Projet : données libres

1. Prélude

Ce projet est volontairement long afin que chacun puisse s'exprimer en fonction de ses compétences, de son éventuelle expérience préalable et de ses goûts. La progression dans le projet est pensée en terme d'exercices à difficulté croissante.

Les exercices 1 à 5 sont accessibles à tous et vous permettent d'obtenir 12. Les exercices suivants demandent plus d'investissement personnel, à vous de voir jusqu'où vous voulez aller : **visez le maximum !** D'autre part, l'expérience que vous accumulerez au fur des premières questions vous fera paraître les questions ultérieures plus simples. Et vous aurez un vrai sentiment d'accomplissement en progressant dans le sujet.

Accrochez-vous, et demandez de l'aide aux autres étudiants (sans copier leur code bien évidemment, cela n'aurait aucun intérêt) ainsi qu'à vos chargés de TD.

Table des matières

2. Introduction

De plus en plus d'organisations, gouvernementales ou autres, mettent leurs données à la disposition du public, afin que celui-ci puisse se les approprier, en trouver des utilisations nouvelles et créatives, et aussi contrôler le fonctionnement de ces organisations sur la base de faits concrets.

Par exemple tout un chacun peut maintenant, grâce aux données mises en ligne récemment par la RATP, implanter une application adaptée à ses besoins spécifiques ; par exemple qui déclenche le réveil matin 25 minutes avant l'arrivée du RER, que celui-ci soit à l'heure ou pas ; ou d'un RER précédent en cas de gros retard annoncé sur la ligne. D'autres données incluent le cadastre ou le processus de rédaction des lois à l'Assemblée nationale, etc. Le citoyen peut ainsi consulter les amendements aux projets de lois auxquels son député a participé, et voir si ceux-ci sont conformes aux engagements de campagne de ce dernier. Voir par exemple les actions de l'association Regards Citoyens ; et en particulier cet article.

L'analyse de ce type de données a par exemple permis en janvier 2016 à des journalistes de mettre au jour le fait que des matchs de tennis de l'ATP avaient été truqués. Les données ainsi que leurs analyses sont disponibles sur `http://github.com/BuzzFeedNews/everything`. On y trouvera d'autres analyses de données tels que les tremblements de terre reliés à l'exploitation des gaz de schiste aux États-Unis, les mouvements des donateurs de la campagne présidentielle américaine lorsqu'un candidat sort de la course, ou une analyse du placement des enfants dans les crèches.

Comme les données peuvent être de grande taille, en extraire des informations intéressantes requiert la plupart du temps des traitements automatisés. Ce projet est l'occasion de

constater que vous avez dès à présent les moyens de commencer à traiter automatiquement ce type de données ; ou d'autres ! Par exemple des résultats d'expériences de physique.

À noter : en d'autres occasions, la bonne approche serait d'utiliser des bibliothèques existantes d'analyse de données et d'apprentissage automatique ; on peut par exemple penser à pandas et scikit-learn. Pour des traitements simples sur de petites données, un tableur suffirait.

Ici nous n'utiliserons que la bibliothèque standard de C++ (et éventuellement la SDL / libMLV pour les graphiques), quitte à réimplanter des outils de base ; en effet l'objectif est aussi de mettre en pratique ce que vous avez appris dans le cours Info 111 et de mieux comprendre comment fonctionnent ces outils.

Exercice 1 (De l'eau, de l'eau)**.**
Dans ce premier exercice, nous ferons des statistiques simples sur un jeu de données contenant les volumes d'eau distribués mensuellement à Paris en 2011.

`http://opendata.paris.fr/explore/dataset/volumes_d_eau_distribues/?tab=table`

Pour simplifier, nous vous fournissons dans l'archive un fichier
`donnees/volumes_d_eau_distribues.txt`
contenant ces données dans un format texte simple : chaque ligne contient un mois et le volume d'eau distribuée ce mois-ci.

(1) Télécharger et extraire l'archive `Projet-DonneesLibres.zip`.
(2) Dans le dossier `donnees`, ouvrez le fichier `volumes_d_eau_distribues.txt`. **Ne le modifiez pas : c'est le fichier que votre programme va analyser.**
(3) Ouvrez le fichier `eau-total.cpp` et complétez la fonction `main` qui calcule et affiche le volume total d'eau distribué à Paris en 2011. **La bonne réponse est 191601800. Vérifiez que c'est ce que vous obtenez.**
(4) Ouvrez le fichier `eau-moyen.cpp` et complétez la fonction `main` qui calcule et affiche le volume d'eau moyen distribué par mois en 2011. On calculera en nombre entier, **vous devez trouver 15966816, vérifiez que c'est bien le cas**.
(5) Ouvrez le fichier `eau-max.cpp` et complétez la fonction `main` qui calcule et affiche le mois où le plus grand volume d'eau a été distribué en 2011, ainsi que ce volume d'eau. **Vérifiez grâce au fichier de données que vous avez la bonne réponse**

Exercice 2 (Traitement des déchets)**.**
Les tonnages de déchets des bacs jaunes récoltés chaque mois de 2011 dans chaque arrondissements de Paris sont disponibles sur :

`http://opendata.paris.fr/explore/dataset/tonnages_des_dechets_bacs_jaunes/?tab=table`

Malheureusement les totaux annuels n'ont pas été inclus comme c'est le cas pour :

`http://opendata.paris.fr/explore/dataset/tonnages_des_dechets_bacs_verts/?tab=table`

Complétez le programme `dechets.cpp` qui utilise le fichier simplifié
`donnees/tonnages_des_dechets_bacs_jaunes.txt`
de l'archive pour calculer le tonnage annuel de déchets des bacs jaunes par arrondissement. Le programme affichera l'arrondissement qui produit le plus de déchets des bacs jaunes par année, ainsi que le tonnage annuel de déchets des bacs jaunes produit par cet arrondissement.

Exercice 3.
On souhaite effectuer **dans un même programme** les différents calculs sur le volume d'eau. Pour ne pas lire le fichier plusieurs fois, on le lit une unique fois et on le transforme en un tableau : les indices du tableau seront les mois et les valeurs, les volumes d'eau correspondant au mois.

Attention ! Les numéros de mois commencent à 1 mais les indices de tableau commencent à 0, on sera obligé de décaler : le mois de janvier aura l'indice 0 et le mois de décembre l'indice 11.

Complétez le fichier `eau-tout-en-un.cpp` contenant plusieurs fonctions dont la documentation est donnée. Après avoir écrit chaque fonction, vous rajouterez dans la fonction `main` un appel à la fonction de test correspondante. **Vérifiez que les tests passent avant de passer à la suite !**

Une fois que tous les tests passent, rajoutez dans la fonction `main` l'affichage du volume total sur l'année, du volume moyen par mois, et du mois avec le volume maximum ainsi que son volume. Voilà l'affichage que vous devez obtenir :

```
Le volume total d'eau distribué dans l'année est de :191601800
Le volume moyen par mois d'eau distribué dans l'année est de : 15966816
Le mois où on a distribué le plus d'eau est 5 avec un volume de 17711200
```

Bonus : affichez le nom du mois plutôt que son numéro !

Exercice 4 (Vers une bibliothèque générique)**.**
On souhaite à présent pouvoir analyser différents fichiers de données sans réécrire le même code plusieurs fois. Le but est d'obtenir une mini-bibliothèque de fonctions réutilisables.

(1) Dans le fichier `dechets-tableaux.cpp`, complétez la fonction `litTableauInt` qui transforme un fichier en un tableau 2 dimensions. Testez votre fonction avec les tests proposés.

(2) Dans ce même fichier, complétez la fonction `colonne` dont la documentation est donnée et testez-la.

(3) Copiez les fonctions `somme`, `moyenne` et `indiceMax` que vous avez déjà écrites.

(4) Complétez la fonction `main` pour que, en plus des tests, le programme demande à l'utilisateur d'entrer un mois et lui donne :
- la somme des déchets sur l'ensemble des arrondissements pour le mois
- la moyenne des déchets par arrondissement pour le mois
- l'arrondissement avec le plus de déchets pour le mois et son tonnage de déchets.

Voilà un exemple d'exécution du programme :

```
Entrez un numéro de mois (entre 1 et 12) : 4
La somme des déchets pour ce mois est de : 6649
La moyenne des déchets pour ce mois est de : 332
L'arrondissement avec le plus de déchets pour ce mois est : 75015
avec 835 tonnes de déchets.
```

Exercice 5.
Pour éviter la duplication de code, nous allons maintenant regrouper toutes les fonctions dans une bibliothèque `tableau-donnees` et utiliser la compilation séparée.

(3) Compléter le fichier `tableau-donnees.cpp` (dont vous avez déjà écrits toutes les fonctions sauf `sommePartielle`). Voir le fichier `tableau-donnees.h` pour la documentation.

(4) Compléter les tests dans `tableau-donnees-test.cpp`.

(5) Pour compiler un programme dépendant de plusieurs fichier, on doit compiler chaque fichier séparément. Pour automatiser les compilations, on utilise un fichier `Makefile` qui permet de sauvegarder les dépendances de fichiers et les lignes de compilations associées. Lancez la commande `make tableau-donnees-test` dans le terminal puis exécutez le fichier `tableau-donnees-test` créé pour tester vos fonctions.

(6) Complétez le fichier `dechets-tableaux-2` qui ne contenir que les fonctions décrites.

(7) Compilez le fichier avec la commande `make dechets-tableaux2` puis exécutez-le. Voilà un exemple d'exécution du programme

```
Entrez un numéro de mois (entre 1 et 12) : 4
La somme des déchets pour ce mois est de : 6649
La moyenne des déchets pour ce mois est de : 332
L'arrondissement avec le plus de déchets pour ce mois est : 75015
avec 835 tonnes de déchets.
---
La somme des déchets annuel est de 82148
La moyenne des déchets pour l'année est de : 4107
L'arrondissement avec le plus de déchets l'année est : 75015
avec 10163 tonnes de déchets.
```

(8) Faites de même avec le fichier `eau-tout-en-un2.cpp` en utilisant les fonctions de la bibliothèque. (attention, on travaillera cette fois avec des tableaux à 2 dimensions donc les indices ne correspondront plus directement aux mois !)

Exercice 6 (Un premier fichier CSV)**.**
L'objectif est de reprendre l'exercice 1, mais en partant du fichier de données tel qu'il est fourni par `opendata.paris.fr` : `donnees/volumes_d_eau_distribues.csv`

Ce fichier est en format CSV (Comma Separated Values). C'est un format texte répandu pour représenter des données tabulaires. D'ailleurs, vous pouvez charger ce type de fichier directement dans un tableur, par exemple pour vérifier à la main que vos résultats sont corrects.

Chaque ligne de texte représente une ligne du tableau, et les cellules sont séparées par des point-virgules. La première ligne d'un fichier CSV est une ligne d'entête qui correspond au nom des colonnes du tableau.

(1) Consulter et essayer le programme `getline-exemple` fourni dans l'archive. Il utilise la fonction `getline`, dont la documentation est donnée ci-dessous.

```
/** @file **/
/** Lit une ligne d'un flux et la stocke dans 'resultat'
 * @param f un flux entrant
 * @param resultat une chaîne de caractères
 * @return le flux f
 **/
istream& getline(istream &f, string &resultat);

/** Lit une chaîne de caractères depuis un flux jusqu'au séparateur
    et la stocke dans 'resultat'
 * @param f un flux entrant
 * @param resultat une chaîne de caractères
 * @param separateur un caractère
 * @return le flux f
 **/
istream& getline(istream &f, string &resultat, char separateur);
```

Vous constaterez que cette fonction *modifie* la chaîne de caractère 'resultat' passée en argument ! De fait, le symbole '&' indique que 'resultat' est passé *par référence*. Vous verrez les détails au second semestre.

(2) À l'aide de la fonction `getline`, implanter les programmes `eau-moyen-csv.cpp`, `eau-total-csv.cpp`, `eau-max-csv.cpp` qui font les mêmes calculs que leurs prédécesseurs, mais à partir du fichier CSV.

Exercice 7 (Généralisation)**.**
L'objectif de cet exercice est d'implanter un début de bibliothèque générique `tableau-donnees-csv.cp` pour les fichiers CSV. Des tests sont fournis dans `tableau-donnees-csv-test.cpp`. Les fonctionalités seront les suivantes :

(1) Afficher un tableau à deux dimensions (usage principal : débogage).

```
/** Affiche le contenu d'un tableau de chaînes à deux dimensions
 * @param tableau un tableau à deux dimensions
 **/
void afficheTableau(vector<vector<string>> tableau);
```

(2) Construire un tableau 2D de chaînes de caractères lu depuis un fichier CSV.

```
/** Construction d'un tableau 2D de chaînes lu depuis un fichier CSV
 * @param fichier le nom d'un fichier contenant un nombre fixe
  * d'entiers par lignes, séparés par des espaces
 * @param nb_colonnes le nombre de colonnes du fichier
 * @return un tableau d'entiers à deux dimensions
 **/
vector<vector<string>> litTableauCSV(string fichier, int nb_colonnes);
```

Si vous avez des difficultés, vous pouvez copier-coller l'implantation fournie dans le fichier `en_cas_d_urgence_briser_la_glace.cpp`. On vous demandera tout de même de savoir expliquer comment elle marche !

(3) ♣ Même chose, mais sans avoir besoin de spécifier le nombre de colonnes

```
/** Construction d'un tableau 2D de chaînes lu depuis un fichier CSV
 * @param fichier le nom d'un fichier contenant un nombre fixe
  * d'entiers par lignes, séparés par des espaces
 * @return un tableau d'entiers à deux dimensions
 **/
vector<vector<string>> litTableauCSV(string fichier);
```

(4) Extraire une colonne d'un tableau de chaînes de caractères.

```
/** Extraction d'une colonne d'un tableau 2D de chaînes de caractères
 * @param t un tableau 2D de chaînes de caractères
 * @param i un numéro de colonne
 * @return la colonne i, représentée par un vecteur
 **/
vector<string> colonne(vector<vector<string>> t, int i);
```

(5) Convertir une colonne de chaînes de caractères en colonne numérique (int, float, double, ...). Indication : utiliser les string stream.

```
/** Conversion d'un vecteur de chaînes de caractères en vecteur d'entiers
 * @param t un vecteur de chaînes de caractères
 * @return le vecteur, converti en vecteur d'entiers
 **/
vector<int> conversionInt(vector<string> t);

/** Conversion d'un vecteur de chaînes de caractères en vecteur de doubles
```

```
 * @param t un vecteur de chaînes de caractères
 * @return le vecteur, converti en vecteur de doubles
 **/
vector<double> conversionDouble(vector<string> t);
```

(6) ♣ Version avancée : éviter la duplication avec une unique fonction de conversion en utilisant un template.

```
/** Conversion en vecteur de valeurs de type T
 * @param t un vecteur de chaînes de caractères
 * @return le vecteur, converti en vecteur de T
 **/
template<class T>
vector<T> conversion(vector<string> t);
```

Exercice 8 (Applications)**.**

(1) Quel est le genre et l'espère de l'arbre le plus haut de Paris ? Le plus vieux ?
`http://opendata.paris.fr/explore/dataset/arbresremarquablesparis2011/`

(2) Calculer le tonnage annuel de déchets des bacs jaunes par arrondissement en partant du fichier CSV d'origine.
`http://opendata.paris.fr/explore/dataset/tonnages_des_dechets_bacs_jaunes/?tab=table`

(3) Quel arrondissement produit le plus de déchets ?

(4) Quelles sont les deux stations de métro / velib / ... les plus proches l'une de l'autre à Paris.
`https://www.data.gouv.fr/fr/datasets/velib-paris-et-communes-limitrophes-idf/`

Exercice 9.

(1) En utilisant la bibliothèque graphique du TP 8, dessiner la carte de toutes les communes de France à partir du fichier CSV fourni sur :
`https://www.data.gouv.fr/fr/datasets/decoupage-administratif-communal-francais-issu`
On pourra par exemple représenter chaque commune par un cercle de centré sur ses coordonnées géographiques et de rayon proportionnel à la population.

(2) Positionner sur cette carte d'autres éléments intéressants.

(3) Positionner sur cette carte plusieurs barycentres de la france (population / surface / ...)

3. Pour aller plus loin

Choisir une ou plusieurs de ces extensions :

Exercice ♣ 10.

(1) Quelle est la voiture commercialisée en France qui consomme le plus ? Le moins ?
`https://www.data.gouv.fr/fr/datasets/emissions-de-co2-et-de-polluants-des-vehicule`

(2) Quelle marque, en moyenne sur toute la gamme commercialisée en France, produit les véhicules qui consomment le plus ? le moins ?

Exercice ♣ 11.

Lister les stations de métros à Paris où ont été retrouvés des lecteurs MP3.

Exercice ♣ 12.
Implanter l'application mentionnée dans l'introduction (RéveilRER) sous forme d'une application Android.

Exercice ♣ 13.
À vous de trouver des applications intéressantes des données ouvertes !

4. Références

– `http://opendata.paris.fr`
– `http://data.gouv.fr`
– `http://www.ideeslibres.org/`