**Projet Système**

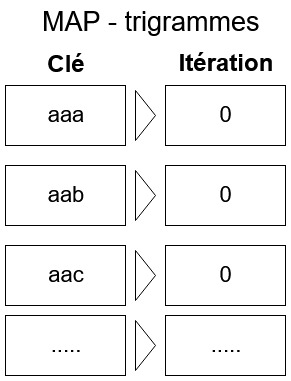
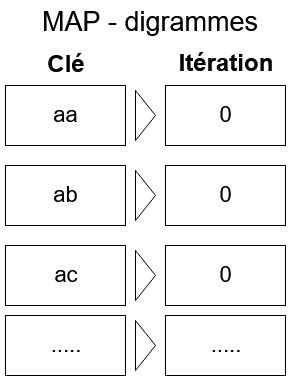
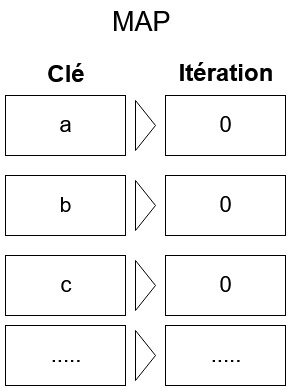
**Sujet**: Fréquences dans des dictionnaires  
 À partir de plusieurs dictionnaires utilisant l'alphabet latin, déterminer pour chacun les fréquences des lettres "a, b, c, ..., z", des digrammes "aa, ab, ac, ..., ba, bb, bc, ..., zz", et la même chose pour les trigrammes "aaa, aab, aac, ... aba, abc, .... baa, bab, ...., zzz". Les résultats sont stockés dans un fichier unique.

Documentation utilisé :  
 [http://www.cplusplus.com](http://www.cplusplus.com/)  
 https://en.cppreference.com

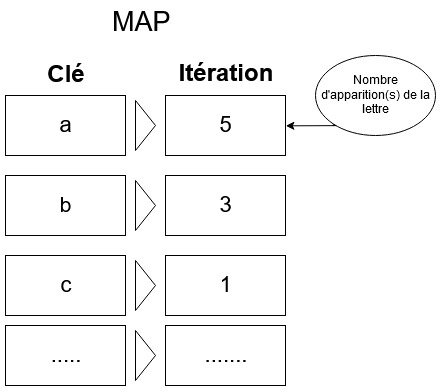
**Explication des algorithmes**

**Algorithme 1  
  
FreqAnalysis-sequentiel.cpp déroulement du programme :**

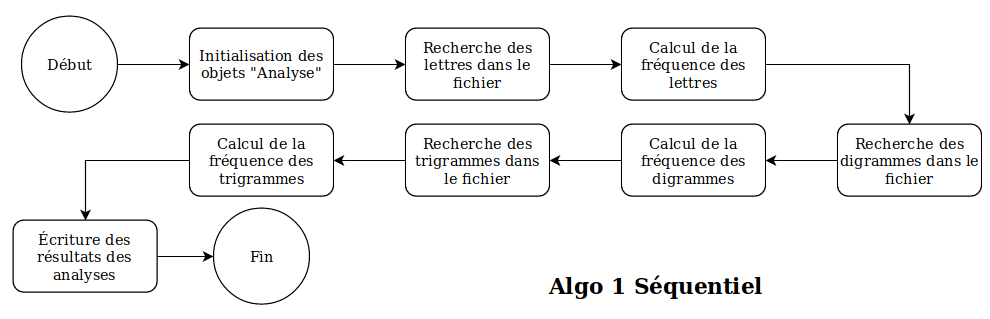
Initialisation :  
 Pour analyser la fréquence d’apparition des lettres un utilise une map contenant un string pour la lettre et un float pour la fréquence d’apparition de celle-ci. On initialise d’abord la map avec toutes les lettres de l’alphabet et leurs fréquences à 0. On fait de même pour les digrammes et les trigrammes : on initialise avec toutes les possibilités à 0.

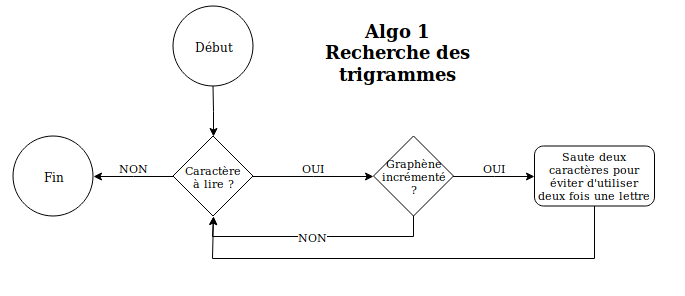
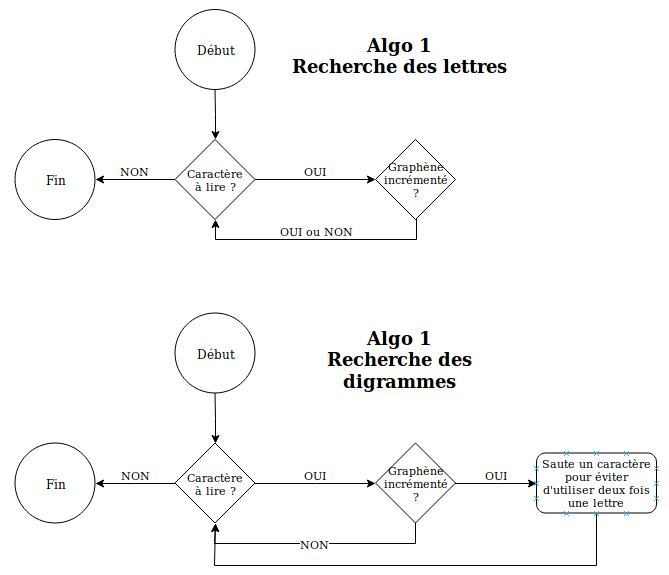


Analyse :  
 Pour chaque objet Analyse on ouvre le fichier concerné. Puis on lit caractère par caractère tant qu’il y en a. A chaques graphène trouver on incrementre son nombre d’apparition à l’aide de la fonction incGraphene(). Cette fonction de la classe Analyse va incrémenter la partie itération de la map correspondant au graphène lu. Et on incrémente le nombre de graphènes total trouvés. Donc on a d’abord l’anlyse des lettres, puis des digrammes et enfin des trigrammes. Ceux qui veut dire trois lectures du fichier



Calcul et écriture des fréquences :  
 Via la méthode calcFreq() de la classe Analyse on calcule la fréquence de chaque lettres en divisant le nombre d’apparition (contenu dans la map) par le nombre de lettre total (donnée membre de la classe). Et on ecrase l’itération de la lettre par sa fréquence dans la map. Et on fait de même pour les digrammes et trigrammes.

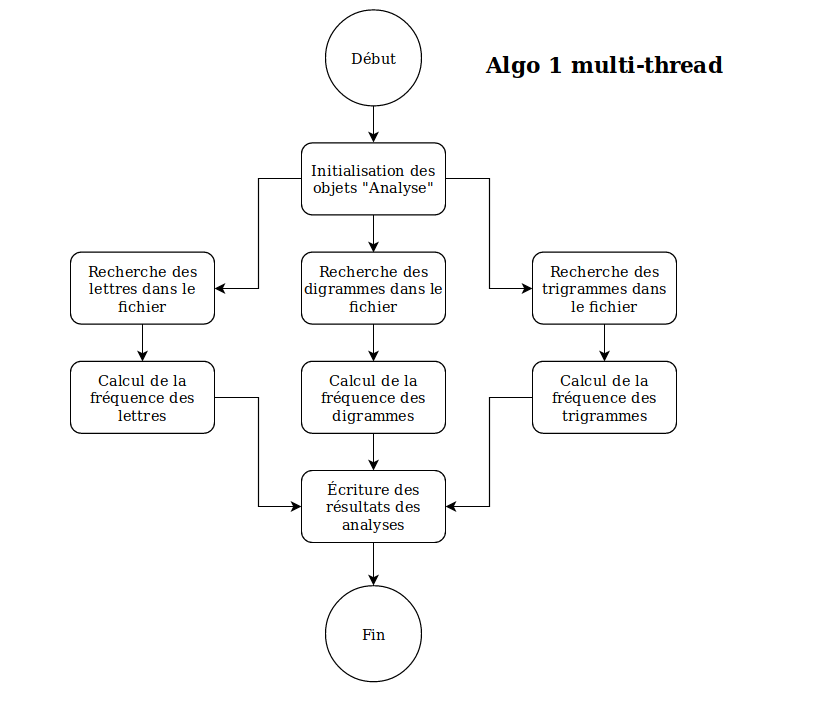
On écrit dans un fichier de sortie le résultat du calcul par la fonction printAnalyse().  
  
Diagramme du fonctionnement simplifié :

Fonction de recherche :

**FreqAnalysis-pthread.cpp déroulement du programme :**

Initialisation :  
Même chose que **FreqAnalysis-sequentiel.cpp**

Analyse :  
On utilise trois threads, un pour la fréquence des lettres, un pour la fréquence des digrammes, un pour la fréquence des trigrammes.  
Les trois threads sont lancés en parallèle

Calcul et écriture des fréquences :  
 La méthode calcFreq() de la classe Analyse on calcule la fréquence de chaque graphène en divisant le nombre d’apparition (contenu dans la map) par le nombre de graphènes total (donnée membre de la classe). Et on ecrase l’itération du graphène par sa fréquence dans la map.  
 Lorsque le premier thread (analyse des lettres) se termine on lance la fonction calcFreq() qui lance le calcul de fréquence de chaque lettres. Lorsque le calcul est fini on écrit les résultat dans le fichier de sortit pour pas perdre de temps. De même lorsque l’analyse des digrammes et de trigrammes finissent.  
 Pour écrire on utilise la fonction printAnalyse().  
  
Diagramme du fonctionnement simplifié :fonction de recherche :  
 Identique au programme séquentiel

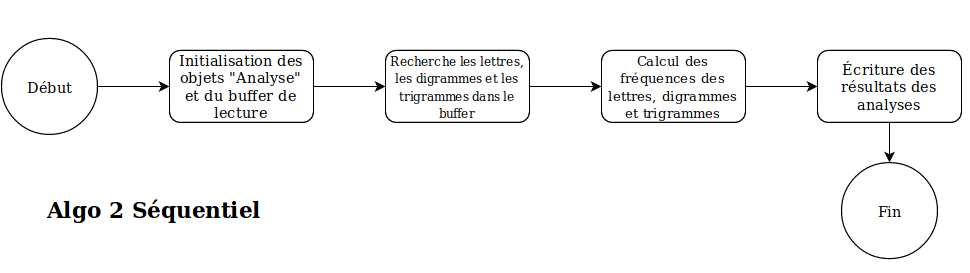
**Algorithme 2**

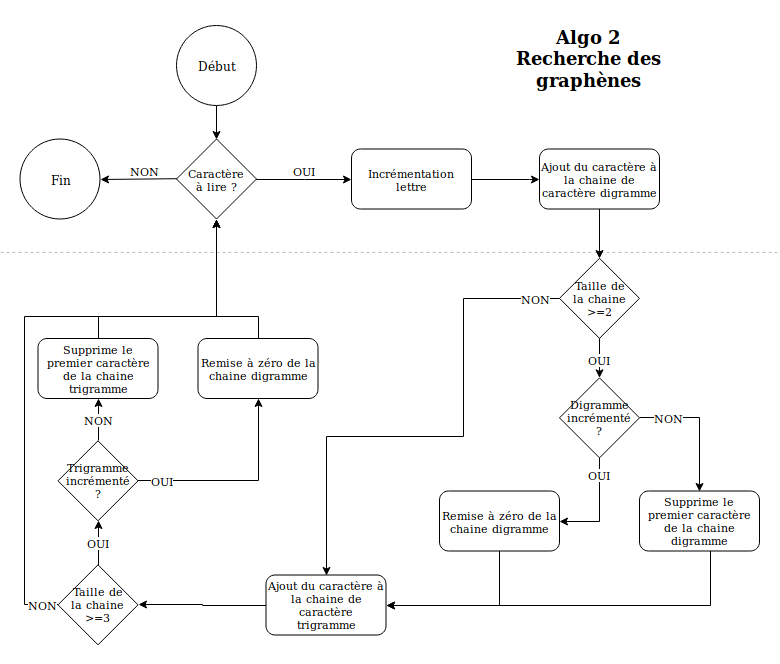
**FreqAnalysis-sequentiel.cpp déroulement du programme :**

Initialisation :  
 Pour analyser la fréquence d’apparition des lettres un utilise une map contenant un string pour la lettre et un float pour la fréquence d’apparition de celle-ci. On initialise d’abord la map avec toutes les lettres de l’alphabet et leurs fréquences à 0. On fait de même pour les digrammes et les trigrammes : on initialise avec toutes les possibilités à 0. Pour le fichier on l’ouvre et on récupère le buffer.

Analyse :  
 Pour faire l’analyse on récupère le buffer et on lit caractère par caractère tant qu’il y en a. Envoie du caractère dans la fonction incGraphène de l’analyse des lettres. Ajout du caractère dans une chaine de caractères temporaire du digrammes et du trigrammes. Si la chaine de caractères digramme contient deux caractères alors on l’envoie dans la fonction incGraphène de l’analyse de digrammes. De même avec les trigrammes pour 3 caractères.   
La fonction incGraphène renvoie vrai s’il a reussi a incrémenter un graphène (ce qui veux dire que l’on veux la fréquence de ce graphène) , faux sinon.  
Si incGraphène de l’analyse des digrammes renvoie faux alors on supprime le premier caractère de la chaine de caractère de digrammes. De même pour les trigrammes.

Calcul et écriture des fréquences :  
 La méthode calcFreq() de la classe Analyse on calcule la fréquence de chaque graphène en divisant le nombre d’apparition (contenu dans la map) par le nombre de graphènes total (donnée membre de la classe). Et on ecrase l’itération du graphène par sa fréquence dans la map.  
 Donc lorsque l’analyse est terminer on lance le calcul des fréquences de toutes les analyses et on les écrit dans le fichier de sortie.

Diagramme du fonctionnement simplifié :

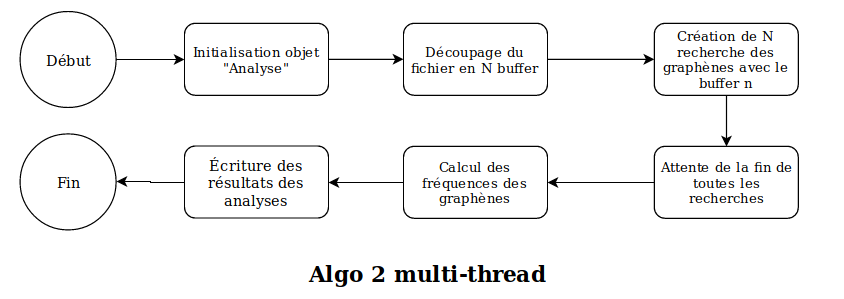
Fonction de recherche :

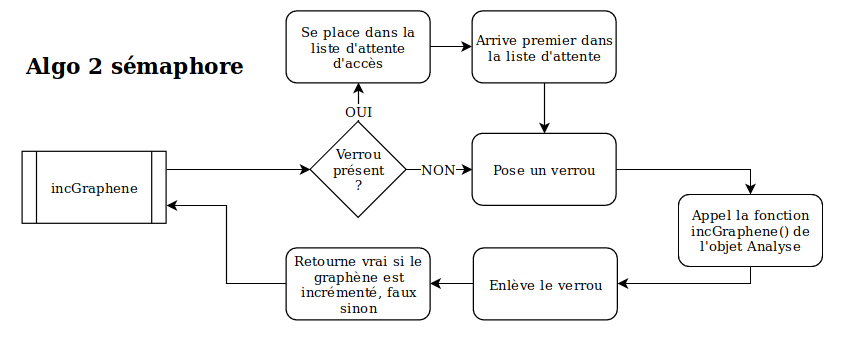
**FreqAnalysis-pthread.cpp déroulement du programme :**

Initialisation :  
 Pour analyser la fréquence d’apparition des lettres un utilise une map contenant un string pour la lettre et un float pour la fréquence d’apparition de celle-ci. On initialise d’abord la map avec toutes les lettres de l’alphabet et leurs fréquences à 0. On fait de même pour les digrammes et les trigrammes : on initialise avec toutes les possibilités à 0. Pour le fichier on l’ouvre et on récupère le buffer. Ce buffer on va ensuite le divisé par le nombre de thread N (4 dans le projet). On aura donc N buffers avec chacun une partie du fichier de base.

Analyse :  
 On utilise N threads, avec chacun un bout du fichier de base sous forme de buffer. Sachant que chaque thread partage les analyse il nous faut un sémaphore pour bloquer l’accès en ecriture pour l’incrémentation des graphène. Pour chaque thread le fonctionnement d’analyse reste le même que l’algo2 du programme séquentiel.

Calcul et écriture des fréquences : Lorsque tous les thread sont terminés on calcul les fréquences de toutes les analyses. Puis on écrit les résultats dans le fichiers de sortie.

Diagramme du fonctionnement simplifié :

Fonction de recherche :  
 C’est la même chose que le séquentiel à l’exception que chaque incrémentation d’un graphène, un verrou est posé sur l’objet « Analyse » concerné.

**Test des programmes**

Machine de test :

Machine 1 :  
- Type machine :Laptop   
- Système d’exploitation : Ubuntu 19.04  
- Processeur : Intel(R) Core(TM) i7-8750H  
- Fréquence processeur : Base : 2.20Ghz ; Turbo : 4.10 GHz   
- Nombre Cœurs : 6  
- Nombre de Threads : 12  
- Cache L1/L2/L3 : 384KiB / 1536KiB / 9MiB  
- RAM : 8034292 kB  
  
Machine 2 :

- Type machine :Laptop   
- Système d’exploitation : Ubuntu 19.04  
- Processeur : Intel(R) Core(TM) i7-8550U  
- Fréquence processeur : Base : 1.80Ghz ; Turbo : 4.00 GHz   
- Nombre Cœurs : 4  
- Nombre de Threads : 8  
- Cache L1/L2/L3 : 256KiB / 1MiB / 8MiB  
- RAM : 8048780 kB