FIX TO CSV

Mission n°5 – Portfolio – Antoine Gandelin

FIX TO CSV

Fichiers à champs de longueur fixe : objectif

L'objectif était dans un premier temps de récupérer les valeurs contenues dans chaque ligne, ces valeurs sont repérées grâce à leur place dans la ligne.

- Nom et Prénom - Longueur : 80 - Position : 1-80 (NOM*PRENOMS)
- Sexe - Longueur : 1 - Position : 81 - (1 = Masculin; 2 = féminin)
- Date de naissance - Longueur : 8 - Position : 82-89 - (AAAAMMJJ)
- Code du lieu de naissance - Longueur : 5 - Position : 90-94 (Code Officiel
- Géographique 2)
- Commune de naissance en clair - Longueur : 30 - Position : 95-124
- Pays de naissance en clair - Longueur : 30 - Position : 125-154
- Date de décès - Longueur : 8 - Position : 155-162 - Type : Numérique (AAAA/MM/JJ)
- Code du lieu de décès - Longueur : 5 - Position : 163-167 (Code Officiel Géographique)
- Numéro d'acte de décès - Longueur : 9 - Position : 168-176

Il fallait traiter ce fichier grâce à un programme écrit en c++, l'objectif étant de passer ce fichier à champ de longueur fixe vers un fichier avec un séparateur de champ (CSV), nous utiliserons le ; comme séparateur. L'objectif était en même temps de ne conserver qu'une partie des champs proposés pour arriver à la structure de sortie suivante.

```
// Programme permettant de tra
// un fichier à champ fixe
// en csv (seuls certains cham

#include <iostream>
#include <fstream>
#include <string>

struct sortie
    {
    std::string nom;
    std::string prenom;
    char sexe;
    std::string date_nais;
    std::string code_com_nais;
    std::string date_dc;
    std::string code_com_dc;
    };
```

Ce programme se décompose en plusieurs étapes :

1. Création d'une structure correspondant au fichier d'entrée.
2. Création d'une structure correspondant au fichier de sortie.
3. Récupération du nom du fichier à traiter passé en paramètre
4. Ouverture des deux fichiers (entrée et sortie), le fichier de sortie s'appelle comme le fichier d'entrée mais avec une extension .csv
5. Lecture du fichier d'entrée grâce à la méthode read() dans la structure d'entrée.
6. Tant que l'on est pas arrivé à la fin du fichier
7. Passage de la structure d'entrée vers la structure de sortie
8. Écriture de la structure de sortie vers le fichier csv à obtenir.
9. Lecture du fichier d'entrée grâce à l'instruction fread() dans la structure d'entrée.
10. Fermeture des fichiers, indication de la fin du traitement, statistiques sur le traitement.

Les fichier suivants étaient à disposition (de 2020 et 2021) :

- deces-2020.txt
- deces-2021-m01.txt
- deces-2021-m02.txt
- deces-2021-m03.txt
- deces-2021-m04.txt
- deces-2021-m05.txt
- deces-2021-m06.txt
- deces-2021-m07.txt
- deces-2021-m08.txt
- deces-2021-m09.txt

L'ensemble de ces fichiers se trouvent à l'adresse suivante :
http://www.sio-reims.fr/periodes/suivi_dc_4.zip

I. Étape 1 – Correction

1. Création d'une structure correspondant au fichier d'entrée

La taille d'une ligne correspond normalement à la somme des tailles des champs, toutefois, il y a ici sur chaque ligne un emplacement prévu pour des utilisations futures, cet espace est nommé « filler » dans la structure suivante, le « filler » intègre aussi les caractères de fin de ligne « CR » et « FF ».

```
    std::string code_com_dc
    };

struct entree
    {
    char nom[81];
    char sexe;
    char date_nais[9];
    char code_lieu_nais[6];
    char com_nais[31];
    char pays_nais[31];
    char date_dc[9];
    char code_lieu_dc[6];
    char acte[10];
    //char filler[24];
    };
```

2. Récupération du nom du fichier à traiter passé en paramètre

Le nom du fichier est obtenu grâce au paramètre passé dans la fonction « main() ». C'est le deuxième élément de la ligne de commande, donc la valeur de b[1].

Le fichier est ouvert en entrée mais aussi en mode binaire, d'où l'opérateur « | » (ou binaire).

```
    fentree.open(b[1], std::ios::in|std::ios::binary);
```

3. Lecture du fichier d'entrée grâce à la méthode read() dans la structure d'entrée

Voir le fichier fixed2csv.cc.

Afin de rendre plus lisible le code, il a fallu déplacer les opérations de lecture dans une fonction.

Pour cela, il a fallu passer la structure comme paramètre en lecture/écriture (en entrée et sortie). Le passage de l'adresse de la structure a été réalisé grâce à l'opérateur « & ».

La fonction renvoie un booléen qui permet ainsi de définir une condition de sortie de la boucle de lecture.

```
bool lire_ligne(entree &ent, std::fstream &f)
```

4. Traitement des éléments de la structure d'entrée

Voir le fichier fixed2csv_2.cc

Il faut maintenant remplir la structure de sortie à partir de la structure d'entrée. Pour cela, chaque élément est traité pour obtenir le résultat voulu.

L'utilisation des méthodes de la classe « std::string » permet de simplifier l'approche.

Méthodes utilisées :

- « find() »
- « substr() »

A chaque fois, une vérification du prototype et du fonctionnement de la méthode est nécessaire afin de l'utiliser en pleine connaissance de cause.

Pour la date, il a fallu utiliser une fonction afin de pouvoir utiliser la fonction deux fois.

```
std::string transfo_date(std::string s1)
```

Voir fichier fixed2csv_3.cc

Dans un premier temps, il faut terminer de passer les données de la structure « e » de type « entree » vers la structure « s » de type « sortie ».

5. Le code de la commune de naissance ne pose aucune difficulté

```
// 4 - s.code_com_nais
s.code_com_nais = e.code_lieu_nais;
```

6. Le nom du pays pose deux problèmes :

- Quand il s'agit de FRANCE, le champ est vide, il faut le réintégrer.
- Il faut supprimer les espaces qui ne servent à rien dans le mot (en fin de champ).

Pour cela, l'utilisation de deux méthodes de « std::string » sont particulièrement utiles ici :

- « erase() »
- « find_last_not_of() »

```
// 5 - s.pays_nais
s.pays_nais = e.pays_nais;
```

Je recherche la position du dernier caractère qui n'est pas un espace. En ajoutant 1 je trouve la position du premier espace de la série d'espaces restants.

```
size_t poses = s.pays_nais.find_last_not_of(" ,") + 1;
```

Je peux maintenant effacer tous les espaces à partir de cette position.

```
s.pays_nais.erase(poses);
```

Je vérifie que la chaîne est vide, si c'est le cas, j'indique FRANCE

```
s.pays_nais=s.pays_nais.empty()?"FRANCE":s.pays_nais;
```

La date de décès fonctionne comme la date de naissance

```
// 6 - s.date_dc
s.date_dc = transfo_date(e.date_dc);
```

7. Le code de la commune de décès ne pose aucune difficulté

```
// 7 - s.code_com_dc
s.code_com_dc = e.code_lieu_dc;
```

Pour écrire, il y a deux étapes

- L'ouverture du fichier en écriture dont le changement d'extension
- L'écriture des données

J'utilise la méthode de « std::string replace() »

```
std::string nom_fic;
nom_fic = b[1];
nom_fic.replace(nom_fic.end()-3, nom_fic.end(), "csv");
fsortie.open(nom_fic.c_str(), std::ios::out);
```

Pour écrire les données, j'utilise simplement l'opérateur de sortie.

Exemple :

```
// Ecriture du fichier
fsortie<<s.nom<<";";
```

puis je termine par un saut de ligne « \n »

II. Automatisation des processus

Il me reste à traiter un ensemble de fichiers.

En les déposant dans un répertoire fic, on peut lancer une commande bash très simple

```
for fic in $*
do
    ./fix2csv $fic
done
```

Voir fichier process.sh