Nom : Ilyass RAMDANI
3A Génie Industriel

[C++] PRALG : Rapport sur le TP Ensembles, tableaux associatifs et Hachage, À l'attention de monsieur Pascal Monasse

Dans ce TP, toutes les fonctions que nous ajoutons au script ont été ajouté dans un nouveau fichier Tools.h et Tools.cpp. Nous aurions pu tout ajouter dans les fichiers town.h et town.cpp. Seulement nous trouvons cela plus clair d'ajouter les opérations qui s'articulent autour de l'étude du fichier des villes dans un nouveau fichier que dans celui qui les défini.

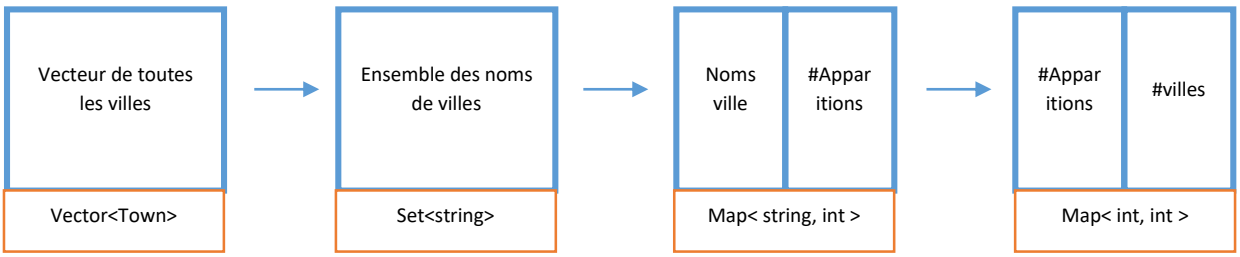
Seulement la définition des opérateurs < pour town et == pour Point2D seront formellement dans leur propre fichier car cela nous paraissait plus logique.

Je ne dis pas que c'est la meilleure manière de procéder, c'était seulement celle avec laquelle j'étais le plus à l'aise. ☺

Question 1

Notre idée pour obtenir les informations nécessaires au tracé de l'histogramme est la suivante :

- Créer un ensemble de nom de villes
- Parcourir l'ensemble des noms de villes et y associer leur nombre d'apparition
- Parcourir le nombre d'apparition et y associer le nombre de villes pour tracer l'histogramme



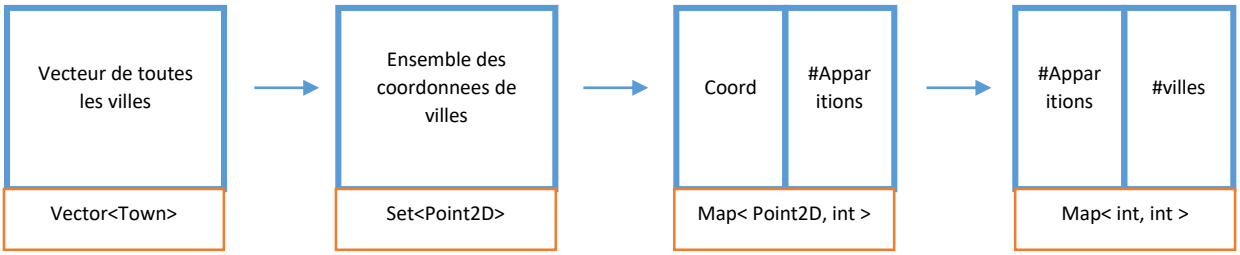
Dans nos petites cases # signifie nombre.

Nous choisirons donc d'utiliser unordered_map car Map (tout comme set) est une séquence ordonnée de clés uniques, alors que dans unordered_map, les clés peuvent être stockées dans n'importe quel ordre, donc sans ordre.

La complexité temporelle des opérations de la carte est de O(Log n) alors que pour unordered_map, elle est de O(1) en moyenne.

Question 2

Pour la question 2, nous procédons d la même manière qu'a la question 1 seulement là avec les coordonnées. Ici nous utiliserons map.



Question 3

Déterminer N

Pour déterminer l'ensemble N des villes qui ont une autre ville de meme nom, nous parcourons l'ensemble des villes et pour toutes celles où leur nom apparait 2 fois ou plus dans notre map, nous l'ajoutons à notre ensemble N.

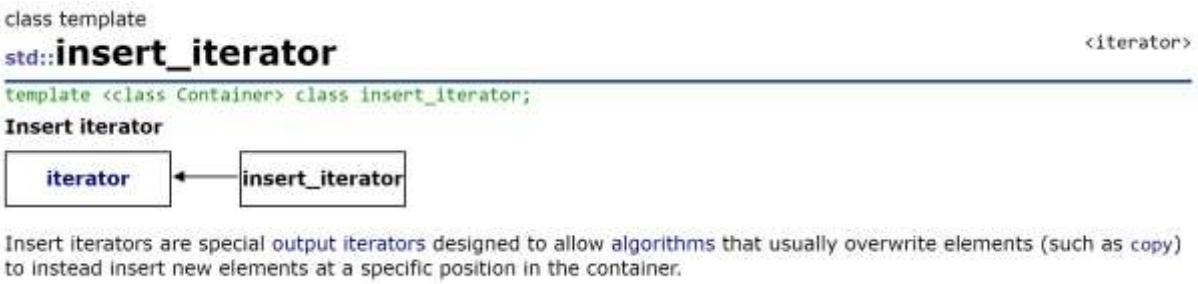
Déterminer C

Pour déterminer l'ensemble C, nous procédons de la même manière.

Determiner $N \cap C$

Enfin, pour déterminer l'ensemble des villes qui sont dans l'intersection de N et C nous avons dû parcourir la documentation, pour trouver la fonction qui nous sera utile.

Nous nous sommes donc naturellement rendu sur le site cplusplus.com où nous avons trouvé insert_iterator qui répond exactement à nos attentes.

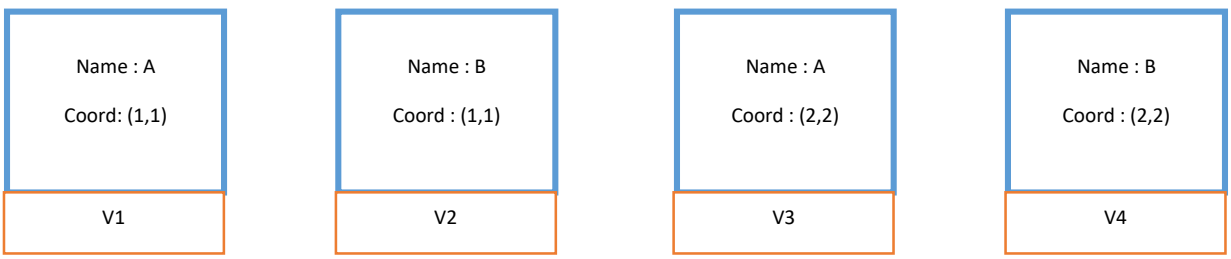


Nous avons donc lu la page pour comprendre aussi comment l'employer.

Questions 4

Pour cette question nous avons décidé de suivre les étapes suivantes :

1. Parcourir $N \cap C$: Notons l'element de l'intersection v1 pour un soucis de comprehension de notre schéma
2. Enumerer les voisins de v1 : ensemble des elements v2
3. Enumerer les homonymes de v1 : ensemble des elements v3
4. Sur les voisins de l'homonyme de v1 ajouter 1 dès lors qu'on est face à un homonyme de v2 : ensemble des elements v4



Question 5

Avec une approche naïve, nous aurions effectué $35\ 180^4$ cas ≈ 10^{18} . Même si les premiers test d'égalité comme la recherche de voisins élaguent.

En effet, il y a 31665 noms de ville qui sont utilisé par 1 seule ville, et 33030 Coordonnées de ville qui ne sont utilisées par une seule ville.

Cela nous fait tout de même :

$$35\ 180\ \times (35\ 180 - 33\ 030) \times (35\ 180\ -\ 31\ 665) \times 35\ 180 =$$

$$=\ 35\ 180\ \times 2\ 150 \times 3515 \times 35\ 180 =\ 9 \times 10^{15}$$ cas ≈ 10^{16}

En utilisant l'intersection de N et C, nous réduisons drastiquement la complexité du problème.

En effet, l'ensemble des villes qui sont dans l'intersection de N et C étant au nombre de 207, cela nous fait alors 207^4 cas ≈ 10^9.