDate.c).*
4) Pour le type **Personne**, définir les fonctions de saisie, d'affichage, de recherche d'une personne par son nom *(fichiers personne.h, personne.c, testPersonne.c).*
   Indication : La recherche du nom d'une personne est effectuée dans un tableau de personnes.
   Quelle est sa déclaration prototype?

Affichage des informations relatives à un acteur ou à un réalisateur

exemple de résultat demandé :

```
Fiche REALISATEUR
Nom : David Fincher
Date de naissance : 10 mai 1962
Nationalité : américaine
```

5) Pour le type **Film**, définir les fonctions de saisie, d'affichage, de recherche de films par leur nom, par année, par durée inférieure à une durée donnée *(fichiers film.h, film.c, testFilm.c).*

Indication : *La recherche selon le critère demandé est effectuée dans un tableau de films.*

Quelle est sa déclaration prototype?

Affichage des informations relatives à un film

exemple de résultat demandé :

```
Fiche FILM
Fight Club (1999)
Réalisateur : David Fincher
Acteurs : Brad Pitt, Edward Norton, Helena Bonham Carter
Durée : 2h15
Genre : Action
```

6) Ecrire une fonction de recherche des titres des films d'un genre donné.

Indication : *La recherche selon le genre est effectuée dans un tableau de films.*

Quelle est sa déclaration prototype?

Affichage des titres de films du genre "Action"

exemple de résultat demandé :

```
Fight Club (1999)
Matrix (1999)
Doberman (1996)
```

7) En considérant qu'un réalisateur tourne par défaut **au plus 5 films** et qu'un acteur joue par défaut dans **au plus 10 films**, créer une fonction donnant la filmographie d'un acteur ou réalisateur.

Indication : *Les recherches des films d'un réalisateur et d'un acteur sont effectuées dans un tableau de films.*

Quelle est sa déclaration prototype?

Option : en cas de dépassement du nombre de films, prévoir un scénario permettant de s'affranchir de cette limite.

Filmographie d'un acteur ou réalisateur :

exemple de résultat demandé pour l'acteur Spacey:

```
Résultats : Spacey (Kevin)
Seven (1995)
Usual Suspects (1994)
American Beauty (1999)
Minuit dans le jardin du bien et du mal (1997)
Terre Neuve (2001)
```

8) Ecrire des fonctions répondant à une requête à critères multiples.

exemple : *"tous les films de David Fincher sortis entre 1995 et 2011 durant plus d'une heure".*

Quelle est la déclaration prototype de la fonction implémentant cette requête?

9) Dans une **fonction test1**, fonction de test du **fichier *test1.c***, définir 3 tableaux, comportant respectivement des films, des réalisateurs, des acteurs.
   Entrer toutes les données en dur dans le programme.
   Effectuer les appels aux différentes fonctions relatives aux films via un menu convivial.
10) Dans une **fonction test2**, fonction de test du **fichier *test2.c***, définir 3 tableaux, comportant respectivement des films, des réalisateurs, des acteurs.
   Faire saisir les données par l'utilisateur.
   Effectuer les appels aux différentes fonctions relatives aux films via un menu convivial.
11) Dans une **fonction test3**, fonction de test du **fichier *test3.c***, définir 3 tableaux, comportant respectivement des films, des réalisateurs, des acteurs.
   Lire les données dans un fichier dont la grille de lecture (format du fichier) a été définie au préalable.
   Effectuer les appels aux différentes fonctions relatives aux films via un menu convivial.
12) La fonction principale du projet doit appeler selon le choix de l'utilisateur l'une des 3 fonctions de test.

Rendu par binôme :

Une archive NOM1-NOM2.zip, contenant les fichiers .h et .c du projet.

Conseils :

1) Pour chaque fonction programmée, s'interroger sur les arguments qui peuvent poser des problèmes. Capturer ces cas à l'aide de pré-assertions (fonction assert).
2) Utiliser au maximum les fonctions déjà programmées dans les nouvelles à écrire. Il faut éviter au maximum la duplication de code.
3) Dès qu'une fonction est écrite, la tester de manière à recouvrir tous les cas significatifs possibles conserver les tests (le but étant de les relancer pour vérifier l'intégrité du programme lors de modifications futures).
4) Commenter le code en évitant absolument les commentaires inutiles. Il faut commenter chaque fonction écrite, juste avant son prototype. Il faut y mentionner les trois points suivants :
   a) une description du rôle de la fonction
   b) une description de ses paramètres
   c) une description de ce qu'elle renvoie
5) Préférer la concision au maximum. Un code concis et lisible possède une grande valeur.