# v1.02 Projet CY-Météo

**FILIERE** préING2 • 2022-2023
**AUTEURS** E.ANSERMIN – R.VERIN – R.GRIGNON
**E-MAIL** ean@cy-tech.fr - rgn@cy-tech.fr

## DESCRIPTION GENERALE

- Le but de ce projet est de réaliser une application qui va traiter un fichier de **données météorologiques** (données à télécharger dans le lien en bas de cet énoncé) dans le but d'afficher des **graphiques**.
- Les données traitées seront multiples et au choix de l'utilisateur : précipitations, températures, humidité, vitesse et orientation des vents, etc… Il pourra également choisir le lieu et la période des données à analyser
- Un script en langage **Shell** permettra à l'utilisateur d'indiquer les données qu'il souhaite analyser, de filtrer ces données et d'en faire des graphiques.
- L'affichage correct d'un graphique nécessite d'avoir des données triées. Ici la base de données est très volumineuse et la méthode utilisée pour effectuer le tri est donc cruciale quant au temps d'exécution du programme. C'est pourquoi le script Shell appellera un programme en langage **C** donc le but est le tri des informations et l'écriture dans un fichier des informations triées.
- Ces tris pourront être effectués via différentes méthodes (à choisir lors de l'exécution) :
  - utilisation d'un **AVL.**
  - utilisation d'un **ABR**
  - utiisation d'un **tableau** ou d'une **liste chaînée.**
- Ainsi, le **programme C** ne fera que **trier** les lignes du fichier de données fourni, et générer un/des fichiers de données en sortie. Le programme C doit retourner le code 0 s'il arrive à effectuer le travail prévu, et une valeur strictement positive sinon.
- Le **script Shell** devra **filtrer** les données en fonction des demandes de l'utilisateur, créer un fichier avec ces données et appeller le programme C pour lui demander de trier ces données. Il créera et affichera ensuite des graphiques (carte de température ou d'humidité dans une certaine zone géographique, carte des vents, courbes de températures mini/maxi/moyennes, etc….)

## SCRIPT SHELL

- Le script doit tout d'abord lire le fichier de données météorologiques et écrire un ou plusieurs fichiers avec les données filtrées en fonction des besoins de l'utilisateur. Ces fichiers de données filtrées seront ensuite triées par le programme C. Les données triées seront enfin utilisées par le script pour générer des graphiques.
- L'utilisateur indiquera les données dont il veut tracer les graphiques au moyen d'arguments passés au script. Ces arguments sont classés par catégories, en

fonction de leurs rôles.

- **Les arguments et options du script [TYPE de données] :**
  - Le script doit permettre de traiter au choix un ou plusieurs types de données comme la température (option associée **-t**), la pression atmosphérique (option associée **-p**), le vent (option associée **-w**), l'humidité (option associée **-m**), ou l'altitude de chaque station (option associée **-h**).
  - Au moins l'une de ces options doit être choisie pour que le script fonctionne. Sinon un message d'erreur doit s'afficher.
  - Il est possible d'en activer plusieurs : il n'y a pas de limitation particulière à ce niveau là.

  - Les options **-t** et **-p** doivent être accompagnées d'un mode :
    **-t<mode>** : (*t*)emperatures.
    **-p<mode>** : (*p*)ressions atmosphériques.
  - Pour ces 2 options, il faut indiquer la valeur du ***<mode>*** :
    - **1** : produit en sortie les températures (ou pressions) minimales, maximales et moyennes par station dans l'ordre croissant du numéro de station.
    - **2** : produit en sortie les températures (ou pressions) moyennes par date/heure, triées dans l'ordre chronologique. La moyenne se fait sur toutes les stations.
    - **3** : produit en sortie les températures (ou pressions) par date/heure par station. Elles seront triées d'abord par ordre chronologique, puis par ordre croissant de l'identifiant de la station

  - L'option -**w** : vent ( (*w*)ind )
    produit en sortie l'orientation moyenne et la vitesse moyenne des vents pour chaque station. Quand on parle de moyenne, il s'agira de faire la somme pour chaque composante du vecteur, et d'en faire la moyenne une fois tous les vecteurs traités. On aura donc une moyenne sur l'axe X et une moyenne sur l'axe Y : les 2 résultats fournissant le module et l'orientation moyens demandés. Les données seront triées par identifiant croissant de la station.

  - L'option **-h** : altitude ( (*h*)eight).
    Produit en sortie l'altitude pour chaque station. Les altitudes seront triées par ordre décroissant.

  - L'option **-m** : humidité ( (*m*)oisture ).
    Produit en sortie l'humidité maximale pour chaque station. Les valeurs d'humidités seront triées par ordre décroissant.

- **Les arguments et options du script [LIEUX] :**
  - Le script doit permettre de filtrer les données sur des critères supplémentaires comme la position géographique des mesures.
  - Ces options sont exclusives entre elles. Cela signifie qu'une seule option à la fois peut être activée au maximum.
  - Ces options ne sont pas obligatoires : si aucune n'est activée, il n'y aura alors pas de limitation géographique sur les mesures traitées.

- La liste des options de limitation géographique est la suivante :
  - option **-F** : (*F*)rance : France métropolitaine + Corse.
  - option **-G** : (*G*)uyane française.
  - option **-S** : (*S*)aint-Pierre et Miquelon : ile située à l'Est du Canada
  - option **-A** : (*A*)ntilles.
  - option **-O** : (*O*)céan indien.
  - option **-Q** : antarcti(*Q*)ue.

➢ **Les arguments et options du script [DATES] :**

- Il est possible d'indiquer un intervalle temporel pour filtrer les données de mesures. Seules les mesures dans l'intervalle de temps seront traitées.
- Cette option n'est pas obligatoire : il n'y aura alors pas de limitation temporelle sur les mesures traitées.

- **-d <min> <max>** : (*d*)ates
  les données de sortie sont dans l'intervalle de dates [<min>..<max>] incluses. Le format des dates est une chaine de type YYYY-MM-DD (année-mois-jour).

➢ **Les arguments et options du script [TRIS] :**

- Il est possible d'imposer le mode de tri des données : soit à l'aide d'un tableau (ou liste chaînée), soit à l'aide d'une structure d'arbre binaire, ABR ou AVL.
- Ces options de tris sont exclusives entre elles : une seule peut être activée à la fois.
- Si aucune option de tri n'est activée, par défaut le tri se fera à l'aide d'un AVL qui sera la plus efficace

- **--tab** : tri effectué à l'aide d'une structure linéaire (au choix un tableau ou une liste chaînée)
- **--abr** : tri effectué l'aide d'une structure de type ABR
- **--avl** : tri effectué à l'aide d'une structure de type AVL

➢ **Les arguments et options du script [FICHIER] :**

- Le nom du fichier d'entrée doit être renseigné pour que le script puisse acquérir toutes les données.

- **-f <nom_fichier>** : (*f*)ichier d'entrée.
  Cette option est obligatoire.

➢ L'option **--help** doit permettre l'affichage d'une aide détaillée à l'utilisation du script.
➢ Les options peuvent être rentrées dans n'importe quel ordre.
➢ De manière générale, si les arguments entrés par l'utilisateur ne correspondent pas au cahier des charges, il faut qu'un message d'erreur s'affiche, et que le programme quitte avec un code d'erreur. Aucun traitement ne doit être fait si les combinaisons d'options (absence ou présence) ne respectent pas le cahier des charges.
➢ Si le script arrive à s'exécuter correctement jusqu'au bout, le code de retour 0 sera renvoyé.

- **Exemple de commande :**

```
> ./script_meteo.sh -p2 -w -A -f data.csv
> ./script_meteo.sh -A -f data.csv -p2 --avl -w
```

Ces 2 commandes sont fonctionnellement identiques : on souhaite tracer les 2 graphiques correspondant aux mesures de vent et de pression moyenne aux Antilles pour n'importe quelle date et les données du fichier data.csv seront triées à l'aide d'un AVL.

- **Fonctionnement du script et communication avec le programme C :**
  - Le script devra vérifier si l'exécutable issu du programme C est présent. Si ce n'est pas le cas, il devra faire en sorte de lancer la compilation automatiquement avant de faire le traitement prévu (et vérifier que la compilation est correcte avant de lancer le traitement).
  - C'est le Script qui lance l'exécution du programme C en fonction des besoins.
  - Les étapes suivantes sont effectuées plusieurs fois, une fois pour chaque type de donnée (options -t, -p, -w, -h, -m), dans n'importe quel ordre :
    - Le script va préalablement filtrer les données du fichier de données en accord avec les options, écrire le résultat du filtrage dans un fichier temporaire. Ce fichier sera le fichier d'entrée du programme C. Le format est laissé au libre choix du développeur.
    - Le script lance le programme C en passant comme argument le nom du fichier filtré et le mode de tri (tableau, ABR ou aVL)
    - Le programme C trie les données en accord avec ses arguments et écrit un fichier en sortie.
    - Le script vérifie que le programme C s'est terminé avec succès et récupère le fichier de données triées qui vient d'être généré.
    - Le script utilise les données triées pour créer un graphique dans une image (en utilisant l'utilitaire gnuPlot).

- **Production des graphiques de sortie :**
  - Le script, une fois les appels au programme C effectués, doit agréger le ou les fichier(s) de données généré(s) afin de créer les images attendues. L'utilitaire ***gnuplot*** permet de créer des images à partir de données.

  - Chacun des diagrammes suivants doit donc être généré par le script, en accord avec les arguments :
  - **températures (-t) / pressions (-p) en mode 1** :
    diagramme de type *barres d'erreur* avec en abscisse l'identifiant de la station, et en ordonnée le minimum, maximum et la moyenne.
  - **Températures (-t) / pressions (-p) en mode 2** :
    diagramme de type *ligne simple* avec en abscisse le jour et l'heure des mesures, et en ordonnée la moyenne des mesures.
  - **températures (-t) / pressions (-p) en mode 3** :
    diagramme de type *multi-lignes* avec en abscisse les jours, et en ordonnée les valeurs mesurées. Ce diagramme contiendra toutes les lignes, 1 par station et par heure.
    Toutes les courbes pour une même heure (ex : 17h) auront la même couleur. Il y aura donc un groupe de lignes de la même couleur (avec autant de lignes qu'il y a de stations). Et il y aura N groupes de lignes, autant qu'il y a d'heures différentes.

- **vent (-w)** :
  diagramme de type *vecteurs* (flèches orientées) avec l'abscisse correspondant à la longitude (axe Ouest-Est) et l'ordonnée correspondant à la latitude (axe Sud-Nord). On ne vous demande pas d'effectuer une projection cartographique précise : vous pourrez donc utiliser les coordonnées GPS de manière linéaire.
- **altitude (-h)** :
  diagramme de type *carte interpolée et colorée* avec l'abscisse correspondant à la longitude (axe Ouest-Est) et l'ordonnée correspondant à la latitude (axe Sud-Nord).
  Les couleurs utilisées seront libres mais il vous est demandé d'utiliser 3 teintes différentes au minimum.
- **humidité (-m)** :
  diagramme de type *carte interpolée et colorée* avec l'abscisse correspondant à la longitude (axe Ouest-Est) et l'ordonnée correspondant à la latitude (axe Nord-Sud).

**PROGR. C**

- Le programme C doit prendre en entrée un fichier de données, les trier et produire un ou plusieurs fichiers en sortie avec les données triées.
- Le programme C ne gère qu'un type de donnée à la fois, si l'utilisateur souhaite avoir des informations sur la température et le vent, le programme C sera appelé deux fois avec des données à traiter différentes en entrée.
- Ce programme sera exécuté avec des arguments correspondant, entre autres, au type de méthode de tri (tableau, ABR, AVL) .
- **Le programme doit être modulé**, et un fichier **Makefile** doit être présent dans votre dossier afin de compiler automatiquement l'ensemble de votre programme.
- Si le programme C arrive à effectuer son traitement correctement jusqu'au bout, il doit retourner la **valeur 0**, sinon il doit retourner une valeur strictement positive. Cette valeur d'erreur sera récupérée et traitée par le script Shell.
- **Les arguments du programme C sont :**
  - option **-f <nom_fichier>** :
    chemin du fichier des données filtrées. Cette option est obligatoire.
  - Option **-o <nom_fichier>** :
    chemin du fichier de sortie. Cette option est obligatoire.
  - option **-r** :
    permet de trier dans le sens décroissant. Cet argument est optionnel. Si il n'est pas renseigné, le tri se fera dans le sens croissant par défaut.
  - options **--tab / --abr / --avl** :
    permettent de choisir la méthode de tri à adopter. Ces options sont exclusives entre elles. Si aucune n'est activée, le tri se fera par AVL.
- **Les valeurs de retour du programme C sont :**
  - **0** : le programme C s'est terminé correctement et les opérations demandées ont été effectuées avec succès.
  - **1** : erreur sur les options activées (mauvaise combinaison, option obligatoire manquante, … )
  - **2** : erreur avec le fichier de données d'entrée (impossible de l'ouvrir, impossible de lire le contenu, format de données incorrect, … )
  - **3** : erreur avec le fichier de données de sortie (impossible de l'ouvrir, d'écrire dedans, … )
  - **4** : toute autre erreur d'exécution interne

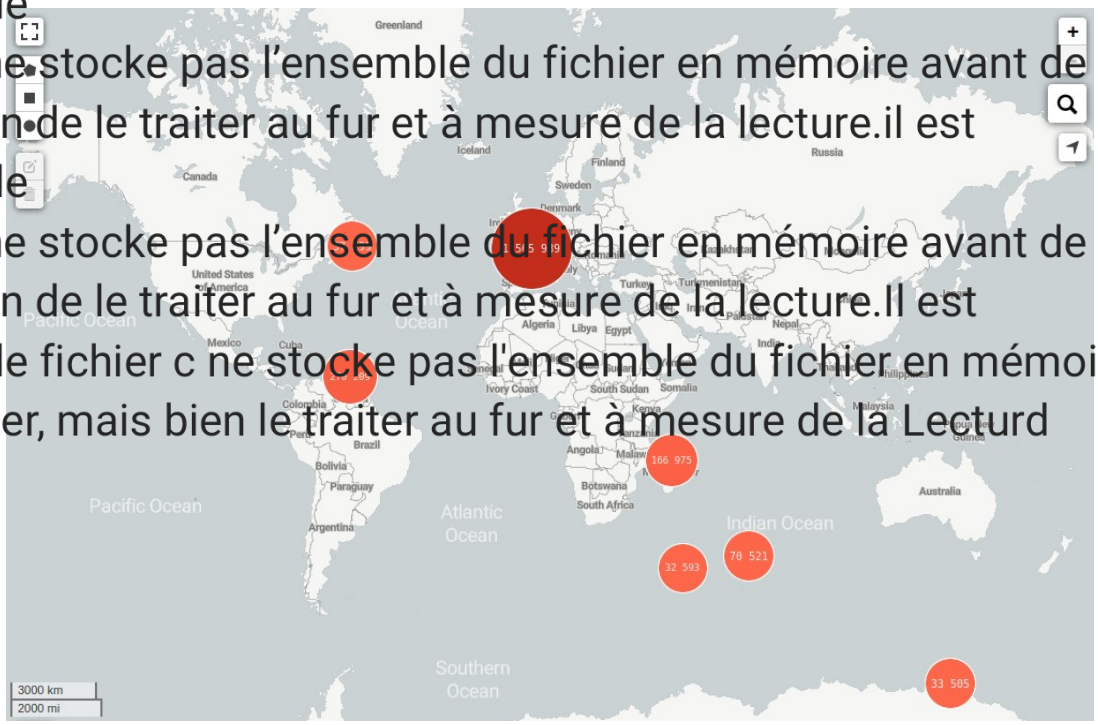
**BONUS**

- Des options du script supplémentaires peuvent être ajoutées si vous le désirez, afin de vous apporter des points bonus :
  - **-g <min> <max>** : lon(*g*)itude.
    Permet de filtrer les données de sortie en ne gardant que les données qui sont dans l'intervalle de longitudes [<min>..<max>] incluses. Cette option est exclusive avec les options de lieux. Le format des longitudes est un nombre réel.
  - **-a <min> <max>** : l(*a*)titude.
    Permet de filtrer les données de sortie en ne gardant que les relevés qui sont dans l'intervalle de latitudes [<min>..<max>] incluses. Cette option est exclusive avec les options de lieux. Le format des latitudes est un nombre réel.
- Vous pouvez ajouter d'autre fonctionnalités de votre choix à partir du moment où le cahier des charges du projet est rempli.

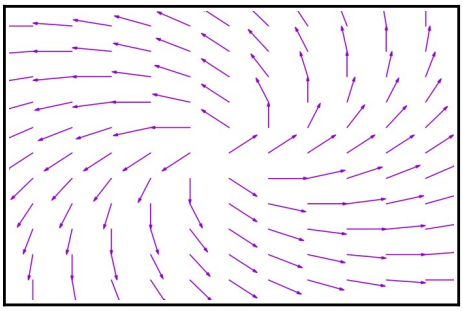
**INFO. SUPP.**

- **Format et contenu du fichier d'entrée**
  Le fichier de données contient des données météorologiques de différentes stations sur les territoires francophones entre 2010 et 2022.
  Dans le fichier de données, il y a plus de 2 millions de relevés, avec environ 60 stations de mesures distinctes, plus de 4500 dates différentes, et environ 8 horaires de mesures par date.
  Ce fichier est un fichier issu d'un tableur. Il est au format csv.
- **Taille des données d'entrée**
  La taille du fichier original est de 228Mo.
  Malgré le filtrage qui sera effectué par le script, il est primordial que le programme C ne stocke pas l'ensemble du fichier en mémoire avant de le traiter, mais bien de le traiter au fur et à mesure de la lecture.
- **Position géographique des relevés**

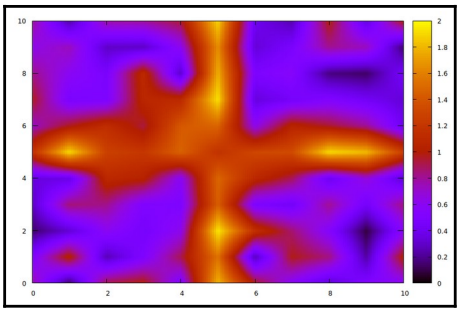
il est primordial que le programme C ne stocke pas l'ensemble du fichier en mémoire avant de le traiter, mais bien de le traiter au fur et à mesure de la lecture.il est primordial que le programme C ne stocke pas l'ensemble du fichier en mémoire avant de le traiter, mais bien de le traiter au fur et à mesure de la lecture.il est primordial que le programme C ne stocke pas l'ensemble du fichier en mémoire avant de le traiter, mais bien de le traiter au fur et à mesure de la lecture.Il est primordial que le fichier c ne stocke pas l'ensemble du fichier en mémoire avant de le traiter, mais bien le traiter au fur et à mesure de la Lecturd



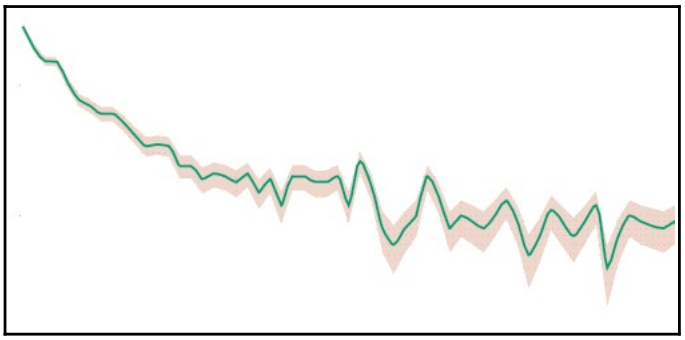
- **Exemple de graphique de type ‘vecteurs’ (carte des vents) :**



- **Exemple de graphique de type ‘carte interpolée’ (altitude, humidité) :**



- **Exemple de graphique de type ‘barres d’erreurs’ :**



## CRITERES DE NOTATION

- Le rendu du travail sera un **lien github** (ou **gitlab**) menant au projet accessible publiquement. Inutile d’envoyer les fichiers par email : le seul livrable attendu et qui sera évalué sera le lien du dépôt git dans lequel doivent se trouver tous les fichiers de votre projet.
- Le dépôt de code contiendra en plus des fichiers de code, un ReadMe contenant les instructions pour compiler ET utiliser votre application, un document au format PDF présentant la répartition des tâches au sein du groupe, ainsi que le planning de réalisation. Enfin ce document contiendra les limitations fonctionnelles (la liste de ce qui n’est pas implémenté, et/ou ce qui est implémenté mais qui ne fonctionne pas correctement).
- Votre rendu contiendra des exemples d’exécution de votre application. Vous mettrez dans des dossiers séparés, les images, fichiers de données d’entrée, intermédiaires et finaux, et vous présenterez ces résultats dans le document PDF cité précédemment.
- Le rendu est un travail de groupe : si des similitudes entre groupes sont trouvées, et/ou si des exemples disponibles sur Internet sont découverts sans être sourcés, une procédure de fraude à un examen pourra être envisagée.
- Le code sera séparé en **modules** (fichiers .c et .h dans des sous-dossiers).

- Un fichier **Makefile** sera présent à la **racine** du projet et il permettra de compiler l'exécutable. La première cible permettra de compiler le projet. Ce fichier inclura entre autres une cible '**clean**' qui permet d'effacer les fichiers générés.
- Votre code sera **commenté** (modules/fonctions/structures/constantes) et correctement **indenté**.
- L'ensemble des **symboles** du code (variables, fonctions, types, …) sera dans la **même langue** que les **commentaires** (soit tout en anglais, soit tout en français, mais pas de mélange entre les langues utilisées).
- Le programme C doit respecter toutes les consignes décrites plus haut.
- Le script shell doit respecter toutes les consignes décrites plus haut.
- Le programme C et le script shell ne doivent en aucun cas générer d'erreur inattendue. Il ne doit y avoir aucune **erreur de segmentation**, **erreur de syntaxe**, ou de nom de **commande inconnue**, etc…
- Si une erreur est détectée par votre programme/script, un message d'erreur doit s'afficher pour indiquer la cause à l'utilisateur, et un code retour avec une valeur strictement positive doit être retournée.
- Les structures allouées temporairement dans votre programme doivent être désallouées explicitement (une bonne pratique est que pour chaque appel de malloc, un appel à free doit être fait avant la fin de l'exécution de votre programme). La quantité de mémoire vive consommée par votre programme pourra être un élément d'évaluation.

## RESSOURCES UTILES

**GitHub**

- site Web : https://github.com/

**Données météorologiques France (SYNOP / SMT / OMM)**

- site Web : https://public.opendatasoft.com/explore/dataset/donnees-synop-essentielles-omm/

**Format CSV**

- site Web : https://fr.wikipedia.org/wiki/Comma-separated_values

**GnuPlot**

- site Web : http://gnuplot.info/
- exemples diagrammes :
  - '***lignes***' : https://gnuplot.sourceforge.net/demo_5.4/simple.html
  - '***barres d'erreur***' : https://gnuplot.sourceforge.net/demo_5.4/errorbars.html
  - '***vecteurs***' : https://gnuplot.sourceforge.net/demo_5.4/vector.html
  - '***carte interpolée***' : https://gnuplot.sourceforge.net/demo_5.4/pm3d.html