

M1103 : Structures de Données & Algorithmes fondamentaux Feuille TP n° 3

Utilisation des Files

OBJECTIFS PEDAGOGIQUES :

- 1.- Apprendre les manipulations de base d'un Type abstrait de Données File
- 2.- Découvrir le principe de gestion des exceptions

RESSOURCES A VOTRE DISPOSITION POUR REALISER CE TP :

Sur le WebCampus, dans la zone associée à ce module APL-M1103 :

- tp3.pdf : le présent sujet de tp
- dans une archive ressourcesTP3.zip :
 - file.h, file.cpp : les fichiers implémentant le type UneFile.
 - jeuDeCartes.h, jeuDeCartes.cpp : le paquetage de gestion de cartes
 - main.cpp : le programme principal du jeu, à compléter
 - jeu51Modele.exe : l'exécutable du programme complet, comme modèle de programme à reproduire

A CODER OBLIGATOIREMENT

- **Code 1** : Premier programme simple d'utilisation de files d'entiers.
- **Code 2** : Le jeu du 51.

Vous disposez de 2 séances de TP cette semaine pour réaliser ce travail. Si vous avez terminé avant, nous ne pouvons que vous recommander de réviser / compléter vos précédents TPs.

DIRECTIVES PARTICULIERES POUR CETTE SEANCE DE TP

1. Créer un répertoire tp3. Il contiendra tous les exercices qui vous sont demandés sur cette feuille de TP.
2. Dans le répertoire tp3 :
 - Vous devrez créer un projet par programme à écrire (mais attendez la suite du TP pour le faire)
 - Vous regrouperez ensuite tous les projets dans un même workspace nommé **tp3** (mais attendez la suite du TP pour le faire)
3. Dézipper le contenu de l'archive ressourcesTP3.zip fournie et placer tous les fichiers .h et .cpp de l'archive à la racine du répertoire tp3

À titre d'illustration, voici ce que contient mon répertoire tp3 et mon Workspace une fois le TP *terminé* :

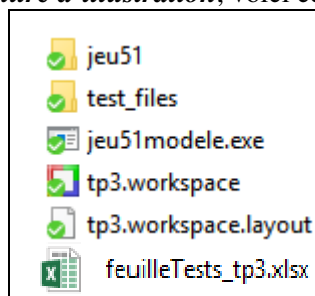


Figure 1 : Contenu de *mon* répertoire tp3 une fois le tp terminé

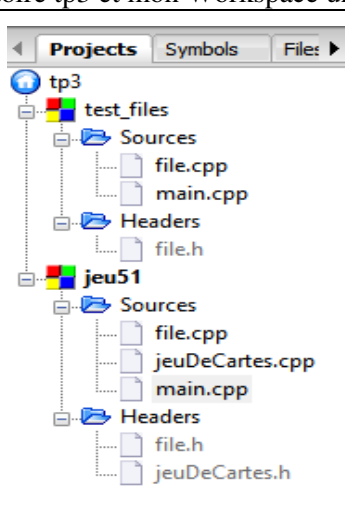


Figure 2: Physionomie de *mon* workspace

Code 1.- Pour se faire la main : premier programme de test simple d'utilisation d'une file d'entiers

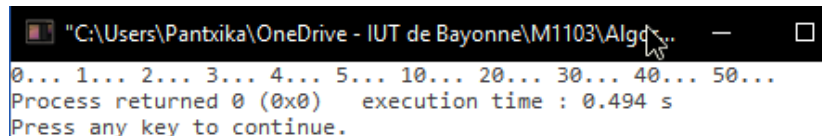
Il s'agit de coder l'exemple d'utilisation d'une file d'entiers présent sur votre polycopié de cours, transparent 31.

4. Dans le répertoire **tp3**, créer un projet **test_files**
5. Depuis votre navigateur de fichiers, dupliquer les fichiers **file.h** et **file.cpp** qui vous ont été fournis et placez-les dans le répertoire de projet **test_files**
6. Depuis CodeBlocks, ajouter à votre projet les fichiers **file.h** et **file.cpp** que vous venez de copier dans votre répertoire de projet
7. Renommer le workspace (tp3) et enregistrer le nouveau nom dans le répertoire tp3.
8. Assurez-vous que les liens entre les fichiers du projet (**main.cpp**, **file.h** et **file.cpp**) sont corrects avant le démarrage du codage. Pour ce faire, compléter le **main()** avec les directives **#include** adéquates.
9. Dans le fichier, main, écrivez le code fourni dans le polycopié de cours, transparent 31. Il s'agit d'un exemple d'utilisation d'une file d'entiers. Pensez à adapter la bibliothèque de gestion de files (fichier **file.h**) pour qu'elle gère bien des files d'entiers.
10. Compilez, exécutez, et vérifiez que vous obtenez les mêmes résultats que ceux prévus sur la fiche de tests. Complétez votre fiche de tests. Si nécessaire, corrigez le code puis consignez les nouveaux tests sur la fiche de tests.
11. Une fois ce premier test terminé, compléter votre programme de sorte à ajouter dans la file les éléments se trouvant dans un tableau d'entiers. Le programme affichera ensuite l'ensemble des éléments contenus dans la file.

Une fois complété, votre programme de test aura la forme suivante :

```
1  #include <iostream>
2  using namespace std;
3  #include "file.h"
4
5  int main()
6  {
7      //Déclaration et initialisation de la file
8      UneFile maFile;
9      initialiser(maFile);
10
11     // Ajout d'éléments dans la file
12     enfiler(maFile, 0); enfiler(maFile, 1);
13     enfiler(maFile, 2); enfiler(maFile, 3);
14     enfiler(maFile, 4); enfiler(maFile, 5);
15
16     // Déclaration et initialisation d'un tableau d'entiers
17     const unsigned short int TAILLE = 5;
18     int monTab[TAILLE] = {10, 20, 30, 40, 50};
19
20     // Ajout dans la file des éléments contenus dans le tableau
21     // à vous de faire
22
23
24
25     // Vider la file pour l'afficher
26     while (!estVide(maFile)) {
27         cout << premier(maFile) << "... " ; defiler(maFile);
28     }
29
30     return 0;
31 }
```

... et son exécution donnera lieu, pour mon jeu d'essai, à l'affichage suivant :



```
"C:\Users\Pantxika\OneDrive - IUT de Bayonne\M1103\Algo"
0... 1... 2... 3... 4... 5... 10... 20... 30... 40... 50...
Process returned 0 (0x0) execution time : 0.494 s
Press any key to continue.
```

Code 2.- Codage du jeu du 51

12. Dans le répertoire **tp3** et workspace **tp3**, créer un projet **jeu51**.
13. Depuis votre navigateur de fichiers, copiez les fichiers **file.h**, **file.cpp**, **jeuDeCartes.h** et **jeuDeCartes.cpp** qui vous ont été fournis et placez-les dans le répertoire de projet **jeu51**.
14. Depuis CodeBlocks, ajouter à votre projet les fichiers que vous venez de copier.
15. Remplacer la totalité du code du fichier **main.cpp** de votre projet par celui du fichier **main.cpp** qui vous a été fourni.
16. Assurez-vous que les liens entre les fichiers du projet sont corrects avant le démarrage du codage. Pensez aussi à adapter la bibliothèque de gestion de files (fichier **file.h**) pour qu'elle gère bien des files de cartes.
17. Assurez-vous que la compilation ne produit plus d'erreur. Dans la racine de votre répertoire **tp3**, supprimez alors tous les fichiers source fournis et ne gardez plus que le fichier **jeu51Modele.exe** : c'est le modèle de programme à produire. Vous alors prêt(e) pour compléter le programme du jeu du 51 selon les algorithmes faits en TD.
18. Prenez le temps d'analyser et de vous approprier le code du programme principal **main.cpp** qui vous a été fourni. Cela fait partie de l'exercice. Il s'agit d'un programme incomplet. Les portions à compléter sont identifiées par la mention */*à compléter*/*. Faites le lien entre les différentes parties de ce programme et les actions de l'algorithme fait en TD.
19. Compléter le code source du fichier **main.cpp** conformément aux directives de la feuille de TD n°3. Toutes les parties à compléter sont identifiées par la mention */*à compléter */*
À noter : Il ne manque que les instructions 'métiers', c'est-à-dire celles qui gèrent le jeu. N'ajoutez donc aucune instruction de 'mise en page' (affichage, saut de ligne, espaces...), tout est déjà prévu dans le programme **main** qui vous a été fourni.
20. Analysez le code de l'action **piocherUneCarte** dans le programme principal.
Cette action *sait prendre en charge* les exceptions (erreurs) éventuellement levées par le paquetage **jeuDeCartes** et évite donc que le programme ne se termine en état d'erreur. Explications :
 - En appelant la primitive **nouvelleCarte()**, le programme *tente d'obtenir* une carte du paquet de cartes (l'instruction **try** enveloppe l'instruction risquant de générer une exception).
 - Si le paquet de cartes est vide, la primitive **nouvelleCarte()** *lèvera l'exception* 'paquetEpuise' (c'est le rôle de l'instruction **throw** dans le corps de la primitive **nouvelleCarte()**).
 - Mais le programme principal *est capable d'intercepter et de gérer une éventuelle erreur* provenant du paquetage **jeuDeCartes** (c'est le rôle de l'instruction **catch**, elle *traite* l'exception levée par le sous-programme **nouvelleCarte()**).
Dans ce cas, le programme sait qu'il n'y a plus de cartes et arrêtera le jeu (proprement) selon la manière prévue dans les spécifications.
21. **Tests**
Complétez votre fiche de tests. Si nécessaire, corrigez le code puis refaites / re-consignez le nouveau test.
22. **Pourquoi ?**
Est-ce que dans ce TP on aurait pu laisser les fichiers **file.h** et **file.cpp** à la racine du répertoire **tp3** afin que les 2 programmes de test utilisent la même bibliothèque ? **Argumentez.**

Vous demandez à votre enseignant(e) ce que vous ne comprenez pas.