Rapport du projet

Projet C-Wildwater

Filière Préing2 2024-2025

Groupe MI2-F :
Syrine Chellat
Aranney Sivakumar

Présentation de l’équipe :

Nous sommes une équipe de deux étudiants en deuxième année du cycle préparatoire intégré à CY Tech.

Présentation du projet :

Le projet WildWater consiste à analyser un réseau de distribution d’eau à partir d’un fichier de données, afin de calculer les volumes d’eau traités, captés et maximaux par usines, d’analyser les pertes d’eau, puis de générer des fichiers exploitables pour produire des histogrammes et pour garder un historique des calculs.

Lors de notre projet :

Nous avons commencé par discuter ensemble des tâches à réaliser et de la structure générale du programme, notamment le choix des structures de données (arbres AVL, structures dynamiques) et l’organisation des fonctions.

Nous avons ensuite décidé de développer les fonctions principales individuellement, chacun étant responsable d’une partie du projet. Le travail sur le main, le script Shell, le makefile et la lecture du fichier de données a en revanche été réalisé ensemble.

Chaque partie a été compilée et testée séparément. Cependant, la mise en commun du code a généré de nombreuses erreurs, parfois difficiles à détecter. Certaines fonctions fonctionnaient correctement seules, mais entraient en conflit une fois intégrées au programme global. Cela nous a demandé beaucoup de temps de débogage et de tests pour comprendre l’origine réelle des problèmes et assurer la cohérence entre toutes les parties du projet.

L’un des principaux problèmes rencontrés concernait la fonction de détection du type des lignes du fichier de données. Dans une première version, certaines lignes correspondant à des usines étaient détectées comme INCONNU, tandis que certaines lignes de type *USINE → STOCKAGE* étaient mal classées. Ces erreurs se propageaient ensuite dans le reste du programme et provoquaient des incohérences dans les résultats, notamment dans les histogrammes (volumes à zéro ou entités non souhaitées affichées).

Nous avons compris que cette fonction était centrale et que son mauvais fonctionnement faussait les calculs réalisés plus loin. Nous avons donc repris l’analyse du fichier de données et de la consigne, puis redéfini précisément les conditions nécessaires pour identifier correctement chaque type de ligne. Une seconde version plus fiable de la fonction a alors été implémentée, ce qui a permis de corriger les incohérences observées.

Nous avons également rencontré quelques difficultés lors de la connexion entre le programme C, le script Shell et gnuplot, notamment dans la gestion des arguments et des fichiers générés. Après plusieurs essais et ajustements, cette partie a été correctement intégrée.

De plus, ce projet nous a permis de mieux comprendre l'intérêt des arbres AVL lorsqu'un grand volume de données doit être traité efficacement. Il nous a également permis de travailler sur un projet concret regroupant plusieurs notions importantes : lecture et écriture de fichiers texte, structures dynamiques, récursivité, automatisation avec un script Shell et génération de graphiques avec gnuplot. Nous avons aussi découvert de nouveaux outils, comme le gnuplot, que nous n'avions pas utilisés auparavant.

Exemples d'exécution et résultats obtenus :

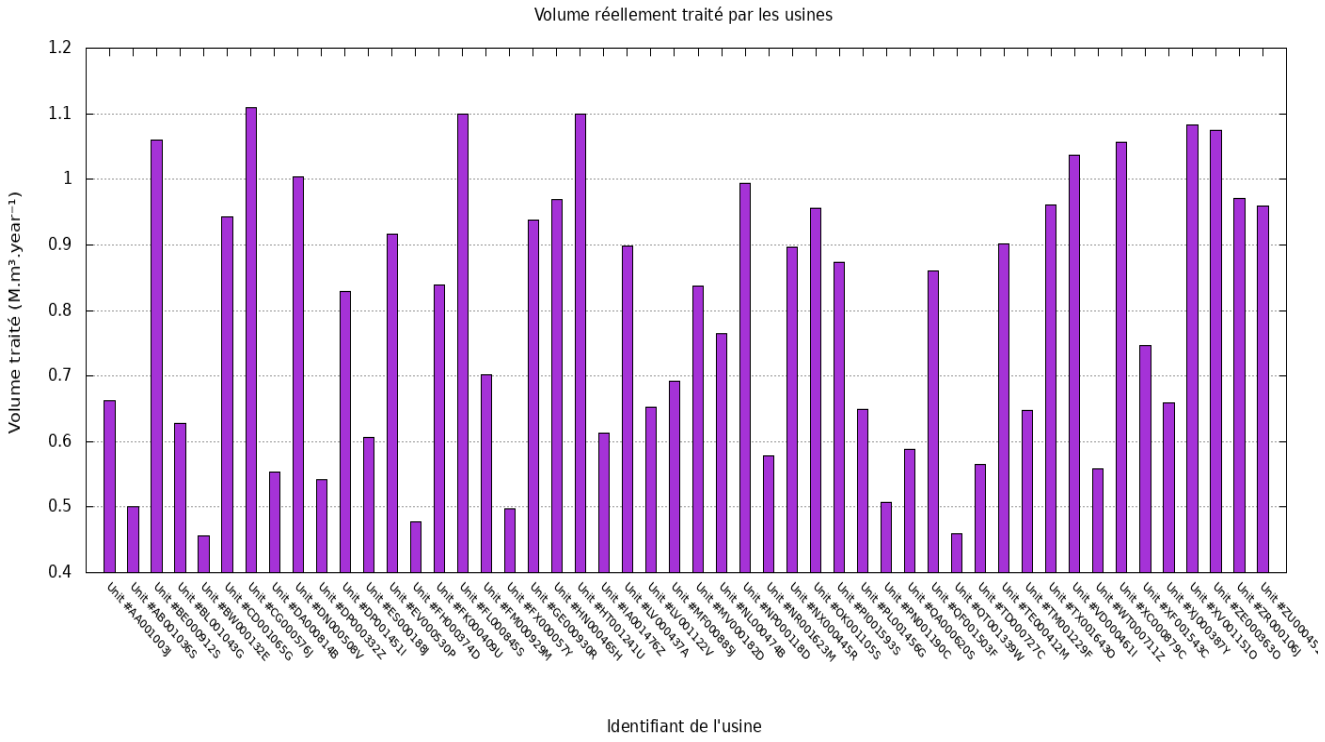
Nous avons regroupé les fichiers générés lors de l'exécution du programme dans un dossier 'tests', contenant à la fois des fichiers de données et des images.

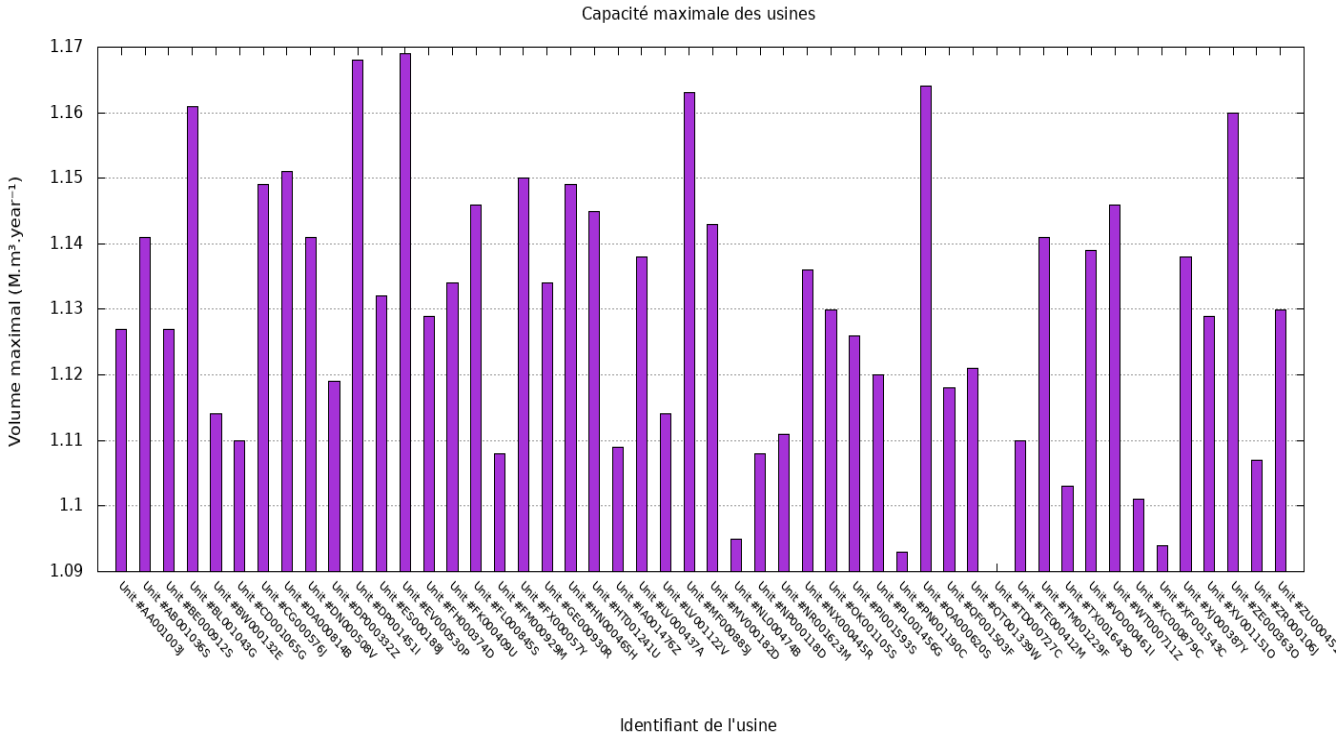
-Après exécution du programme avec la commande histo, les fichiers suivants (contenant une première ligne décrivant les colonnes et ensuite les données triées par identifiant d'usine) sont générés:

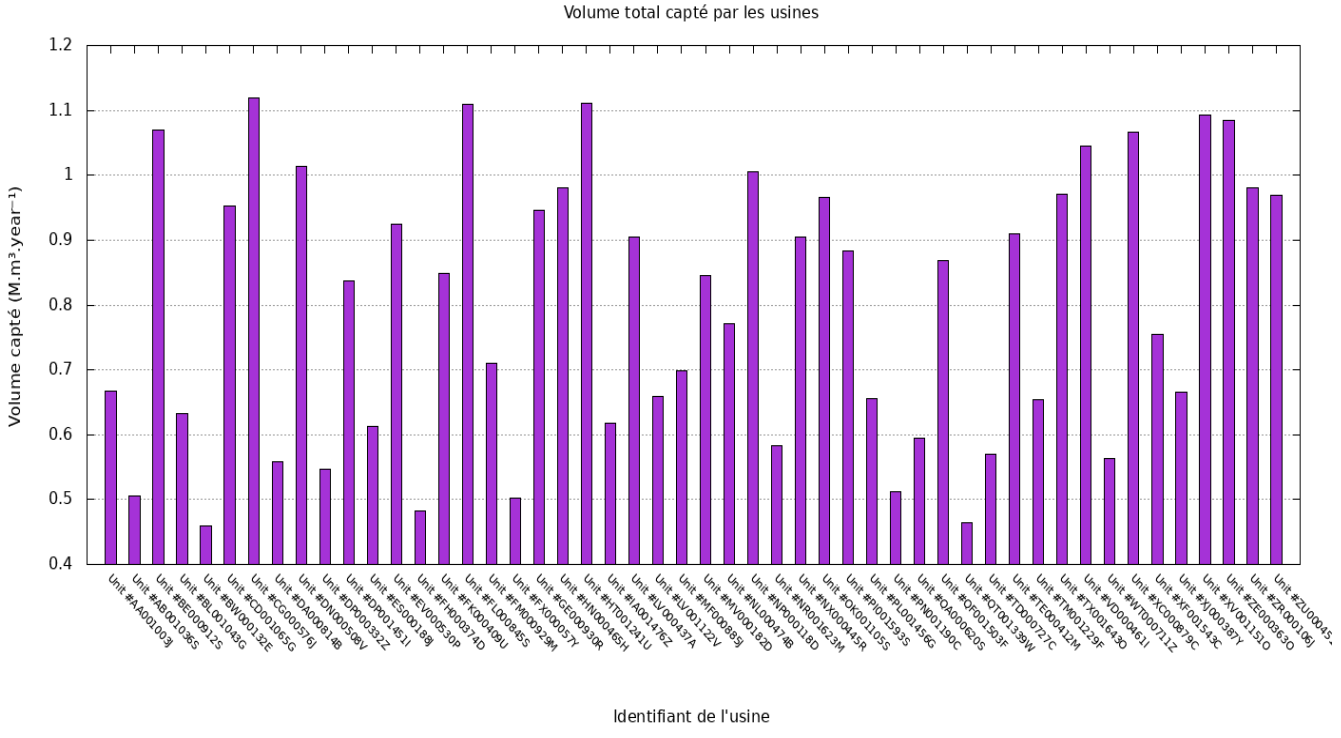
- 'vol_traitement.dat' : volume réellement traité par les usines
- 'vol_max.dat' : capacité maximale de traitement des usines
- 'vol_captation.dat' : volume total capté depuis les sources

À partir de ces fichiers, le script shell et Gnuplot génèrent automatiquement des histogrammes :

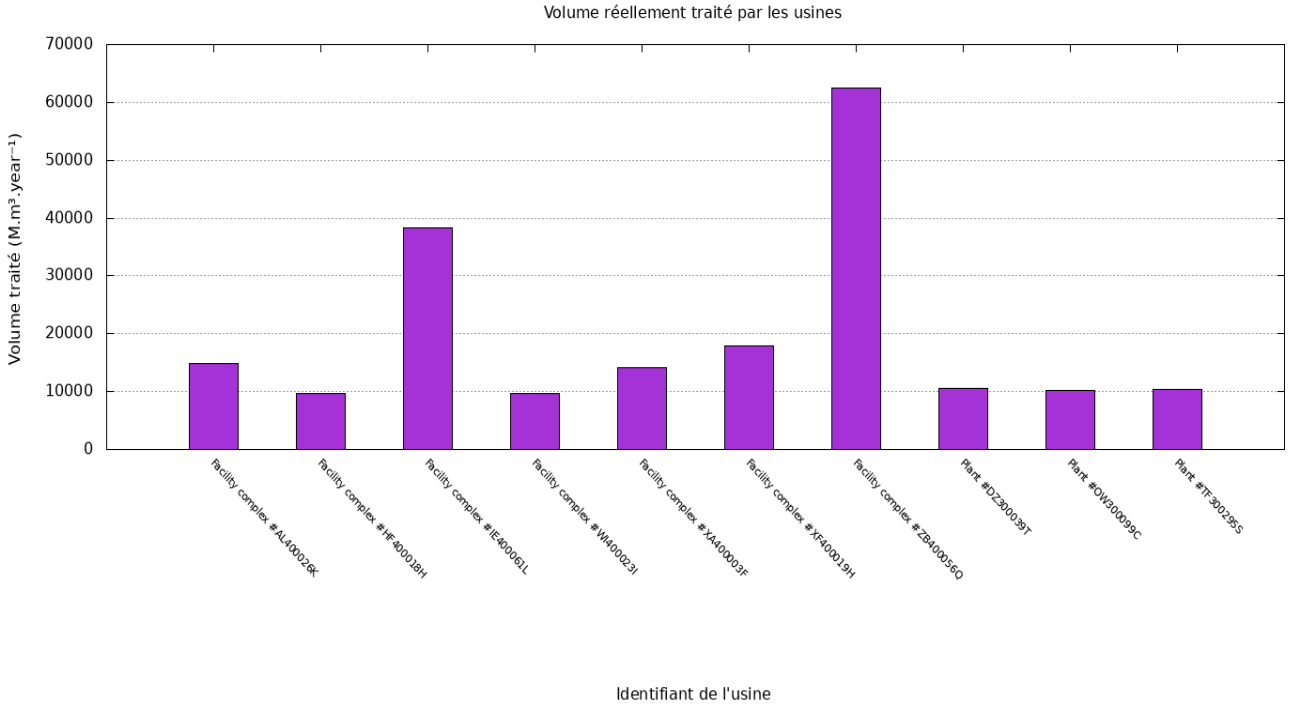
- 50 plus petites usines :

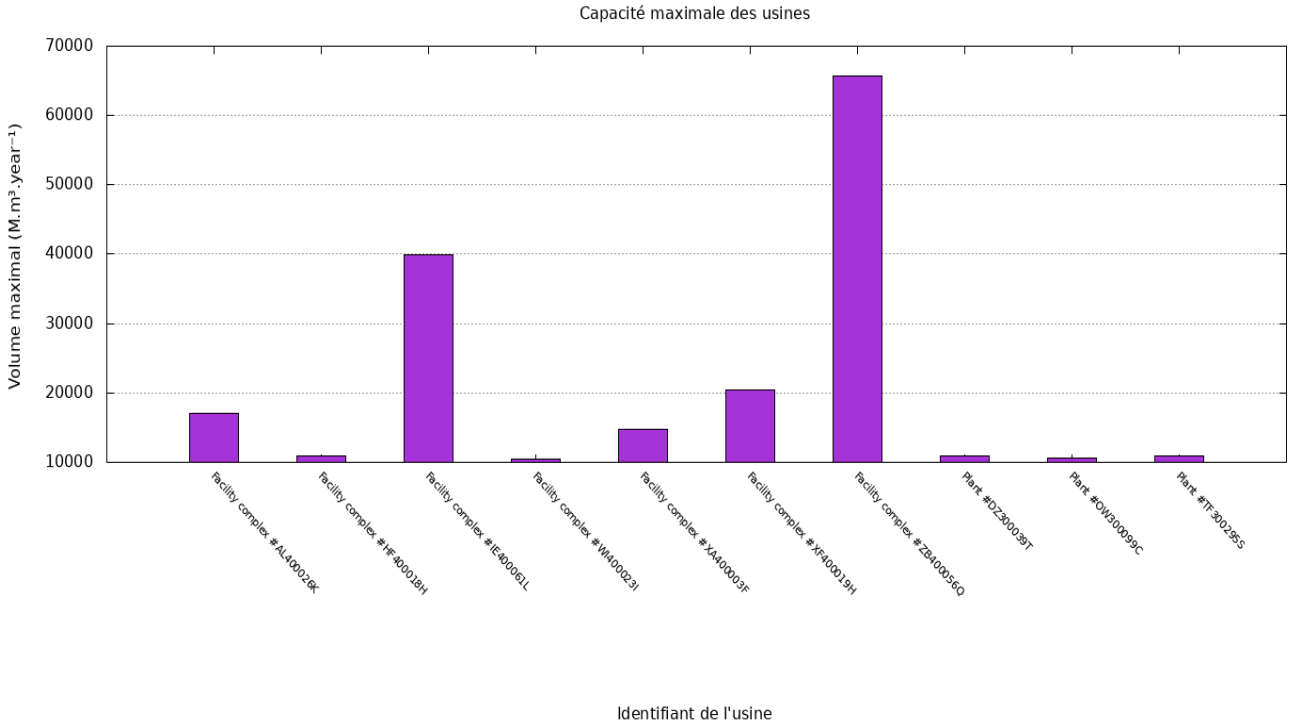


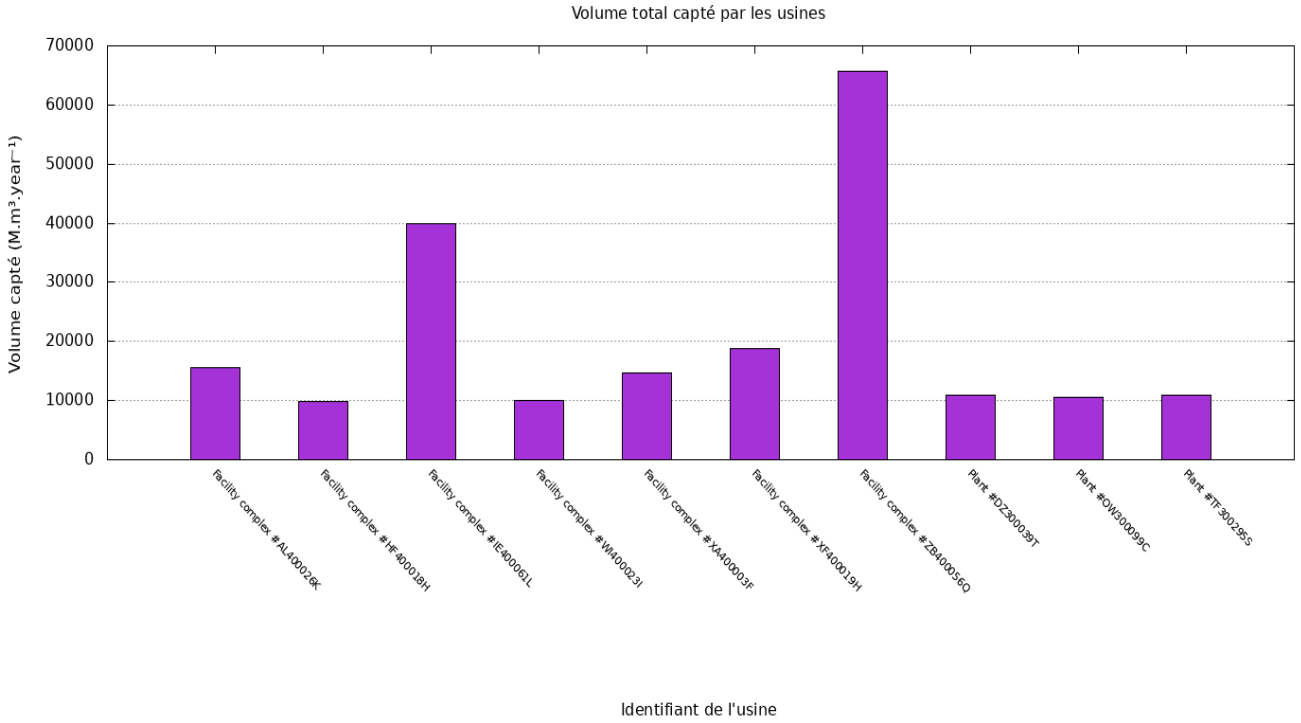




- 10 plus grandes usines :


Volume réellement traité par les usines
Volume traité (M.m³.year⁻¹)
70000
60000
50000
40000
30000
20000
10000
0
Facility complex #AL400026K
Facility complex #HF400018H
Facility complex #IE400061L
Facility complex #WI400023I
Facility complex #XA400003F
Facility complex #XF400019H
Facility complex #ZB400056O
Plant #DZ300039T
Plant #OW300099C
Plant #TF300295S
Identifiant de l'usine


Capacité maximale des usines
Volume maximal (M.m³.year⁻¹)
70000
60000
50000
40000
30000
20000
10000
Facility complex #AL400026K
Facility complex #HF400018H
Facility complex #IE400061L
Facility complex #WI400023I
Facility complex #XA400003F
Facility complex #XF400019H
Facility complex #ZB400056O
Plant #DZ300039T
Plant #OW300099C
Plant #TF300295S
Identifiant de l'usine



-Après exécution du programme avec la commande leaks, un fichier rendements.dat est créé (contenant l'identifiant de l'usine et le volume d'eau potable perdu sur chaque ligne).

Ainsi ce projet nous a permis de consolider nos connaissances en langage C, d'améliorer nos capacités de débogage et de travailler efficacement en binôme sur un projet de taille conséquente malgré les difficultés rencontrées, en particulier lors de la mise en commun des différentes parties du code.