Fix2csv

Programme C++

TOUROLLE Antoine

Fix2csv - Programme C++

TOUROLLE Antoine

I. Objectif :

L'objectif était dans un premier temps de récupérer les valeurs contenues dans chaque ligne. Ces valeurs sont repérées grâce à leur emplacement dans les lignes du fichier.

- Le nom et le prénom - Longueur : 80 - Position : 1-80, séparés par le caractère * (NOM*PRENOMS)
- Le sexe - Longueur : 1 - Position : 81 - (1 = Masculin; 2 = féminin) • Date de naissance - Longueur : 8 - Position : 82-89 - (AAAAMMJJ)
- Le code du lieu de naissance - Longueur : 5 - Position : 90-94 (Code Officiel • Géographique 2)
- La commune de naissance (en clair) - Longueur : 30 - Position : 95-124
- Le pays de naissance (en clair) - Longueur : 30 - Position : 125-154 (peut ne pas être renseigné)
- La date de décès - Longueur : 8 - Position : 155-162 - Type : Numérique (AAAA/MM/JJ)
- Le code du lieu de décès - Longueur : 5 - Position : 163-167 (Code Officiel Géographique)
- Le numéro d'acte de décès - Longueur : 9 - Position : 168-176

Pour traiter ce fichier, nous avons écrit un programme en C++.

L'objectif étant de passer ce fichier à champs de longueur fixe vers un fichier avec un séparateur de champ (CSV). Nous utiliserons le point-virgule ";" comme séparateur pour le fichier CSV. La structure de sortie, dans laquelle on ne conserve qu'une partie des champs proposés, est la suivante :

Structure de sortie attendue

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

II. Construction du programme

Création de la structure d'entrée

La taille d'une ligne doit normalement correspondre à la somme des tailles des champs, cependant, nous avons, pour chacune des lignes, un emplacement prévu en cas d'utilisations futures : ***filler***. Il intègre aussi les caractères de ligne (CR & FF)

Structure d'entrée se basant sur le fichier à champs fixe

```
// Structure d'entrée
struct entree
   {
   // Déclaration des variables de type char*
   char nom[81];
   char sexe;
   char date_nais[9];
   char code_lieu_nais[6];
   char commune_nais[31];
   char pays_nais[31];
   char date_dc[9];
   char code_lieu_dc[6];
   char acte[10];
   char filler[24];
   };
```

Récupération du fichier à champs fixe passé en paramètre

Extrait du terminal

```
debian:~$ cd public_html/prog_csv/
debian:~/public_html/prog_csv$ c++ -o fix2csv fix2csv.cc
debian:~/public_html/prog_csv$ ./fix2csv
debian:~/public_html/prog_csv$ ./fix2csv deces-2020.txt
```

Le nom du fichier à champs fixe est récupéré grâce au paramètre passé dans la fonction main() : c'est le deuxième élément de la ligne de commande (c'est-à-dire la valeur de b[1])

Lecture du fichier d'entrée

Pour rendre le code plus lisible, nous avons créer une méthode regroupant les opérations de lecture : lire_ligne().

- La structure est passée comme paramètre en lecture/écriture (en entrée et sortie).
- L'opérateur '&' permet de passer l'adresse de la structure.
- La fonction renvoie un booléen afin de permettre une condition de sortie de la boucle de lecture.

Méthode lire_ligne()

```
// Méthode permettant la lecture des données du fichier d'entrée
bool lire_ligne(entree &ent, std::fstream &f)
   {
   // Déclaration d'une variable b de type boolean, initialisée à true par défaut
   bool b=true;
   f.read((char *)&ent.nom,80);
   if(!f.good())
      {
      b=false;
      }
   f.read(&ent.sexe,1);
   f.read((char *)&ent.date_nais,8);
   f.read((char *)&ent.code_lieu_nais,5);
   f.read((char *)&ent.commune_nais,30);
   f.read((char *)&ent.pays_nais,30);
   f.read((char *)&ent.date_dc,8);
   f.read((char *)&ent.code_lieu_dc,5);
   f.read((char *)&ent.acte,9);
   f.getline(ent.filler, 26);
   return b;
   }
```

Traitement des données

Il faut maintenant remplir la structure de sortie avec les données de la structure d'entrée.

Pour obtenir le résultat espéré, nous traitons chaque élément en utilisant les méthodes de la classe std::string → *find()* et *substr()*.

Concernant les dates, nous utilisons la fonction *transfo_date()* qui permet de modifier le format des dates passées en paramètre

Méthode transfo_date()

```
// Méthode de transformation du format de la date :
// Date par défaut ->
std::string transfo_date(std::string s1)
   {
   std::string s2,s3,s4;
   s2 = s1.substr(0,4);
   s2 += "-";
   s3 = s1.substr(4,2);
   s3 = s3=="00"?"01":s3;
   s2 += s3;
   s2 += "-";
   s4 = s1.substr(6,2);
   s4 =  s4=="00"?"01":s4;
   s2 += s4;
   return s2;
   }
```

Séparation du nom et du prénom

```
// 1 - Valorisaton de s.nom et de s.prenom
// à partir de e.nom
s.nom = e.nom;
s.prenom = e.nom;
// On cherche la position correspondante aux caractère de séparation * et /
size_t poset = s.nom.find('*');
size_t possl = s.nom.find('/');
// Découpage de la chaine de caractères en 2 sous-chaînes
// Première sous-chaîne : indice 0 jusqu'au caractère '*'(en l'excluant)
s.nom = s.nom.substr(0, poset);
// On ajoute 1 à la valeur de poset afin de passer le caractère de séparation
poset++;
// Seconde sous-chaîne : indice[poset+1] jusqu'au caractère '/' (en l'excluant)
s.prenom = s.prenom.substr(poset, possl-poset);
```

Changement de format de la date

```
// 3 - s.date_nais sous format de date changé
s.date_nais = transfo_date(e.date_nais);


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

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

- Lorsque le pays de naissance est « FRANCE », le champ est vide dans le fichier d'entrée, nous allons le rajouter.

```
// Opérateur ternaire:
// Si le pays de naissance n'est pas renseigné, sa valeur sera, par défaut, "FRANCE"
s.pays_nais=s.pays_nais.empty()?"FRANCE":s.pays_nais;
```

- On supprime les espaces inutiles en fin de champ avec les méthodes de std::string → *erase()* et *find_last_not_of()*.

On recherche la position du dernier caractère qui n'est pas un espace. En rajoutant 1 à cette position, on trouve la position du premier espace de la série d'espaces restants puis on efface tous les espaces à partir de cette position :

```
// On cherche la position du dernier caractère de séparation correspondant à ','
size_t poses = s.pays_nais.find_last_not_of(" ,") + 1;
s.pays_nais.erase(poses);
```

Écriture du fichier de sortie

L'écriture du ficher de sortie se passe en 2 étapes :

- L'ouverture du fichier en écriture avec changement d'extension vers CSV

- L'écriture des données

Ouverture du fichier en écriture

Pour modifier l'extension du fichier, on utilise la méthode *std::string* ***replace()*** :

Utilisation de la méthode replace()

```
std::string nom_fic;
// La variable nom_fic est initialisée par le nom du fichier
// passé en paramètre dans le terminal : ./fix2csv [nom_du_fichier].txt
nom_fic = b[1];
// Création du nouveau fichier (de type CSV)
nom_fic.replace(nom_fic.end()-3, nom_fic.end(), "csv");
// Ouverture du fichier en sortie
fsortie.open(nom_fic.c_str(), std::ios::out);
```

Écriture des données

Ecriture sur le fichier de sortie

```
// Ecriture du fichier de sortie
fsortie<<s.nom<<";";
fsortie<<s.prenom<<";";
fsortie<<s.sexe<<";";
fsortie<<s.date_nais<<";";
fsortie<<s.code_com_nais<<";";
fsortie<<s.pays_nais<<";";
fsortie<<s.date_dc<<";";
fsortie<<s.code_com_dc<<"\n";
}
```

Vérification du fichier de sortie

5 premières lignes

	A	B	C	D	E	F	G	H	I
1	LATIRE	MARCELLE BERNADETTE LOUISE	2	1933-05-18	50412	FRANCE	2021-01-11	1004	
2	GIROUD	EMILE ALIX	1	1940-07-05	1054	FRANCE	2021-01-14	1004	
3	BIARD	VALERIE ANNE	2	1963-07-10	1053	FRANCE	2021-01-14	1004	
4	MARTINES	ANNA	2	1924-08-16	69266	FRANCE	2021-01-15	1004	
5	PERRIN	LUCIENNE	2	1930-07-29	38084	FRANCE	2021-01-08	1004	

5 dernières lignes

66102	CORNAILLE	DANIEL PHILIPPE	1	1953-06-11	99131	BELGIQUE	2020-07-20	99131	
66103	FASSON	CHRISTIAN LOUIS	1	1927-09-27	8183	FRANCE	2020-08-23	99131	
66104	HANEUSE	LOUIS DOMINIQUE	1	1949-03-18	75051	FRANCE	2020-07-22	99131	
66105	SMADJA	ABRAHAM ALBERT	1	1926-11-20	99351	TUNISIE	2019-04-16	99131	
66106	BONDON	DOMINIQUE ROGER	1	1951-11-29	75114	FRANCE	2020-10-31	99131	
66107									

Automatisation des processus

Il est possible de traiter un ensemble de fichiers avec ce programme.

Après les avoir déposé dans un répertoire « fic », nous pouvons lancer une commande bash :

Fichier process.sh exécutable depuis le terminal

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