Travail 1 – SuperDir

1 Instructions générales

- La date de remise finale est le vendredi 24 février 2022 à 23 h 59
- Le travail s'effectue individuellement
- Les modalités de remise sont décrites à la fin de ce document
- La grille de correction se trouve à la fin de ce document

2 Introduction

Vous connaissez la commande `dir` qui permet de lister les fichiers contenus dans un répertoire. Ce travail consiste à réaliser une commande similaire, nommée `SuperDir`.

3 Énoncé

3.1 Étape 1 – Départ

Dans cette étape, vous devez :

- Créer un dépôt GIT vide
- Créer une solution Visual Studio
- Créer un projet SuperDir qui ne comporte qu'une fonction main vide, qui compile sans erreur et qui s'exécute correctement
- Créer un fichier .gitignore

3.2 Étape 2 – Création du tableau d'instances

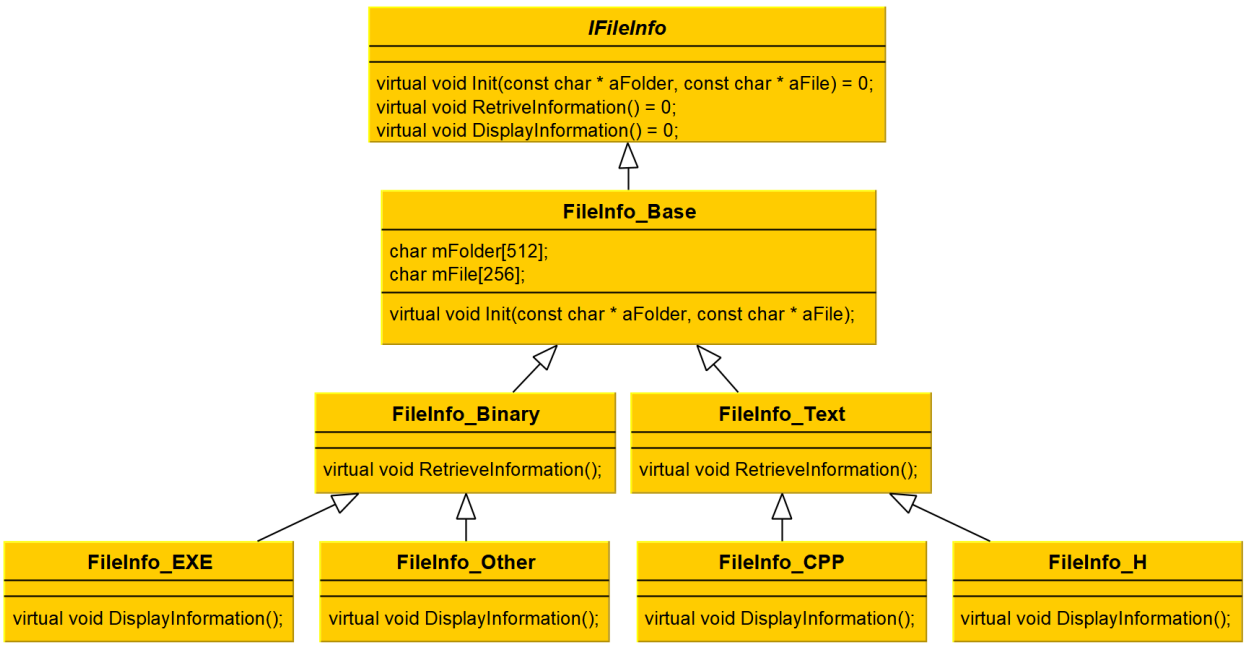
Dans cette section, vous allez créer une nouvelle fonction dans un nouveau fichier nommé `Functions.cpp`.

```
IFileInfo ** FindFiles(const char * aFolder);
```

Cette fonction commence par créer un tableau dynamique qui peut au maximum contenir 100 pointeurs vers l'interface `IFileInfo`. Elle retournera ce tableau. Si le répertoire contient moins de 100 fichiers, les pointeurs supplémentaires doivent être `NULL`.

Elle utilise les fonctions `FindFirstFile` et `FindNextFile` pour trouver l'ensemble des fichiers dans le répertoire indiqué.

En fonction de l'extension du fichier, une instance d'une des classes qui implémentent l'interface `IFileInfo` sera créée et le pointeur vers cette instance sera placé dans le tableau.



Notez que le diagramme de classe comporte toutes les classes à réaliser, mais que la description des classes ne contient que les méthodes publiques et les attributs évidents.

N'oubliez pas d'appeler votre nouvelle fonction à partir de votre fonction `main` en lui passant le nom de répertoire passé à la commande comme premier argument.

3.3 Étape 3 – Récupération des informations

Pour cette étape, vous devez ajouter le code nécessaire aux classes, dans les méthodes `RetrieveInformation`, pour retrouver les informations importantes. Pour les fichiers texte, il faut compter le nombre de lignes de texte. Pour les fichiers binaires, il faut obtenir la taille du fichier en octets.

Aussi, ajoutez une autre fonction dans le fichier des fonctions.

```
void RetrieveInformation(IFileInfo ** aFiles);
```

Celle-ci doit appeler la méthode `RetrieveInformation` pour chacune des instances du tableau. N'oubliez pas que le nombre maximum d'éléments dans le tableau est 100 et qu'il peut y avoir moins de 100 fichiers dans le répertoire.

Naturellement, votre fonction `main` doit aussi appeler cette nouvelle fonction.

3.4 Étape 4 – Affichage des informations

Pour cette section, vous devez créer une autre fonction.

```
void DisplayInformation(IFileInfo ** aFiles);
```

Sans surprise, cette nouvelle fonction doit appeler la méthode `DisplayInformation` de chacune des instances dans le tableau.

Vous devez aussi compléter les méthodes `DisplayInformation` des différentes classes. Chacune d'entre elles doit afficher une seule ligne de texte. Voici un exemple d'affichage.

```
Test.h            23 lignes  Langage C/C++ - Fichier entete
Test.cpp         178 lignes  Langage C++ - Fichier source
Test.exe           128 Mio  Executable
Test.bin             980 o
```

Il est important de noter que la taille des fichiers binaires est indiquée sans fraction en utilisant l'unité le plus appropriée parmi :

Abréviation	Unité	Pour les fichiers
o	Octet	De 0 à 1024 octets
Kio	Kilooctet informatique	De 1 à 1024 kilooctets
Mio	Mégaoctet informatique	De 1 à 1024 Mégaoctets
Gio	Gigaoctet informatique	De 1 Gigaoctet et plus

3.5 Étape 5 – Libération de la mémoire

Pour cette étape, créer une nouvelle fonction.

```
void ReleaseMemory(IFileInfo ** aFiles);
```

Cette fonction détruit chacune des instances et libère la mémoire occupée par le tableau.

3.6 Étape 6 (optionnel) – Les fichiers .BMP

Ajouter la classe `FileInfo_BMP` et faites que la taille de l'image, en pixel, soit affichée comme dans l'exemple qui suit.

```
Test.bmp            250 Kio  Image – 640 x 480 pixels
```

4 Modalité de remise

Après avoir terminé chacune des étapes, vous devez la montrer à votre enseignant et la commettre dans votre dépôt GIT.

Le travail doit être remis via LÉA avant le 24 février 2021 à 23 h 59.

Attention de ne pas remettre les sous-répertoires suivants de votre solution Visual Studio.

- .vs
- x65
- Debug
- Release

En évitant d'inclure les dossiers temporaires mentionnés, créer une archive en format .zip de votre solution et remettez la par LEA.

5 Références

5.1 Page WEB

# 1	FindFirstFileA function (fileapi.h) https://docs.microsoft.com/en-us/windows/win32/api/fileapi/nf-fileapi-findfirstfilea
#2	FindNextFileA function (fileapi.h) https://docs.microsoft.com/en-us/windows/win32/api/fileapi/nf-fileapi-findnextfilea

6 Grille de correction

Item	Description	Pondération
1	Étape 1 • Le nom de l'utilisateur GitHub est-il significatif ? • Le nom du dépôt GIT est-il significatif ? • Le fichier .gitignore est-il complet ? • Le fichier .gitignore est-il exempt de ligne inutile ? • Le fichier .gitignore est-il exempt de ligne problématique ? • Le programme compile-t-il ? Le programme s'exécute-t-il ?	10
2	Étape 2 • Le programme compile-t-il ? Le programme s'exécute-t-il ? • La fonction alloue-t-elle le tableau correctement ? • La fonction trouve-t-elle l'ensemble des fichiers ? • La fonction crée-t-elle les instances comme demandé ? • Le code est-il facile à comprendre et à maintenir ?	15
3	Étape 3 • Le programme compile-t-il ? Le programme s'exécute-t-il ? • Les classes retrouvent-elles l'information correctement ? • Le code est-il facile à comprendre et à maintenir ?	20
4	Étape 4 • Le programme compile-t-il ? Le programme s'exécute-t-il ? • Les classes affichent-elles l'information correctement ? • Le code est-il facile à comprendre et à maintenir ?	20
5	Étape 5 • Le programme compile-t-il ? Le programme s'exécute-t-il ? • La fonction détruit-elle les instances ? • La fonction libère-t-elle la mémoire utilisée pour le tableau ? • Le code est-il facile à comprendre et à maintenir ?	15
6	Étape 6 • Le programme compile-t-il ? Le programme s'exécute-t-il ? • La nouvelle classe retrouve-t-elle l'information correctement ? • La nouvelle classe affiche-t-elle l'information correctement ? • Le code est-il facile à comprendre et à maintenir ?	0
9	Remise finale • Les corrections discutées ont-elles été apportées ? • L'archive est-elle remise par LEA ? • L'archive est-elle en format .zip ? • L'archive est-elle exempte de répertoire temporaire ?	20
10	Qualité de la langue	(10 %)
	Total	100 %