# T.P. 1

# Premiers programmes, entrées/sorties, statistiques

## 1.1 Création et compilation d'un premier programme

### 1.1.1 Un premier programme

Considérons un programme très simple :

```
#include <iostream>
#include <cmath>

double aire_du_cercle(double x) {return M_PI*x*x;}

int main () {
        double r;
        std::cout << "Entrez le rayon du cercle:" << std::endl;
        std::cin >> r;
        std::cout << "Son aire est " << aire_du_cercle(r) << std::endl;
        return 0;
}
```

1. Créez un répertoire `TP1` sur votre bureau de travail.
2. Ouvrez un terminal et tapez `cd Desktop/TP1` ou `cd Bureau/TP1` (selon la langue du système) pour vous déplacer dans le répertoire créé. Laissez le terminal ouvert.
3. Ouvrez le logiciel Geany, créez un nouveau fichier, copiez le code ci-dessus et sauvegardez le fichier sous le nom `premierprog.cpp` dans le répertoire `TP1`. *Nous vous conseillons de le recopier à la main en essayant de comprendre chaque ligne.*
4. Revenez dans le terminal puis tapez

   ```
   g++ premierprog.cpp -o premierexec
   ```

   (selon la machine, il faut éventuellement ajouter `-lm` à la fin). Si un message apparaît, c'est qu'une erreur a été commise en recopiant le programme : corrigez-la.

5. Tapez à présent `./premierexec` dans le terminal. Il ne vous reste plus qu'à interagir avec votre programme !
6. Ajouter une fonction au programme précédent pour qu'il puisse calculer également l'aire d'un carré.
7. Modifier la fonction `main()` pour demander la longueur du côté d'un carré et afficher son aire.

### 1.1.2 Un deuxième programme

Considérons le code suivant :

```
#include __________
#include _________
#include ________

___ _____() {
        ___ n;
        _________ << "Entrez un nombre entier <100:" << std::endl;
        std::cin __ n;
        std::vector<int> tableau(n);
        for(__________________) {
                tableau[i]=i*i;
        }
        _____ofstream fichier("donnees.dat");
        fichier << "Voici les carrés des entiers:" << std::endl;
        for(__________________) {
                fichier << i <<": " <<tableau[i] << std::endl;
        }
        fichier._______;
        return 0;
}
```

1. Écrire ce programme dans un fichier `programme2.cpp` dans le répertoire `TP1`. Remplacer tous les `_____` par ce qu'il faut. Le compiler et l'exécuter. Que voyez-vous apparaître dans le répertoire `TP1` ?
2. Modifier ce programme pour avoir, dans le fichier `donnees.dat` sur les mêmes lignes, également les cubes des entiers.
3. Modifier le programme pour que les données soient écrites dans l'ordre décroissant dans le fichier.

## 1.2 Un brève présentation des entrées/sorties vers les fichiers

— L'écriture dans le terminal se fait avec `std::cout` et l'opérateur `<<`. Cet objet est défini dans la bibliothèque `iostream`.
— La lecture dans le terminal se fait avec `std::cin` et l'opérateur `>>`. Cet objet est défini dans la bibliothèque `iostream`.

— Un saut de ligne se fait avec l'opérateur `std::endl` .
— La déclaration d'un fichier en écriture se fait par

```
std::ofstream F ("Nom du fichier");
```

Cette commande est définie dans la bibliothèque `<fstream>` . On alimente le fichier en contenu avec l'opérateur d'injection `<<` selon la commande `F << CONTENU` où `CONTENU` est une valeur ou une variable.
— La déclaration d'un fichier en lecture se fait par

```
std::ifstream F ("Nom du fichier");
```

Cette commande est définie dans la bibliothèque `<fstream>` . On lit les données entre deux espaces avec `>>` selon `F >> x` où `x` est la variable de stockage des données lues.
— Avant la fin du programme, tout fichier doit être fermé avec `F.close();` .
— Il est possible de se débarrasser de tous les préfixes `std::` en écrivant :

```
using namespace std;
```

juste après les `include` .
— Toute la documentation des classes de la STL (vecteurs, listes, etc) est disponible sur http://www.cplusplus.com/reference/stl/ et sur http://www.cplusplus.com/reference/std/ pour celle sur les algorithmes, l'aléatoire, les complexes, etc.

## 1.3 Quelques calculs statistiques simples

### 1.3.1 Sans la bibliothèque `<algorithm>`

L'archive de données `fichiersTP.zip` contient un fichier `smalldata.txt` . Récupérez-le et placez-le dans votre répertoire de travail pour ce TP.

Ce fichier contient 2500 personnes, décrites par leur prénom, leur ville, leur âge et leur temps de course à l'épreuve du 100 mètres. Pour décrire une personne, nous introduisons la structure suivante :

```
struct Fiche {
        std::string prenom;
        std::string ville;
        int age;
        double temps;
};
```

Le type `std::string` permet de stocker des chaînes de caractères et est défini dans la bibliothèque `<string>` .

1. Créez dans `TP1` un programme `analyse.cpp` qui contient les bibliothèques nécessaires, la définition de la structure ci-dessus, une fonction `main()` qui ouvre le fichier `smalldata.txt` en lecture.
2. Déclarez dans ce programme un tableau `vdata` de taille 2500 et contenant des objets de type `Fiche`. Remplir ce tableau avec les données du fichier.
3. En utilisant uniquement des boucles `for`, des tests logiques `if` et en déclarant des variables, écrivez un programme (ou des programmes si vous préférez faire le travail en plusieurs fois) qui répond aux questions suivantes :
   (a) Combien de personnes habitent Lyon ? Quelle est le pourcentage de Lyonnais ?
   (b) Combien de personnes habitent Lyon et ont strictement moins de 30 ans ?
   (c) Existe-t-il un Toulousain dont le prénom commence par la lettre *A* ?
   (d) Quel est l'âge minimal ? L'âge maximal ? Comment s'appelle le plus jeune ? Le plus âgé ?
   (e) Quel est l'âge moyen des personnes du fichier ? Quel est l'écart-type de leur âge ?
   (f) Les Parisiens sont-ils en moyenne plus rapides au 100 mètres que les Marseillais ?
   (g) Produire un fichier `toulousains.txt` qui contient toutes les informations sur les personnes qui habitent Toulouse. On remplacera dans ce fichier leur âge par leur date de naissance (on supposera que les âges ont été déclarés en 2018).
   (h) Quelle est la covariance empirique entre âge et temps à l'épreuve du 100 mètres sur cet échantillon de Toulousains ?
   (i) Afficher dans le terminal la liste des villes représentées. *Attention, ce n'est pas si facile ! Vous pouvez utiliser si vous le souhaitez le conteneur* `std::set` *pour avoir une solution rapide ou sinon tout refaire à la main.*
4. (bonus) Supposons à présent que nous n'ayons pas donné initialement le nombre de personnes du fichier : cela empêcherait la déclaration du tableau statique `individu` à la bonne taille. Réécrire le début du programme en utilisant à présent un tableau `individu` de type `std::vector<Fiche>` de la classe `<vector>`. *Indication : le remplir avec* `push_back()` *en parcourant le fichier.*

### 1.3.2 Avec la bibliothèque `<algorithm>`

1. Refaire intégralement toutes les questions précédentes 3.(a) jusqu'à 3.(i) **sans écrire une seule boucle** `for` et en utilisant intensivement la bibliothèque standard `<algorithm>` dont une documentation est disponible sur le site suivant :

   http://www.cplusplus.com/reference/algorithm/

   Vous vous inspirerez des exemples décrits sur cette page pour chaque fonction. Pour certaines questions, vous pourrez également utiliser la fonction `std::accumulate` de la bibliothèque `<numeric>`.

   La plupart des fonctions de `<algorithm>` prennent en argument une fonction de test ou de comparaison : vous pourrez, au choix, soit déclarer ces fonctions dans le préambule de votre programme, soit dans le corps de la fonction `main()` en utilisant des lambda-fonctions du standard C++11.

   **Exemple :** pour la question (3)-a, il suffit d'écrire :

```
auto is_from_Lyon=[](Fiche f) {
                return (f.ville=="Lyon");}
int nb_Lyon=std::count_if(vdata.begin(),vdata.end(),is_from_Lyon);
std::cout << "Il y a " << nb_Lyon << " Lyonnais.\n";
```

2. Dans `<algorithm>` , il existe une fonction de tri `std::sort` qui fonctionne de la manière suivante. Si `v` est un vecteur d'objets de type `T` et `compare` une fonction de prototype :

```
bool compare(T x, T y)
```

qui renvoie `true` si `y` est plus grand que `x` et `false` sinon, alors l'instruction

```
std::sort(v.begin(),v.end(),compare)
        // pour v de type std::vector ou std::list
```

trie le tableau par ordre croissant. Produire un fichier `data_tri.txt` qui contient les 100 personnes les plus rapides au 100 mètres triées par vitesse décroissante.

### 1.3.3 Quelques questions additionnelles plus difficiles pour plus de réflexion

Vous pourrez traiter ces questions à la fois en écrivant vous même les boucles nécessaires et en définissant les variables nécessaires au calcul et à la fois en vous appuyant sur les outils de la bibliothèque standard.

1. Quel est le plus petit écart entre les temps de courses au 100 mètres de deux personnes (indice en note de bas de page [1]) ?
2. Créer deux vecteurs `jeunes` et `moinsjeunes` contenant respectivement les fiches des personnes de moins de 40 ans et de strictement plus de 40 ans (indice en bas de page [2]).
3. Écrire dans un fichier `ordre.dat` la liste des 2500 personnes classées selon l'ordre suivant :
   — par ordre alphabétique des prénoms
   — en cas d'égalité, par ordre alphabétique des villes
   — en cas d'égalité à nouveau, de la plus âgée à la plus jeune
   — en cas d'égalité à nouveau, de la plus lente à la plus rapide.
   et, bien sûr, vérifier que le fichier produit est correct.
4. Nous souhaitons établir l'histogramme des âges qui permette de connaître la répartition des âges. En utilisant le conteneur `std::map<int,int>` , calculer l'histogramme et l'afficher ligne par ligne dans le terminal. Quelle est la classe d'âge la plus nombreuse ?

1. Vous pouvez utiliser `std::sort` , `std::adjacent_difference` et rechercher un extremum. Nous vous conseillons tout d'abord d'extraire les temps de courses dans un `std::vector<double>` avant d'appliquer `std::adjacent_difference` à des `double` et non des `Fiche` .

2. Vous pourrez utiliser `std::partition_copy` et les itérateurs `std::back_inserter` de `<iterator>` si vous souhaitez utiliser `<algorithm>` pleinement et ne pas avoir à calculer au préalable le nombre de personnes de chaque catégorie.