université PARIS-SACLAY | FACULTÉ DES SCIENCES D'ORSAY

Programmation Impérative

Données Libres

Rapport

SAMINADIN Samuel et RAVENDIRANE Gayathiri

Licence Portail MI, L1 S1 MI4

Année Universitaire : 2022-2023

Responsable de TD :

Daniel Tamayo, Daniel.Tamayo-Jimenez@universite-paris-saclay.fr

I. Auteurs

- RAVENDIRANE Gayathiri
- SAMINADIN Samuel
- Professeurs (pour l'exercice 8)

II. Résumé du travail effectué

- Exercice 1 :

Auteur : Samuel
Fiabilité : 99,9 % Toute chose a des imperfections.
Fichiers cpp complétés, compilent et passent les tests.

1)

```
(base) samuel.saminadin@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$ info-111 g++ mariage-total.cpp -o ma
riage-total
INFO:travo:Running: g++ -g -Wall -pedantic -Wno-sign-compare -Wno-unused-value -std=c++17 -DNDEBUG -D_FORTIFY_SOURCE=2 -O2 -isystem /opt/
conda/include -Wl,-O2 -Wl,--sort-common -Wl,--as-needed -Wl,-z,relro -Wl,-z,now -Wl,--disable-new-dtags -Wl,--gc-sections -Wl,-rpath,/opt
/conda/lib -Wl,-rpath-link,/opt/conda/lib -L/opt/conda/lib mariage-total.cpp -o mariage-total
(base) samuel.saminadin@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$ ./mariage-total
Le nombre total de mariages celebres a Paris entre 2010 et 2015 est 9269
(base) samuel.saminadin@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$
```

2)

```
(base) gayathiri.ravendirane@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$ info-111 g++ mariage
-samedi.cpp -o mariage-samedi
INFO:travo:Running: g++ -g -Wall -pedantic -Wno-sign-compare -Wno-unused-value -std=c++17 -DNDEBUG -D_FORTIFY_SOURCE=2 -O2 -is
ystem /opt/conda/include -Wl,-O2 -Wl,--sort-common -Wl,--as-needed -Wl,-z,relro -Wl,-z,now -Wl,--disable-new-dtags -Wl,--gc-se
ctions -Wl,-rpath,/opt/conda/lib -Wl,-rpath-link,/opt/conda/lib -L/opt/conda/lib mariage-samedi.cpp -o mariage-samedi
(base) gayathiri.ravendirane@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$ ./mariage-samedi
Le nombre total de mariages celebres un samedi a Paris entre 2010 et 2015 est 31418
(base) gayathiri.ravendirane@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$
```

- Exercice 2 :

Auteur : Gayathiri
Fiabilité : 99,9 %
Fichiers cpp complétés, compilent et passent les tests.

```
(base) samuel.saminadin@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$ info-111 g++ prenoms.cpp -o prenoms
INFO:travo:Running: g++ -g -Wall -pedantic -Wno-sign-compare -Wno-unused-value -std=c++17 -DNDEBUG -D_FORTIFY_SOURCE=2 -O2 -isystem /opt/
conda/include -Wl,-O2 -Wl,--sort-common -Wl,--as-needed -Wl,-z,relro -Wl,-z,now -Wl,--disable-new-dtags -Wl,--gc-sections -Wl,-rpath,/opt
/conda/lib -Wl,-rpath-link,/opt/conda/lib -L/opt/conda/lib prenoms.cpp -o prenoms
(base) samuel.saminadin@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$ ./prenoms
Entrez une année entre 2006 et 2021 : 2015
En 2015, il y a eu 31970 naissancess
Le prenom le plus donne a ete : Adam (donne 356 fois).
(base) samuel.saminadin@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$
```

- Exercice 3 :

Auteur : Gayathiri
Fiabilité : 99,9 %
Fichiers cpp complétés, compilent et passent les tests.

```
(base) gayathiri.ravendirane@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$ info-111 g++ mariage-complet.cpp -o mariage-complet
INFO:travo:Running: g++ -g -Wall -pedantic -Wno-sign-compare -Wno-unused-value -std=c++17 -DNDEBUG -D_FORTIFY_SOURCE=2 -O2 -isystem /opt/conda/include -Wl,-O2 -Wl,--sort-commo
n -Wl,--as-needed -Wl,-z,relro -Wl,-z,now -Wl,--disable-new-dtags -Wl,--gc-sections -Wl,-rpath,/opt/conda/lib -Wl,-rpath-link,/opt/conda/lib -L/opt/conda/lib mariage-complet.c
pp -o mariage-complet
(base) gayathiri.ravendirane@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$ ./mariage-complet
Le nombre de total de mariages celebres entre 2010 et 2015 est de 55342
Le nombre de mariages celebres en moyenne par an est de 9223
L'annee ou l'on a celebre le plus de mariages est 2014 avec 9866 mariages
Le jour de la semaine ou l'on a celebre le plus de mariages est le Samedi avec 31418 mariages
Le pourcentage de mariages celebres le samedi est de 56.7706%
(base) gayathiri.ravendirane@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$
```

- Exercice 4 :

Auteur : Samuel

Fiabilité : 99,9 %

Fichiers cpp complétés, compilent et passent les tests.

Les tests :

```
(base) gayathiri.ravendirane@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$ info-111 g++ prenoms-tableau.cpp -o prenoms-tableau
INFO:travo:Running: g++ -g -Wall -pedantic -Wno-sign-compare -Wno-unused-value -std=c++17 -DNDEBUG -D_FORTIFY_SOURCE=2 -O2 -isystem /opt/conda/include -Wl,-O2 -Wl,--sort-commo
n -Wl,--as-needed -Wl,-z,relro -Wl,-z,now -Wl,--disable-new-dtags -Wl,--gc-sections -Wl,-rpath,/opt/conda/lib -Wl,-rpath-link,/opt/conda/lib -L/opt/conda/lib prenoms-tableau.c
pp -o prenoms-tableau
(base) gayathiri.ravendirane@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$ ./prenoms-tableau
Lancement des test de afficheTableau :
M 2011 Bubulle 3
F 2012 Bichette 4
F 2011 Babouche 7
F 2011 Ziboulette 1
Lancement des test de litTableau
Lancement des test de colonne
Lancement des test de selectLignes
Lancement des test de conversionInt
(base) gayathiri.ravendirane@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$
```

Les affichages en fonctions des prénoms choisi :

- Octave :

```
(base) gayathiri.ravendirane@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$ ./prenoms-tableau
Nombre total de naissance : 571717
Choisissez un prenom : Octave
Le prenom Octave a ete donne a 839 garcon entre 2006 et 2021
L'annee la plus forte est 2021 avec 82 enfants
Le prenom Octave n'a ete donne a aucun fille entre 2006 et 2021(base) gayathiri.ravendirane@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$
```

- Agathe :

```
(base) gayathiri.ravendirane@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$ ./prenoms-tableau
Nombre total de naissance : 571717
Choisissez un prenom : Agathe
Le prenom Agathe n'a ete donne a aucun garcon entre 2006 et 2021
Le prenom Agathe a ete donne a 1476 fille entre 2006 et 2021
L'annee la plus forte est 2017 avec 111 enfants
(base) gayathiri.ravendirane@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$
```

- Camille :

```
(base) gayathiri.ravendirane@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$ ./prenoms-tableau
Nombre total de naissance : 571717
Choisissez un prenom : Camille
Le prenom Camille a ete donne a 1177 garcon entre 2006 et 2021
L'annee la plus forte est 2019 avec 97 enfants
Le prenom Camille a ete donne a 3859 fille entre 2006 et 2021
L'annee la plus forte est 2006 avec 287 enfants
(base) gayathiri.ravendirane@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$
```

- Exercice 5 :

Auteur : Samuel

Fiabilité : 99,9 %

Fichiers cpp complétés, compilent et passent les tests.

```
(base) gayathiri.ravendirane@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$ ./mariage-complet-2
Le nombre de total de mariages celebres entre 2010 et 2015 est de 55342
Le nombre de mariages celebres en moyenne par an est de 9223
L'annee ou l'on a celebre le plus de mariages est 2014 avec 9866 mariages
Le jour de la semaine ou l'on a celebre le plus de mariages est le Vendredi avec 9866 mariages
Le pourcentage de mariages celebres le samedi est de 56.7706%
(base) gayathiri.ravendirane@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$
```

- Exercice 6 :

Auteur : Gayathiri

Fiabilité : 99.9 %

Compile bien avec les fonctions écrites (sauf les fonctions avec templates).

```
(base) gayathiri.ravendirane@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$ make tableau-donnees-avance-test
/opt/conda/bin/x86_64-conda-linux-gnu-c++ -Wall -pedantic -std=c++11 -g -Wno-sign-compare -DNDEBUG -D_FORTIFY_SOURCE=2 -O2 -isystem /opt/conda/include  -c -o tableau-donnees-avance.o tableau-donnees-avance.cpp
tableau-donnees-avance.cpp: In instantiation of 'std::vector<T> conversion(std::vector<std::__cxx11::basic_string<char> >) [with T = int]':
tableau-donnees-avance.cpp:69:54:   required from here
tableau-donnees-avance.cpp:61:1: warning: no return statement in function returning non-void [-Wreturn-type]
   61 | }
      | ^
tableau-donnees-avance.cpp: In instantiation of 'std::vector<T> conversion(std::vector<std::__cxx11::basic_string<char> >) [with T = double]':
tableau-donnees-avance.cpp:70:60:   required from here
tableau-donnees-avance.cpp:61:1: warning: no return statement in function returning non-void [-Wreturn-type]
tableau-donnees-avance.cpp: In instantiation of 'std::vector<T> groupBy(std::vector<std::vector<std::__cxx11::basic_string<char> > >, std::vector<std::__cxx11::basic_string<char> >, int, int) [with T = int]':
tableau-donnees-avance.cpp:71:102:   required from here
tableau-donnees-avance.cpp:66:1: warning: no return statement in function returning non-void [-Wreturn-type]
   66 | }
      | ^
tableau-donnees-avance.cpp: In instantiation of 'std::vector<T> groupBy(std::vector<std::vector<std::__cxx11::basic_string<char> > >, std::vector<std::__cxx11::basic_string<char> >, int, int) [with T = double]':
tableau-donnees-avance.cpp:72:108:   required from here
tableau-donnees-avance.cpp:66:1: warning: no return statement in function returning non-void [-Wreturn-type]
/opt/conda/bin/x86_64-conda-linux-gnu-c++ -Wall -pedantic -std=c++11 -g -Wno-sign-compare -DNDEBUG -D_FORTIFY_SOURCE=2 -O2 -isystem /opt/conda/include -Wl,-O2 -Wl,--sort-common -Wl,--as-needed -Wl,-z,relro -Wl,-z,now -Wl,--disable-new-dtags -Wl,--gc-sections -Wl,-rpath,/opt/conda/lib -Wl,-rpath-link,/opt/conda/lib -L/opt/conda/lib  tableau-donnees-avance-test.o tableau-donnees-avance.o   -o tableau-donnees-avance-test
(base) gayathiri.ravendirane@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$
```

```
(base) gayathiri.ravendirane@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$ ./tableau-donnees-avance-test
Lancement des tests de cherche indice
Lancement des tests de distinct
Lancement des tests de group by (int)
Lancement des tests de conversion double
Lancement des tests de group by (double)
(base) gayathiri.ravendirane@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$
```

```
(base) gayathiri.ravendirane@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$ ./prenoms-tableau-avance
Il y a eu 295151 naissance de garcon et 276566 naissance de filles
L'annee qui a eu le plus de naissance est 2010 avec 33259 naissances
En moyenne naissent 31762 enfants par an
Il y a eu 1410 prenoms de fille differents et 1317 prenoms de garcons
Le prenoms feminin le plus populaire est Angie0 naissances
Le prenoms masculin le plus populaire est Ari avec 0 naissances
(base) gayathiri.ravendirane@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$
```

- Exercice 7 :

Auteurs : Gayathiri et Samuel

Fiabilité : 99,9 %

Fichiers cpp complétés, compilent et passent les tests.

```
(base) gayathiri.ravendirane@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$ info-111 g++ prenoms-csv.cpp -o prenoms-csv
INFO:travo:Running: g++ -g -Wall -pedantic -Wno-sign-compare -Wno-unused-value -std=c++17 -DNDEBUG -D_FORTIFY_SOURCE=2 -O2 -isystem /opt/conda/include -Wl,-O2 -Wl,--sort-common -Wl,--as-needed -Wl,-z,relro -Wl,-z,now -Wl,--disable-new-dtags -Wl,--gc-sections -Wl,-rpath,/opt/conda/lib -Wl,-rpath-link,/opt/conda/lib -L/opt/conda/lib prenoms-csv.cpp -o prenoms-csv
(base) gayathiri.ravendirane@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$ ./prenoms-csv
Entrez une année entre 2006 et 2021 : 2015
En 2015, il y a eu 31970 naissancess
Le prénom le plus donné a été : Adam (donné 356 fois)
(base) gayathiri.ravendirane@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$
```

```
#include <stdexcept>
/** @file **/
```

```cpp
#include <fstream>
#include <iostream>
#include <string>
#include <sstream>
using namespace std;

/** Calcule et affiche le prénom le plus donné une année donnée
 * ainsi que le nombre de naissance cette année là **/
int main() {
    int choice;
    cout << "Entrez une année entre 2006 et 2021 : " ;
    cin >> choice;

    ifstream fichier("donnees/liste_des_prenoms.csv");
    string header;
    getline(fichier, header);
    string given;
    string sexe;
    string year;
    string name;
    string nb;
    int occurence;

    int occurence_max = 0;
    int total = 0;
    string name_max;

    while (fichier) {
        getline(fichier, given, ';');
        getline(fichier, sexe, ';');
        getline(fichier, year, ';');
        getline(fichier, name, ';');
        getline(fichier, nb);
        if (nb.size() > 0 and nb[nb.length()-1] == '\r') {
            nb.resize(nb.length()-1);
        }

        istringstream(nb) >> occurence;
        if (fichier) {
            if (stoi(year) == choice) { // convert into int
                total += occurence;
                if (occurence > occurence_max) {
                    name_max = name;
```

```
                    occurence_max = occurence;
                }
            }
        }
    }
    fichier.close();
    cout << "En" << choice << ", il y a eu " << total << " naissances"
<< endl;
    cout << "Le prenom le plus donne a ete " << name_max << " (donne "
<< occurence_max << " fois)";
    return 0;
}
```

- Exercice 8 :

Auteur : PROFS
Fiabilité : 100 %.
Fichiers cpp compilés et passent les tests

- Exercice 9 :

Auteurs : Gayathiri et Samuel
Fiabilité : 99,9 % Toute chose a des imperfections (question 1 et 2).
Fichiers cpp complétés, compilent et passent les tests.

1) Nom, genre et espèce de l'arbre le plus haut de Paris

```
(base) gayathiri.ravendirane@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$ make arbres-hauteur
make: 'arbres-hauteur' is up to date.
(base) gayathiri.ravendirane@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$ ./arbres-hauteur
Nom francais : Platane, genre : Platanus, espece de l'arbre le plus haut de Paris : x hispanica avec hauteur : 45metre(s).
(base) gayathiri.ravendirane@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$
```

```
#include <stdexcept>
/** @file **/
#include <iostream>
#include "tableau-donnees.hpp"
#include "tableau-lecture-csv.hpp"
using namespace std;

/** Affiche le genre et l'espece de l'arbre le plus haut de Paris
 * parmi les "arbres remarquables"
 **/
int main() {
    vector<vector<string>> data = litTableauCSV("donnees/arbresremarquablesparis.csv");
    vector<int> heigth = conversionInt(colonne(data, 12));
    int i = indiceMax(heigth);
    vector<string> tree = data[i];
    cout << "Nom francais : " << tree[1] << ", genre : " << tree[2] << ", espece de l'arbre le plus haut de Paris : " << tree[3] << " avec hauteur : " << heigth[i]
<< "metre(s)." << endl;
}
```

2) Nombre de genre "Platanus"

```
(base) gayathiri.ravendirane@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$ make arbres-platanus
make: 'arbres-platanus' is up to date.
(base) gayathiri.ravendirane@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$ ./arbres-platanus
Il y a 32 de genre de platanus d'ardres et 2d'especes differentes pour les platanus.
(base) gayathiri.ravendirane@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$
```

```cpp
#include <stdexcept>
/** @file **/
#include <iostream>
#include "tableau-donnees.hpp"
#include "tableau-donnees-avance.hpp"
#include "tableau-lecture-csv.hpp"
using namespace std;

/** Affiche le nombre d'arbres du genre Platanus, et le nombre d'espèces
 *  différentes pour ce genre Platanus, parmi les "arbres remarquables"
 **/
int main() {
    vector<vector<string>> data = litTableauCSV("donnees/arbresremarquablesparis.csv");
    vector<vector<string>> platanus = selectLignes(data, 2, "Platanus");
    vector<string> species = distinct(platanus, 3);
    cout << "Il y a " << platanus.size() << " de genre de platanus d'ardres et " << species.size() << "d'especes
differentes pour les platanus." << endl;

}
```

3)

```cpp
int main() {
    vector<vector<string>> data =
litTableauCSV("donnees/statistiques_de_creation_d_actes_d_etat_civil_par_arrondissement.csv", 4);
    vector<string> years = distinct(data, 1);
    vector<vector<string>> births = selectLignes(data, 0, "Naissances");
    vector<int> nb = groupBy<int>(births, years, 1, 3); // problème
    int max = indiceMax(nb);
    cout << "Il y eu le plus de declaration de naissance en " << years[max] << endl;


    vector<vector<string>> mariages = selectLignes(data, 0, "Mariages");
    nb = groupBy<int>(mariages, years, 1, 3);
    max = indiceMax(nb);
    cout << "Il y eu le plus de declaration de mariages en " << years[max] << endl;
}
```

```
(base) gayathiri.ravendirane@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$ make actes-civils
make: 'actes-civils' is up to date.
(base) gayathiri.ravendirane@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$ ./actes-civils
Segmentation fault (core dumped)
(base) gayathiri.ravendirane@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$
```

```
(base) gayathiri.ravendirane@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$ make actes-civils
make: 'actes-civils' is up to date.
(base) gayathiri.ravendirane@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$ ./actes-civils
Il y eu le plus de declaration de naissance en
free(): invalid pointer
Aborted (core dumped)
(base) gayathiri.ravendirane@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$
```

- <u>Exercice 15 :</u>

Auteurs : Gayathiri et Samuel
Fichiers cpp complétés, mais ne compilent pas. (Application : bilan social)

```cpp
int main() {
    vector<vector<string>> data = litTableauCSV("donnees/bilan_social.csv");

    int nb_femme = somme(conversionInt(colonne(data, 5)));
    cout << "Il y a " << nb_femme << " de femmes en total" << endl;

    int nb_homme = somme(conversionInt(colonne(data, 6)));
    cout << "Il y a " << nb_homme << " de femmes en total" << endl;

    int nb_temps = moyenne(conversionInt(colonne(data, 6)));
    cout << "En moyenne " << nb_homme << " d'heure" << endl;
```

```
(base) gayathiri.ravendirane@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$ make bilan_social
/opt/conda/bin/x86_64-conda-linux-gnu-c++ -Wall -pedantic -std=c++11 -g -Wno-sign-compare -DNDEBUG -D_FORTIFY_SOURCE=2 -O2 -isystem /opt/conda/include -Wl,-O2 -Wl,--sort-common -Wl,--as-needed -Wl,-z,relro -Wl,-z,now -Wl,--
disable-new-dtags -Wl,--gc-sections -Wl,-rpath,/opt/conda/lib -Wl,-rpath-link,/opt/conda/lib -L/opt/conda/lib  bilan_social.cpp   -o bilan_social
bilan_social.cpp: In function 'int main()':
bilan_social.cpp:18:9: warning: unused variable 'nb_temps' [-Wunused-variable]
   18 |     int nb_temps = moyenne(conversionInt(colonne(data, 6)));
      |         ^~~~~~~~
/opt/conda/bin/../lib/gcc/x86_64-conda-linux-gnu/9.4.0/../../../../x86_64-conda-linux-gnu/bin/ld: /tmp/ccdHWsPJ.o: in function `main':
/home/jovyan/ProgImperative/Projet-DonneesLibres/bilan_social.cpp:10: undefined reference to `litTableauCSV(std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >)'
/opt/conda/bin/../lib/gcc/x86_64-conda-linux-gnu/9.4.0/../../../../x86_64-conda-linux-gnu/bin/ld: /home/jovyan/ProgImperative/Projet-DonneesLibres/bilan_social.cpp:12: undefined reference to `colonne(std::vector<std::vector
<std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >, std::allocator<std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > >, std::allocator<std::vector<std::__cxx11::
basic_string<char, std::char_traits<char>, std::allocator<char> >, std::allocator<std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > > > > >, int)'
/opt/conda/bin/../lib/gcc/x86_64-conda-linux-gnu/9.4.0/../../../../x86_64-conda-linux-gnu/bin/ld: /home/jovyan/ProgImperative/Projet-DonneesLibres/bilan_social.cpp:12: undefined reference to `conversionInt(std::vector<std::
__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >, std::allocator<std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > > >)'
/opt/conda/bin/../lib/gcc/x86_64-conda-linux-gnu/9.4.0/../../../../x86_64-conda-linux-gnu/bin/ld: /home/jovyan/ProgImperative/Projet-DonneesLibres/bilan_social.cpp:12: undefined reference to `somme(std::vector<int, std::all
ocator<int> >)'
/opt/conda/bin/../lib/gcc/x86_64-conda-linux-gnu/9.4.0/../../../../x86_64-conda-linux-gnu/bin/ld: /home/jovyan/ProgImperative/Projet-DonneesLibres/bilan_social.cpp:15: undefined reference to `colonne(std::vector<std::vector
<std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >, std::allocator<std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > > >, std::allocator<std::vector<std::__cxx11::
basic_string<char, std::char_traits<char>, std::allocator<char> >, std::allocator<std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > > > > >, int)'
/opt/conda/bin/../lib/gcc/x86_64-conda-linux-gnu/9.4.0/../../../../x86_64-conda-linux-gnu/bin/ld: /home/jovyan/ProgImperative/Projet-DonneesLibres/bilan_social.cpp:15: undefined reference to `conversionInt(std::vector<std::
__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >, std::allocator<std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > > >)'
/opt/conda/bin/../lib/gcc/x86_64-conda-linux-gnu/9.4.0/../../../../x86_64-conda-linux-gnu/bin/ld: /home/jovyan/ProgImperative/Projet-DonneesLibres/bilan_social.cpp:15: undefined reference to `somme(std::vector<int, std::all
ocator<int> >)'
/opt/conda/bin/../lib/gcc/x86_64-conda-linux-gnu/9.4.0/../../../../x86_64-conda-linux-gnu/bin/ld: /home/jovyan/ProgImperative/Projet-DonneesLibres/bilan_social.cpp:18: undefined reference to `colonne(std::vector<std::vector
<std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >, std::allocator<std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > > >, std::allocator<std::vector<std::__cxx11::
basic_string<char, std::char_traits<char>, std::allocator<char> >, std::allocator<std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > > > > >, int)'
/opt/conda/bin/../lib/gcc/x86_64-conda-linux-gnu/9.4.0/../../../../x86_64-conda-linux-gnu/bin/ld: /home/jovyan/ProgImperative/Projet-DonneesLibres/bilan_social.cpp:18: undefined reference to `conversionInt(std::vector<std::
__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >, std::allocator<std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > > >)'
/opt/conda/bin/../lib/gcc/x86_64-conda-linux-gnu/9.4.0/../../../../x86_64-conda-linux-gnu/bin/ld: /home/jovyan/ProgImperative/Projet-DonneesLibres/bilan_social.cpp:18: undefined reference to `moyenne(std::vector<int, std::a
llocator<int> >)'
collect2: error: ld returned 1 exit status
make: *** [<builtin>: bilan_social] Error 1
(base) gayathiri.ravendirane@universite-paris-saclay.fr@jupyterhub:~/ProgImperative/Projet-DonneesLibres$
```

III. Motivations

Nombre d'heures consacrées : ≈ 45 heures (Code + rapport + préparation à la présentation)

D'abord, nous allons commencer par expliquer les raisons derrière notre choix du sujet "Données Libres". Au début, nous ne cacherons pas que le sujet sur le jeu Mixmo nous a surement paru plus intéressant que celui sur les données libres. Toutefois, nous n'avons pas très bien saisi les enjeux du sujet à savoir les règles et principes du jeu. De plus, bien que ce sujet ait l'avantage d'être un sujet libre, cela sous-entend également qu'il n'est pas forcément facile de démarrer car il est à nous de tout mettre en place depuis le début.

Cependant, cela n'était pas le cas avec le sujet données libres car ce dernier était construit de façon à avoir un fil directeur avec des questions guidées donc pour des "débutants" comme nous qui faisons pour la première fois de la vraie programmation, ce sujet nous a paru beaucoup plus abordable pour le délai donné. Un autre avantage qu'on a vu dans le choix de ce sujet est qu'il nous a obligé à exploiter et mettre en application quasiment la totalité des notions vues en cours, TD et en TP et ceci pour des exemples très concrets et dans un contexte précis. Ainsi, il nous a donné la possibilité de réimplanter à la fois des fonctions classiques plus ou moins similaires à celles vues en cours mais aussi et surtout des fonctions plus poussées ce qui nous a donc permis de découvrir des concepts plus avancés comme les templates class par exemple.

Enfin, nous avons également apprécié le fait que le sujet était conçu pour être de difficultés croissantes alors qu'avec le sujet Mixmo il est plus dur de savoir le niveau de difficulté attendu bien que le barème fournisse quelques éléments pour en avoir une idée.

IV. Organisation du travail

Nous avons principalement utilisé deux moyens pour travailler ensemble sur ce projet. Dans un premier temps, on s'est réparti le travail en partageant les exercices obligatoires. Ainsi, nous avons pu finir séparément les exercices un à six.

Mais, on s'est rapidement rendu compte que le projet a été conçu avec un fil directeur de questions progressives. Nous avons donc commencé à mettre en commun nos codes pour essayer de corriger par nous-même, tant que possible, les erreurs à la compilation et à l'exécution pour ces exercices.

Toutefois, cela n'a pas toujours été le cas car il n'était pas toujours évident de réussir à repérer la source de l'erreur et nous avons donc essayé de demander de l'aide au Bureau d'Intervention Pédagogique. Ce dernier nous a été très utile pour nous aider à retrouver nos erreurs pour les corriger par la suite. Ensuite, nous avons travaillé ensemble sur les exercices sept et neuf. *Étant bloqué pour l'exercice huit, nous avons fini par utiliser le code fourni par le professeur afin de pouvoir faire les applications dans l'exercice 9.*

Nous avons donc également consacré plusieurs heures à faire de la "programmation côte à côte". De temps en temps, nous avons donc aussi fait du "pair programming" pour travailler et analyser les erreurs sur le même bout de programme afin de rendre la correction plus rapide.
Pour déposer et récupérer les programmes des uns des autres, nous avons utilisé le Drive.

Finalement, nous avons pu importer tous les programmes complétés avec Visual Studio Code (outil principal pour écrire les codes) sur Jupyter ce qui nous a également permis de réaliser la compilation avec la commande make pour l'exercice 5 et de tout finaliser avant de déposer les programmes compilés et le rapport.

V. Présentation global des exercices et démonstration des exos 7 et 9

Les deux premiers exercices nous ont surtout permis de revoir des manipulations basiques de fonctions qui utilisent des fichiers. Par exemple, dans le premier exercice nous avons pu revoir comment calculer le nombre total de mariages à partir d'un fichier (question 1) ainsi que le nombre de mariages célébrés un jour en particulier comme le samedi (question 2).

#Ensuite, dans l'exercice deux, nous avons eu l'occasion d'utiliser l'instruction "*cin"* (entrée standard) en plus du "*cout*" (sortie standard) déjà utilisé pour faire des affichages dans l'exercice 1.

Par ailleurs, dans l'exercice 3, nous avons pu implanter quelques fonctions classiques tels que la fonction "*somme*", "*moyenne*" et "*IndiceMax*" qui ont été indispensable dans l'appel des fonctions dans main pour l'exercice 4. #De plus, avec l'exercice 4, nous avons également eu l'occasion de mettre en place quelques fonctions essentielles sur les tableaux à deux dimensions pour afficher un tableau 2D, extraire des lignes, colonnes ou encore transformer un fichier en tableau.

#Avec l'exercice 5 nous avons pu exploiter l'utilisation de la commande "Make" pour faire de la compilation séparée avec les programmes déjà écrits dans l'exercice 4. # L'exercice 6 nous a encore permis d'approfondir les traitements de tableaux et de surcroît découvrir la notion de template.

#Dans l'exercice 7, que nous allons maintenant aborder plus en détail, nous avons enfin pu traiter des fichiers CSV en s'inspirant du même modèle que l'exercice 2. Nous avons donc pu utiliser une fonction très importante pour cet exercice mais également pour la suite qui est la fonction getline qui permet de parcourir un fichier CSV ligne par ligne afin d'effectuer les mêmes calculs de l'exercice deux.

Finalement, avec l'exercice neuf, nous avons enfin eu l'occasion de mettre en application tout ce qu'on a implanté tout au long des exercices dans le contexte de nombres d'arbres dans une espèce comme le Platanus. Cet exercice nous a permis de réutiliser la fonction "*LitTableauCSV*" que nous avions déjà implanté dans l'exercice 8. Cette dernière est similaire à la fonction "*litTableau*" déjà implantée dans l'exercice 4 mais cette fois-ci pour un fichier *.csv* et non plus *.txt* .

VI. Prise de recul

Nous pensons que ce projet nous a été très utile pour plusieurs raisons.
Tout d'abord, il nous a permis de réutiliser le logiciel Visual Studio Code pour coder en C++ que nous avons déjà utilisé dans le cadre d'un devoir maison en Python. Ce logiciel nous a facilité la compréhension des messages d'erreurs avec IDE (Integrated Development Environment) une interface très agréable pour programmer. En effet, les problèmes de serveur avec Jupyter Hub et les bogues à l'exécution dans des programmes qui, pourtant, fonctionnent bien avec Jupyter en local ont été les sources de nos premières difficultés. Cela nous a empêché de travailler efficacement sur le projet depuis chez nous.

De plus, même si nous n'avons pas rencontré de problèmes aussi importants avec le serveur en local, la compilation était parfois tout de même assez lente. Nous avons alors également eu l'occasion d'apprendre à utiliser le debugger pour cibler exactement où se trouve l'erreur dans le programme.

Par ailleurs, nous avons dû mettre en place une méthode pour trouver la structure des programmes pour quelques fonctions afin de traiter certains exercices qui soulevaient certaines notions que nous n'avions pas encore vues ou du moins pas abordées en détail en cours, par exemple les templates class dans le fichier "*tableau-donnees-avance.cpp"*. En effet, contrairement à un langage de programmation comme Python où les erreurs sont explicitement affichées dans la console avec le numéro de lignes ainsi que le type d'erreur, ici, les messages d'erreurs affichés sur le serveur Jupyter n'ont pas toujours été évident à comprendre et parfois l'erreur n'a pas du tout été précisée ce qui a rendu la correction du programme plus compliquée. Cependant, les commandes abordées en cours ont été pratiques pour débugger comme par exemple la commande "*gdb --tui --silent nom*".

Ainsi, pour surmonter les difficultés rencontrées, nous avons été amené à découvrir et à revoir beaucoup de choses essentielles pour la suite de notre cursus en informatique, notamment la manipulation de fichiers et également d'approfondir certaines notions déjà vues en cours comme les tableaux en deux dimensions.

VII. Conclusion

Au final, le choix de ce sujet a été un moyen très efficace de découvrir une première façade de l'analyse de données en C++. Ainsi, ce projet a été un entrainement très pratique pour bien réviser et préparer au mieux notre examen final en programmation impérative et de surcroît indispensable pour enrichir nos compétences dans le langage C++.