Nicolas Doens
Rubens PETROVICH

Polytech Nice Sophia
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 :

```
Rubens@PETROVICH MSYS ~/ELEC4cpp/tp1
$ perl -n -e 'if(!/^\s*\/\/*/ && /\S/) {print}' map1.cpp | wc -l
50

Rubens@PETROVICH MSYS ~/ELEC4cpp/tp1
$ perl -n -e 'if(!/^\s*\/\/*/ && /\S/) {print}' map2.cpp | wc -l
60

Rubens@PETROVICH MSYS ~/ELEC4cpp/tp1
$ perl -n -e 'if(!/^\s*\/\/*/ && /\S/) {print}' histogram.cpp | wc -l
52
```

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

Histogramme :

```
Rubens@PETROVICH MSYS ~/ELEC4cpp/tp1
$ ./histogram.exe data/data_100000.txt

number of elements = 99957, median = 1715.81, mean = 2124.64

     0     0
   100     0
   200   334 ****
   300   857 ***********
   400  1706 ***********************
   500  2392 ********************************
   600  3130 *****************************************
   700  3684 ************************************************
   800  3948 *****************************************************
   900  4270 ********************************************************
  1000  4466 ***********************************************************
  1100  4432 ***********************************************************
  1200  4327 *********************************************************
  1300  4175 *******************************************************
  1400  4031 *****************************************************
  1500  3897 ***************************************************
  1600  3739 *************************************************
  1700  3388 *********************************************
  1800  3334 *********************************************
  1900  3140 *******************************************
  2000  2856 ***************************************
  2100  2656 ***********************************
  2200  2471 *********************************
  2300  2340 *******************************
  2400  2155 *****************************
  2500  1958 **************************
  2600  1894 *************************
  2700  1693 **********************
  2800  1538 ********************
  2900  1493 ********************
  3000  1371 ******************
  3100  1298 *****************
  3200  1155 ***************
  3300  1143 ***************
  3400   999 *************
  3500   902 ************
  3600   882 ***********
  3700   813 **********
  3800   748 **********
  3900   669 ********
  4000   647 ********
  4100   561 *******
  4200   539 *******
  4300   513 ******
  4400   470 ******
  4500   431 *****
  4600   403 *****
  4700   366 ****
```

Map1 :

```
Rubens@PETROVICH MSYS ~/ELEC4cpp/tp1
$ ./map1 data/full_100.txt
query> 8789d2ce5b3b
value[8789d2ce5b3b] = 2638.9
query> 8a5b74971f70
value[8a5b74971f70] = 1990.52
query> 8a5b74972f70
This ID does not exists
query> END
Bye...

Rubens@PETROVICH MSYS ~/ELEC4cpp/tp1
$
```

Map2 :

```
Rubens@PETROVICH MSYS ~/ELEC4cpp/tp1
$ ./map2 data/full_1000.txt
query> 44e2d4b8d7aa
value[44e2d4b8d7aa] = 1358.56
query> +5000
value[375df8b1ac86] = 5022.42
query> +616
value[1201267a89a7] = 615.25
value[7860f4b10a57] = 615.25
value[f6a5f1e9f733] = 612.69
query> END
Bye...

Rubens@PETROVICH MSYS ~/ELEC4cpp/tp1
$
```

Les « containers »

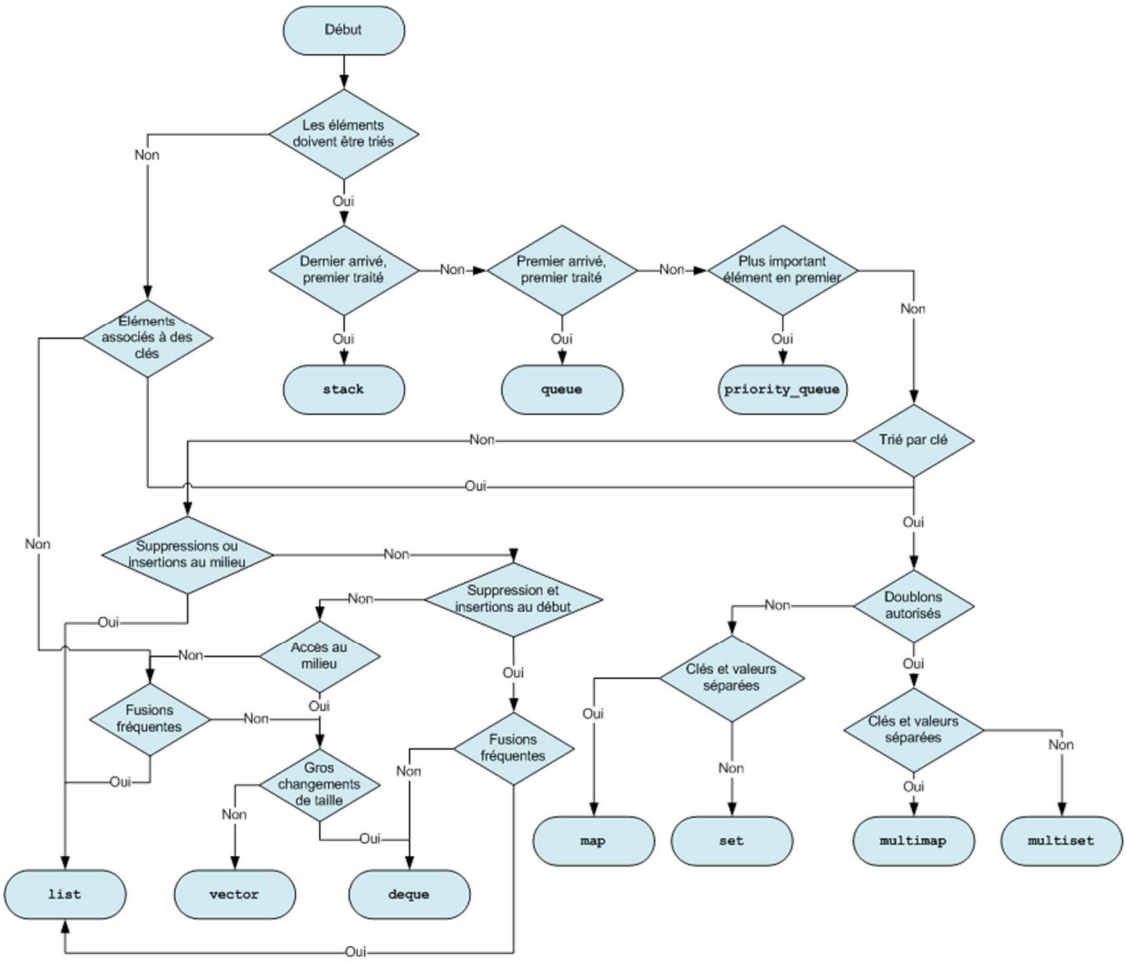
Il existe en C++ de nombreux container tel que les list, les vecteurs les map…etc

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) :



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 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.

```
Rubens@PETROVICH MSYS ~/ELEC4cpp/tp1
$ ./optionnel.exe 100000 2000 200 test.txt

Rubens@PETROVICH MSYS ~/ELEC4cpp/tp1
$ ./optionnel.exe 10 2000 200 test-10.txt

Rubens@PETROVICH MSYS ~/ELEC4cpp/tp1
$ ./optionnel.exe 100 6000 500 test-100.txt

Rubens@PETROVICH MSYS ~/ELEC4cpp/tp1
$ ./optionnel.exe 1000 2000 100 test-1000.txt
```

Exemple de fichier généré :

```
1052.3
3669.2
1751.67
5430.91
7477.53
4155.33
276.577
4237.6
61.5854
534.737
5494.17
7443.48
4215.42
5231.35
5609.52
6097.58
379.716
2625.87
6051.28
2922.71
7860.39
6026.84
581.486
7077.65
3491.29
3821.85
2199.25
1332.06
7181.24
484.514
```