TP 8 - Exceptions et Fichiers

Programmation Orientée Objet

Objectif

– Comprendre le principe des exceptions à travers un exemple de manipulation de fichiers.

1 Manipulation de fichiers

En c++, la manipulation de fichiers peut s'effectuer en utilisant les fichiers de la même manière qu'en C, ou bien en utilisant des classes C++ spécifiques. C'est cette dernière solution que nous allons adopter. Vous trouverez un annexe un exemple d'utilisation des primitives C++ pour les fichiers.

On vous demande de créer une classe permettant de rendre plus facile l'utilisation des bibliothèques C++ pour l'écriture des fichiers. Pour cela, vous devrez implémenter les opérations de la classe `DepartExo1.cpp` ci-dessous.

```
class GestionFichier{
        string sNom;
  public :
        GestionFichier(string nom="");
        ~GestionFichier();
        void LireFichier(string ** tab, int & iNbElt);
        void EnregistrerFichier (const T * tab, const int & iNbElt);
}
```

Cette classe permet la création d'un objet gestionnaire de fichier. Celui-ci a pour attribut le nom du fichier à manipuler. Il offre une opération permettant l'écriture d'un tableau de `string` dans le fichier, et une opération permettant de remplir un tableau de `string` à partir de la lecture d'un fichier. Ecrire un petit programme principal afin de la tester.

2 Gestion des exceptions

A partir de la classe précédente, mettez en place une gestion des exceptions, tant fonctionnelles que non fonctionnelles.

3 Généralisation

Nous cherchons toujours à améliorer notre gestionnaire de fichier. Pour cela, on souhaite pouvoir lui transmettre, au lieu d'un simple tableau de string, un tableau de classe quelconque. Modifiez votre classe en ce sens. Quelle restriction doit-on imposer à la classe transmise ?

Annexes

```
// Ouverture d'un fichier "donnees.txt" en lecture

#include <iostream>
#include <fstream>

using namespace std;

int main()
{
```

```
    ifstream fichier("donnees.txt");

    fichier.close();                        //fermeture du flux

    return 0;
}

// Lecture du contenu de "donnees.txt"
#include <iostream>
#include <fstream>

using namespace std;

int main()
{

    ifstream fichier("donnees.txt");
    char caractere;

    while(fichier.get(caractere))
        cout << caractere;

    cout << endl << endl;
    fichier.close();                        //fermeture du flux

    return 0;
}

// Ouverture du fichier "donnees.txt" en Ã©criture
#include <iostream>
#include <fstream>

using namespace std;

int main()
{

    ofstream fichier("donnees.txt");

    fichier.close();                        //fermeture du flux

    return 0;
}


//Ecrire dans "donnees.txt"
#include <iostream>
#include <fstream>

using namespace std;

int main()
{

    char bonjour[10] = "bonjour !"
    ofstream fichier("donnees.txt");

    fichier << bonjour;                 //Ã©criture de la chaine bonjour dans donnees.txt
    fichier.close();                        //fermeture du flux

    return 0;
}

//Differents types d'ouverture de flux
/*
Par defaut, ofstream cree automatiquement un fichier si celui prÃ©cise n'existe pas. Mais vous pouvez rajouter

    Voici la liste des parametres possibles :
        ios::app : Ouvre le fichier en ajout, a la fin, au lieu de supprimer son contenu au prealable.
        ios::ate : Permet de se placer en fin de fichier.
        ios::trunc : Comportement par defaut : le contenu est supprime Ã  l'ouverture.
        ios::nocreate : Provoque un echec d'ouverture si le fichier n'existe pas.
        ios::noreplace : Provoque un echec de l'ouverture si le fichier existe dÃ©jÃ .
*/
```