Nicolas Doens Polytech Nice Sophia  
Rubens PETROVICH ELEC4\_GSE

Cours de C++

# TP introduction

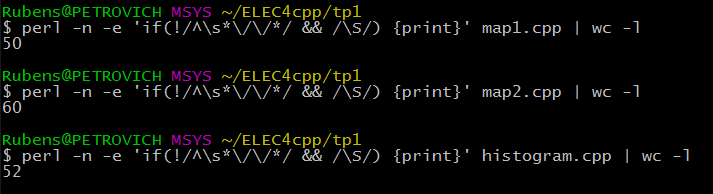
## Introduction

Ce TP est une découverte du langage C par l’amélioration et la création de programme. Nous commencerons par modifier un fichier qui calcule la moyenne et la médiane d’une suite de nombres en ajoutant leurs histogrammes. Ensuite nous créerons deux nouveaux programmes map1 et map2 qui nous permettront de retrouver une valeur dans un fichier à l’aide de son identifiant(unique) et inversement au pourcent près pour map2.

## Programmes

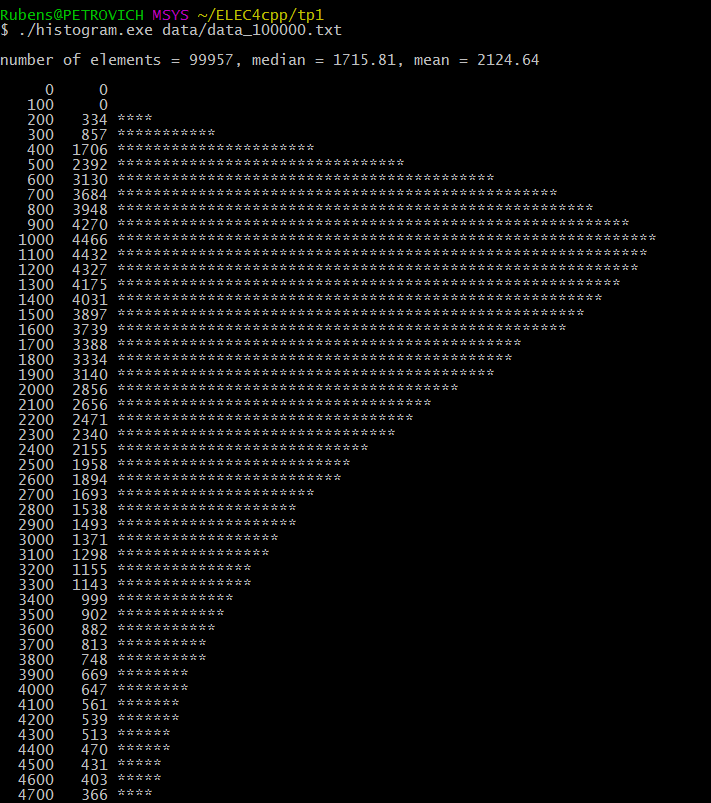
Nous avons écrit les 3 codes sources : histogram.cpp, map1.cpp et map2.cpp

Nous calculons les lignes :

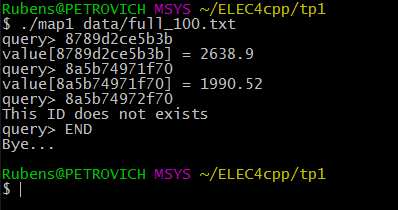


Nous pouvons voir que nous avons atteint le quota de lignes ; 50 pour map1 et 60<50 pour map2

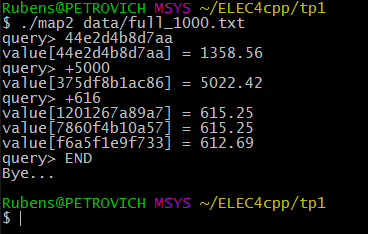
## Histogramme :



## Map1 :



## Map2 :



## Les « containers »

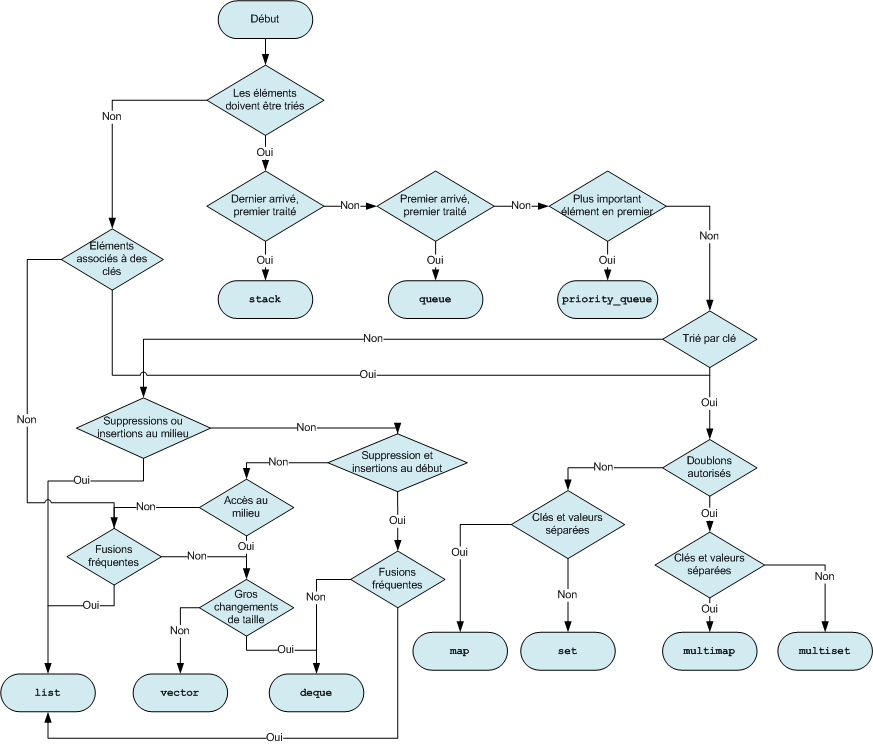
Il existe en C++ de nombreux container tel que les list, les vecteurs les map…etc

Pour choisir le nôtre, nous avons listé nos besoin et contraire :

Nous avons besoin d’un container dont les éléments sont associés à des clés, dont les doublons ne sont pas autorisés avec les couple clés valeur séparé.

Le container qui nous a permis de répondre à chaque critère est le map. En effet, par exemple une list ou un vecteur n’aurai pas pu répondre à tous ces critères.

Nous nous sommes aussi aidé d’un organigramme ([www.openclassroom.com](http://www.openclassroom.com)) :



## Complexité d’une notre « query »

La complexité de la query n’est pas constante, elle dépend du nombre de valeurs n :

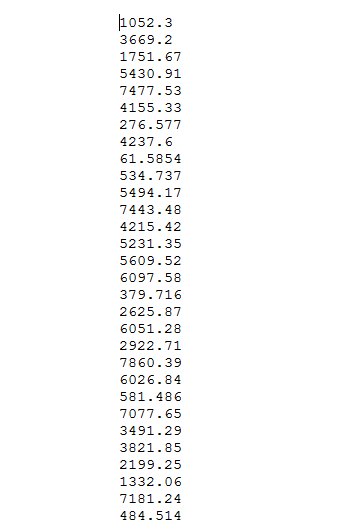
On part du principe que nous trouvons la valeur au milieu du tableau, si on multiplie le nombre de valeurs du fichier par n, alors la query dure n fois plus longtemps lors de l’exécution.

De même pour la recherche d’ID par les valeurs, nous parcourons le tableau à chaque exécution du programme. Donc si on multiplie par n le nombre de lignes, on multipliera aussi par n la complexité.

A l’aide de la librairie windows.h, nous avons calculé les temps d’exécution des programmes. :

Pour 10'000 couple ID/Valeur, nous obtenons un temps d’environ 0.001s et environs 0.009s pour un nombre de couple ID/Valeur de 100'000. Cela nous montre que la complexité est bien linéaire. Malheureusement nous n’avons pas pu effectuer plus de test du fait de la rapidité du programme (nous ne pouvons as affiché sous 0.001s).

## Optionnel

Fonctionnement : Le programme optionnel.cpp Exemple de fichier généré :  
génère un fichier avec n données dont l’histogramme est similaire à celui des fichiers data\_100000.txt. Pour cela l’utilisateur doit donner 2 arguments à l’appel de la fonction :

* Le premier argument correspond on nombre de valeurs du fichier.
* On utilise une distribution normale, le 2ieme argument est la moyenne de cette distribution.
* Le 3ieme argument correspond au sigma.
* Le 4ieme désigne le nom du fichier. Si le fichier n’existe pas il est créé, s’il existe sont contenu est effacé avant d’y écrire les valeurs.

