PARIS DESCARTES IUT

DUT Informatique - 1ère année
SDA - Structures de Données et Algorithmes

TRAVAUX PRATIQUES – Semaine n°6

Thèmes

- Entrées-sorties sur les flots fichiers
- Sérialisation d'items et de pile d'items

Les données ont une durée de vie limitée à l'exécution du programme qui les contient. Il est cependant possible de copier la valeur d'une variable sur un support moins volatile (e.g., un fichier). Les fonctions de sérialisation sont des fonctions dédiées à l'écriture et à la lecture de variables d'un type donné sur un tel support. Le but de ces exercices est i) d'écrire les fonctions de sérialisation d'un item de type donné, ii) d'ajouter au composant de pile d'items du TP n°4 des fonctions de sérialisation de pile utilisant celles de l'item.

Attention :
Pour la recopie de sources de Projet1 dans Projet2 (sans partage de source) (1) au niveau de Projet2, <clic droit> et sélectionnez "Ouvrir le dossier dans l'explorateur de fichier". Dans le répertoire ainsi ouvert : copier les fichiers de Projet1. (2) au niveau du Projet2, <clic droit> et sélectionnez « ajouter / Elément existant", sélectionnez les fichiers à copier, puis ajouter. Les fichiers apparaissent au niveau de Projet2 aux emplacements appropriés (header et sources).

Exercice 1. Insertion d'item dans un flot

Le but de cet exercice est d'écrire une fonction d'écriture d'un item de type `Position` dans un flot binaire de type `ofstream` préalablement ouvert.

1.1. Créez un nouveau Projet/Solution `sem05-tp` de nom `Exo1`. Importez à partir des sources de TP les fichiers de `Exo1` (`Position.h`, `Position.cpp` et `testEcritureBinairePosition.cpp`).

1.2. Ajoutez au fichier d'entête **`Position.h`** l'inclusion du fichier d'entête nécessaire aux entrées-sorties sur fichier et le prototype de la fonction d'écriture de la position `p` dans le flot `fE` :

```
void ecrireFB(std::ofstream& fE, const Position& p);
```

Ajoutez le code de cette fonction au fichier source **`Position.cpp`**.

1.3. Complétez le code du fichier **`testEcritureBinairePosition.cpp`** qui teste la fonction `ecrireFB`. Exécutez le programme qui crée le fichier binaire correspondant à vos entrées, vérifiez sa création.

Exercice 2. Extraction d' item à partir d'un flot

Le but de cet exercice est d'écrire une fonction de lecture d'un item de type `Position` dans un flot binaire de type `ifstream` préalablement ouvert.

2.1. Créez un nouveau projet `Exo2`. Recopiez les fichiers `Position.h` et `Position.cpp` du projet précédent, importez `testLectureBinaire Position.cpp` des sources de TP `Exo2`.

2.2. Ajoutez au contenu du fichier d'entête **`Position.h`** le prototype de la fonction de lecture de la position `p` à partir du flot `fL` :

```
void lireFB(std::ifstream& fL, Position& p);
```

Ajoutez le code de cette fonction au fichier source **`Position.cpp`**.

2.3. Complétez le code du fichier **`testLectureBinairePosition.cpp`** qui teste la fonction `lireFB`. Testez le programme en relisant le fichier binaire créé à l'exercice 1.

Exercice 3. Insertion de pile dans un flot

Le but de cet exercice est d'écrire une fonction de sauvegarde d'une pile d'items de type `Position` dans un flot binaire de type `ofstream` préalablement ouvert. Cette fonction doit utiliser pour sauvegarder un élément de la pile la fonction ecrireFB écrite à l'exercice 1.

3.1. Créez un nouveau projet Exo3. Recopiez les fichiers `Position.h` et `Position.cpp` du projet précédent, importez les fichiers `ConteneurTDE.h`, `ConteneurTDE.cpp`, `Item.h`, `Pile.h`, `Pile.cpp` et `testEcritureBinaire PilePositions.cpp` à partir des sources de TP `Exo3`.

3.2. Ajoutez au contenu du fichier d'entête **`Pile.h`** le prototype de la fonction d'écriture de la pile `p` dans le flot `fe` :

```
void ecrireFB(std::ofstream& fe, const Pile& p);
```

Ajoutez le code de cette fonction au fichier source **`Pile.cpp`**.

3.3. Complétez le code de **`testEcritureBinairePilePositions.cpp`** qui teste la fonction `ecrireFB`. Exécutez le programme de test.

Exercice 4. Extraction de pile à partir d'un flot

Le but de cet exercice est d'écrire une fonction de lecture d'une pile d'items de type `Position` dans un flot binaire de type `ifstream` préalablement ouvert. Cette fonction doit utiliser pour lire un élément de la pile la fonction `lireFB` écrite à l'exercice 2.

4.1. Créez un nouveau projet Exo4. Recopiez les fichiers `Position.h`, `Position.cpp`, `ConteneurTDE.h`, `ConteneurTDE.cpp`, `Item.h`, `Pile.h` et `Pile.cpp` du projet précédent, importez `testLectureBinairePile Positions.cpp` des sources de TP `Exo4`.

4.2. Ajoutez au contenu du fichier d'entête **`Pile.h`** le prototype de la fonction de lecture de la pile `p` dans le flot `fl` :

```
void lireFB(std::ifstream& fl, Pile& p);
```

Ajoutez le code de cette fonction au fichier source **`Pile.cpp`**.

4.3. Complétez le code de **`testLectureBinairePilePositions.cpp`** qui teste la fonction `LireFB`. Testez le programme en relisant le fichier binaire créé à l'exercice 3.