**IFT1166 - Session Automne 1999** **Mohamed Lokbani**

**FINAL**

**Date:15 Décembre 1999** **Durée: 3 heures (de 18h30: 21h30) Local: B-2285**

**Directives:**

- Il vous est permis d'utiliser un livre de votre choix.
- Les documents de cours ne sont pas autorisés.
- Ordinateurs personnels prohibés.
- Calculatrice non programmable permise.

L'examen compte:

6 pages incluant celle-ci.
7 exercices obligatoires et un (1) exercice (exercice 8) optionnel (bonus).

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Exercice 1 (10 Points)

## Question -1.1-

Écrire la définition de la classe `s_date` correspondant à la structure suivante:

```
struct s_date {
private:
      int jour;  // le jour de naissance de la personne
      int mois; // le mois de naissance
      int an; // l'année de naissance
};
```

## Question -1.2-

Dans le cas où la fonction `main` fait appel à un constructeur de recopie de la classe `s_date`, la classe `s_date` doit-elle avoir un constructeur de recopie explicite (justifiez votre réponse)? Si la réponse est oui, écrire la définition du constructeur de recopie explicite pour la classe `s_date`.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Exercice 2 (10 Points)

## Question -2.1-

Écrire la définition de la classe `s_fiche` correspondant à la structure suivante:

```
struct s_fiche {
private:
      char *nom; // le nom de la personne
      int numero; // son numéro d'assurance sociale
      struct s_date naissance; // sa date de naissance
};
```

Pour la structure s_date, vous devez utiliser la classe s_date définie dans l'exercice 1.

## Question -2.2-

Dans le cas où la fonction `main` fait appel à un constructeur de recopie de la classe `s_fiche`, la classe `s_fiche` doit-elle avoir un constructeur de recopie explicite (justifiez votre réponse)? Si la réponse est oui, écrire la définition du constructeur de recopie explicite pour la classe `s_fiche`.

# Exercice 3 (15 Points)

## Question -3.1-

Le format en entrée étant libre mais en respectant quand même le type des données à saisir, écrire la définition de la fonction `SaisirDate` membre de la classe `s_date` déjà définie dans l'exercice 1. Cette fonction a comme rôle de saisir, à partir de l'entrée standard, puis de remplir les champs privés de la classe `s_date`.

## Question -3.2-

Le format en entrée étant libre mais en respectant quand même le type des données à saisir, écrire la définition de la fonction `SaisirFiche` comme membre de la classe `s_fiche` déjà définie dans l'exercice 2. Cette fonction a comme rôle de saisir, à partir de l'entrée standard, puis de remplir les champs privés de la classe `s_fiche`.

## Question -3.3-

Le format d'affichage en sortie étant libre, écrire la(les) définition(s) de la fonction `Afficher` nécessaire(s) pour afficher, sur la sortie standard, les informations suivantes issues des deux classes `s_date` (déjà définie dans l'exercice 1) et `s_fiche` (déjà définie dans l'exercice 2):

```
// Nom de la personne
// Numéro son assurance sociale
// Date de Naissance i.e. jour/mois/année
```

## Exercice 4 (10 Points)

Soit le programme suivant:

```
#include <fstream>
#include <string>

class Exo {
private:
  ifstream *entree;
  static int taille_fenetre = 130; //question -4.1-
  string chaine;
public:
  Exo(char *fichier_in);
  void process(){
    string xs("");
    if (chaine.size() > 4){
      xs.assign(chaine,4,taille_fenetre);
    }
    cout << xs << endl;
    exit(1);
  }
  ~Exo() {
    entree->close();
  }
};
Exo::Exo(char *fichier) {
  entree = new ifstream(fichier,ios::in);
  if ((*entree)) {
    while (getline((*entree),chaine)) {
       process();
    }
    cout << "Ok!\n";
  } else {
    cout << "probleme d'ouverture fichier\n";
    exit(1);
  }
}
int main() {
    Exo C("test.txt"); // question -4.2-
    return 0;
}
```

### Question -4.1-

La ligne de code suivante est-elle correcte (élaborez)? Dans le cas contraire, corrigez la.

```
static int taille_fenetre = 130;
```

### Question -4.2-

Supposez que le programme compile correctement, écrire la sortie du programme dans le cas où le fichier `test.txt` contient le texte suivant (l'espace blanc dans le texte est représenté par ce symbole: □):

```
□□□□NEW□YORK,□3□dec□(AFP)□□L'ancien□champion□du□monde□des□lourds□Muhamm
ad□Ali,□qui□s'etait□lui□meme□autoproclame□comme□"le□plus□grand",□a□ete□
designe□champion□du□siecle□par□le□magazine□specialise□americain□"Sports
□Illustrated",□au□cours□d'une□ceremonie□mercredi□soir□a□New□York.
```

## Exercice 5 (15 Points)

### Question -5.1-

Écrire la définition classe abstraite `peintre` qui contient deux fonctions virtuelles pures: `Saisir` et `Afficher.`

### Question -5.2-

Écrire la définition de la classe `Monet` qui dérive "publiquement" de la classe `peintre`, et a comme membres privés un tableau `TAB` contenant `NB_ELT` éléments du type `string`.

## Exercice 6 (15 Points)

### Question -6.1-

Écrire une fonction `main` permettant de faire ce qui suit:

Soit `Vdata` un conteneur séquentiel `vector` de la librairie standard (Standard Template Library). Écrire le code nécessaire permettant de déclarer puis de remplir ce conteneur avec les objets `objA` et `objB` (dans l'ordre), du type `OBJT`. Par la suite, afficher sur la sortie standard le contenu de ce vecteur en utilisant pour cela une fonction `Afficher` membre public de la classe `OBJT`.

### Question -6.2-

En vous inspirant de la question précédente, écrire le patron de fonctions (fonction template) `temp_print(.....)` qui lorsqu'elle est appliquée à un vecteur d'éléments du type T, elle se charge d'imprimer sur la sortie standard le contenu de ce vecteur. Par vecteur on entend la structure `vector` de la librairie standard (Standard Template Library). Vous pouvez supposer que la classe T possède une fonction membre `Afficher.`

Pour ce cas-ci, l'utilisation d'une telle fonction sur l'exemple de la question -1- sera comme suit:

```
int main() {

      // votre code ……


      temp_print(Vdata);

      return 0;

}
```

## Exercice 7 (25 Points)

Soit un tableau `TAB` contenant `NB_ELT` éléments: des mots, du type `string`. Chaque mot peut être écrit en majuscules et/ou en minuscules (sans accents). Écrire un programme permettant de réaliser les tâches suivantes:

Charger les données à la manière du TP1, à travers la fonction `load_data,` dont le prototype est comme suit:

```
void load_data (string tab[], unsigned int & taille);
```

La fonction `load_data` permet d'initialiser un tableau `tab` où chaque élément est du type `string`. L'initialisation du tableau, vous permettra de réaliser la tache de saisie des données initiales de manière simple. La variable `taille` est une référence à la taille du tableau. Elle indique le nombre d'éléments dans le tableau après remplissage.

Calculer la fréquence d'apparition de chaque mot dans le tableau `TAB`.

Afficher les mots et leur fréquence respective dans un fichier `test.txt,` dans un ordre décroissant par rapport à la fréquence d'apparition. Dans le cas ou deux mots ont la même fréquence, affichez ces mots dans un ordre croissant alphabétique (`devoir` vient avant `livre`).

Nous considérons que deux mots sont identiques s'ils ne différent que par le fait que certains de leur caractères sont en majuscule et minuscule (devoir ou Devoir ou DEVOIR sont considérés ici comme identiques, le même mot).

L'utilisation de la librairie standard est interdite (pas de STL), de même que pour le programme de tri (sort) de la librairie: vous devez en écrire un.

**Exemple**

Pour le tableau suivant,

| |
|---|
| LIVRE |
| cahier |
| devoir |
| livre |
| Devoir |

`taille` vaut `5`, et le fichier `resultat.txt`, obtenu suite à l'exécution de votre programme, contient ce qui suit:

```
devoir     2
livre      2
cahier     1
```

## Exercice 8 - optionnel - n’est pas obligatoire - (Bonus 5 Points)

La ligne suivante est dangereuse lors de son utilisation dans un programme. Expliquer les raisons, et réécrivez la d’une autre manière.

int & ref = new int;

## BONNES FÊTES!