POLYTECH DIJON

TRAVAUX DIRIGÉS

Programmation Orientée Objets / C++

Fichiers binaires

OBJECTIFS

A travers cet exercice, vous toutes les notions vues lors des précédentes séances de TD :

- Manipuler des fichiers textes et binaires,
- Structurer votre programme avec des classes.

CONTEXTE

Par un hasard plus ou moins heureux, vous avez récupéré le sujet de la prochaine épreuve de C++. Malheureusement, le fichier a été encodé par son auteur... Ou devrait-on dire, les fichiers : **`epreuve.index`** et **`epreuve.parts`** (téléchargez l'archive epreuve.zip).

Lorsque l'on ouvre le fichier **`epreuve.index`** avec un éditeur de texte, on peut y lire de nombreuses lignes constituées de deux nombres séparés par le caractère ':'.

```
0:0
19:12
153:22
159:31
129:40
60:51
…
```

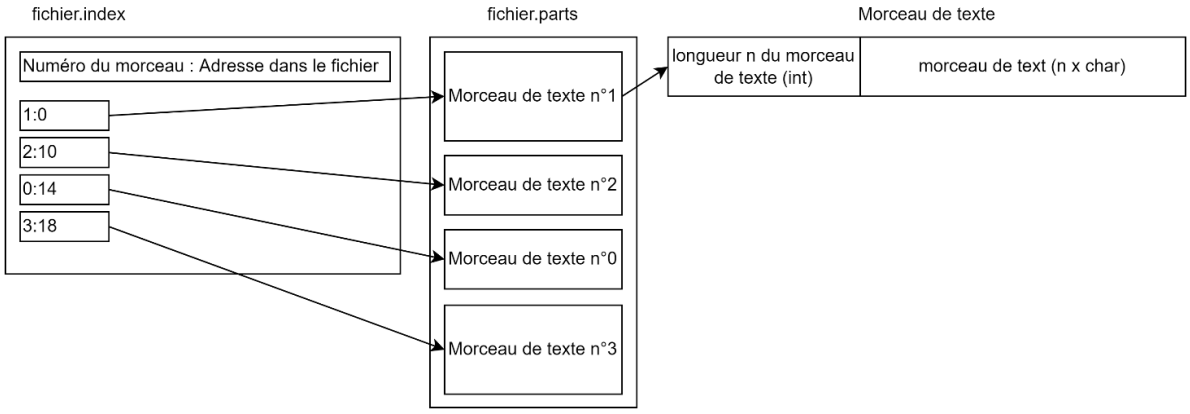
Pour ce qui est du fichier **`epreuve.parts`**, sa lecture dans un fichier texte est incompréhensible. Des données binaires semblent avoir été associé à des morceaux de texte. C'est un peu comme si tout le document original avait été déchiqueté en petits morceaux.

Après quelques minutes de réflexion, vous pensez avoir compris comment fonctionne l'encodage. Le document initial a été découpé en plusieurs morceaux de taille variable. Puis tous les morceaux ont été enregistrés dans le fichier **`epreuve.parts`** dans un ordre aléatoire. Le fichier **`epreuve.index`** sert simplement à retrouver la position de chaque morceau dans le fichier pour tout remettre dans l'ordre.

Ainsi, lorsqu'on lit **19:12** dans le fichier **epreuve.index**, il faut comprendre que le morceau de texte n°19 se trouve à l'octet n°12 du fichier **epreuve.parts**.

Lorsque vous irez lire le fichier **epreuve.parts** à partir de l'octet n°12, vous trouverez les 4 octets d'un entier représentant la taille de la chaine de caractères qui le suit. A ce moment-là, il sera possible d'extraire ce morceau de texte.

Voici un schéma qui explique tout cela :



LECTURE DU FICHIER .INDEX

Première étape, être capable de lire le contenu du fichier **epreuve.index**. Il s'agit d'un fichier texte qui ressemble beaucoup à un fichier CSV si ce n'est que le séparateur utilisé ici est « : » au lieu de « ; ».

- Créez un nouveau projet C++ vide avec Visual Studio Community.
- Ecrivez un programme qui lit le contenu de **epreuve.index** et qui affiche, pour chaque ligne du fichier, le numéro du morceau de texte et l'adresse à atteindre dans le fichier **epreuve.parts**.

```
Morceau 0           = Adresse 0
Morceau 19          = Adresse 12
Morceau 153         = Adresse 22
…
```

> **Astuce**
> Placez les fichiers epreuve.index et epreuve.parts dans le même dossier que les sources de votre programme. Ce sera plus simple pour y accéder.

LECTURE DU FICHIER .PARTS

Le fichier **epreuve.parts** n'est pas un fichier texte, mais un fichier binaire. Les octets du fichier ne représentent donc plus systématiquement des caractères.

- Créez une fonction **extractPieceOfText** qui prendra en paramètres :
  - Le fichier duquel extraire le morceau de texte
  - L'adresse à laquelle se trouve le morceau de texte recherché dans le fichier

```
// Pour se déplacer dans un fichier binaire
file.seekg(position);
```

```
// Lire un entier dans un fichier binaire
int entier;
file.read((char*)&entier, sizeof(entier));

// Lire 7 caractères dans un fichier binaire
// Crée une string initialisée avec 7 caractères vides
std::string str(7, '');
// Remplit la string avec 7 caractères lus dans le fichier
file.read((char*)str.c_str(), str.size());
```

- Dans la fonction mail de votre programme, appelez la fonction **extractPieceOfText** pour chaque ligne du fichier **epreuve.index** et affichez chaque morceau de texte extrait pour confirmer le bon fonctionnement de **extractPieceOfText**.
- Modifiez votre application pour que les morceaux de texte apparaissent dans le bon ordre.

GENERATION DU DOCUMENT INITIAL

- Modifiez votre programme pour créer un fichier TXT qui contiendra le texte original reconstruit.
- Testez le bon fonctionnement.

CHALLENGE

M'est avis que le code que vous avez produit manque quelque peu de structure :-)

- Réorganisez votre code pour obtenir une structure propre et maintenable :
  - La fonction main ne doit contenir que l'essentiel permettant le démarrage du programme.
  - Le rôle de chaque fonction et classe doit être bien délimité (par exemple, si une classe s'occupe de l'extraction des données d'un fichier, elle ne gère pas l'affichage des informations dans la console).