



METHODOLOGIE SCIENTIFIQUE



ANNEE 2019-2020

NICOLAS EVAIN ET AXEL PHANOR

L2 Informatique, Parcours Numérique pour les Sciences du Vivant
Collège Sciences et technologies pour l'Énergie et l'Environnement

Épigraphe

"L'expérience prouve que celui qui n'a jamais confiance en personne ne sera jamais déçu."

Léonard De Vinci

Remerciement

Nous tenons à remercier Monsieur Naïja Mohammed pour l'enseignement ainsi que toutes les informations qu'il nous a transmises pour nous aider à monter ce rapport. Nous tenons également à remercier l'ensemble de l'équipe pédagogique de l'UPPA pour son enseignement qui nous a permis d'acquérir les compétences informatiques nécessaire à la réalisation de ce projet.

Avant-propos

Cette unité d'enseignement est destinée aux étudiant désireux d'acquérir une connaissance sur la méthodologie scientifique. La méthode scientifique désigne l'ensemble des canons guidant ou devant guider le processus de production des connaissances scientifiques, qu'il s'agisse d'observations, d'expériences, de raisonnements, ou de calculs théoriques.

Ce rapport comporte notre méthodologie scientifique sur la problématique du réchauffement climatique.

Un projet de récupération, d'analyse et de traitement de données environnementales basé sur des expérimentations. Le but étant de construire un model capable d'anticiper des événements à partir de données afin de proposer une analyse décisionnelle. Le tout basé sur la création de courbes et de graphique.

Pour conclure cet avant-propos, je remercie, par avance, le lecteur pour toute remarque de fonds ou de forme.

Nicolas Evain et Axel Phanor
Étudiants en L2 Informatique Numérique pour les Sciences du Vivant
Faculté des sciences et Techniques de la Côte Basque
Université de Pau et des Pays de l'Adour

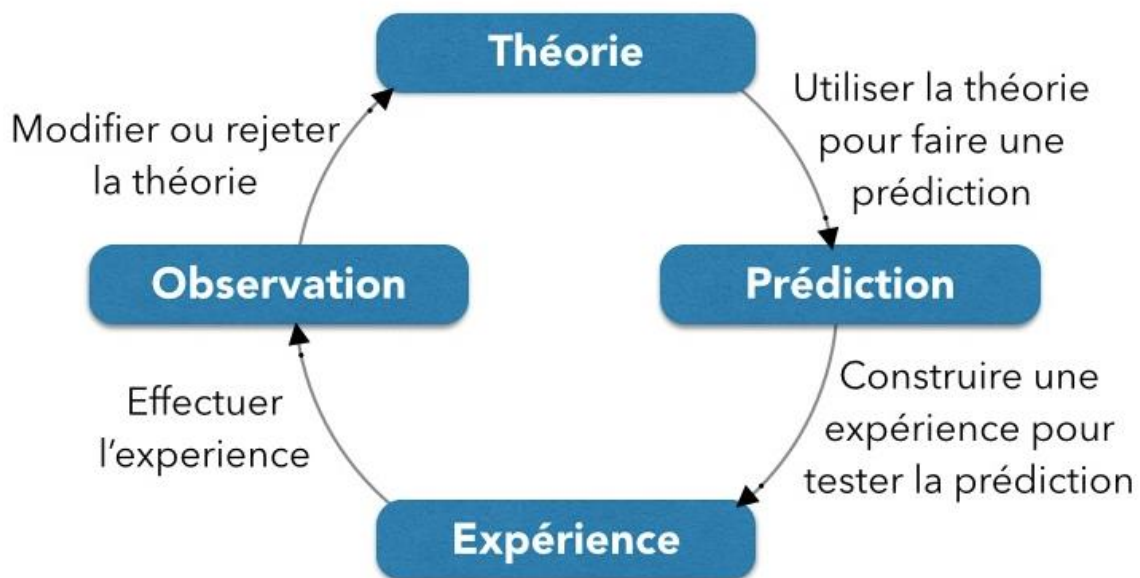
Sommaire

Table des matières

Épigraphe	2
Remerciement.....	3
Avant-propos	4
Sommaire.....	5
1. La Méthode Scientifique	6
2. Le Réchauffement Climatique : Observation et théorie	7
3. Identification et Hypothèses	9
4. Expérimentation	11
4.1. Expérience : Augmentation des températures.....	11
4.1.1. Récupération de données en utilisant une API	12
4.1.2. Exploitation et traitement des données Absolute	22
5. Analyse des résultats et Conclusion	27
Annexes	30
Code de getTemp.c :	30
Code de main.c :	31
Macro Excel (VBA) temp :	36

1. La Méthode Scientifique

La "Méthode Scientifique" consiste à appliquer le processus suivant sur une problématique donnée :



Avant dévoiler notre projet voici une courte présentation de la "Méthode Scientifique", processus que nous allons suivre tout au long de notre présentation :

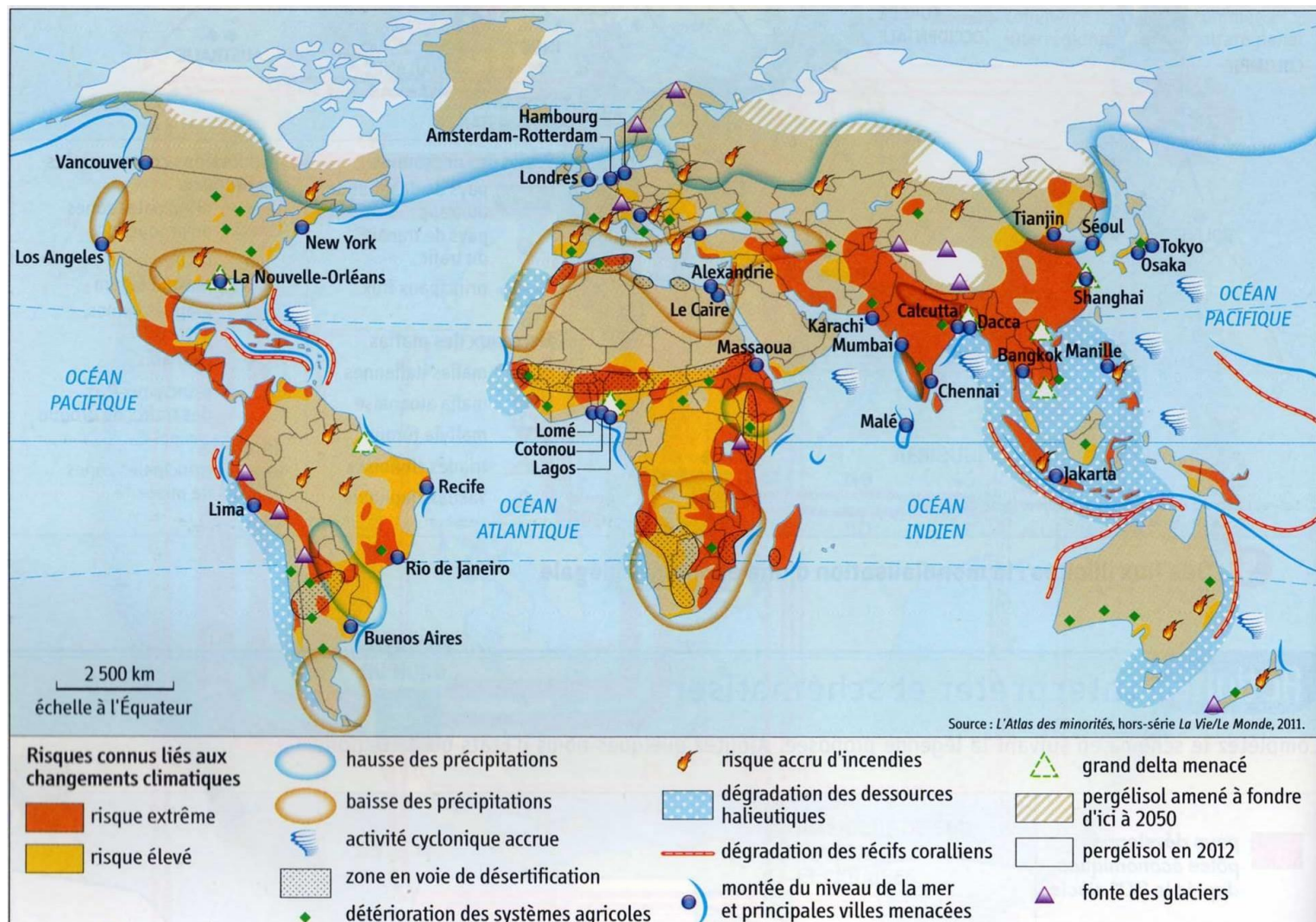
1. Identification du problème : Trouver le problème à résoudre, dans notre cas nous allons étudier quelques aspects du réchauffement climatique.
2. Formulation d'hypothèse : prévoir la réponse à la question.
3. Expérimentation : expérimenter pour trouver la réponse à la question. Il faut noter des observations et recueillir des données
4. L'analyse des résultats : faire des tableaux et des graphiques à partir de nos données pour comprendre les résultats.
5. Conclusion : On communique les découvertes. On fait une synthèse, on rappelle le but, on donne une réponse à la question et on fait un retour sur l'hypothèse.

2. Le Réchauffement Climatique : Observation et théorie

Le réchauffement climatique est la principale préoccupation du xxi^e siècle à l'échelle mondiale. Les températures à la surface du globe ne cessent d'augmenter depuis 1950, ce réchauffement est principalement dû à l'augmentation des gaz à effets de serres d'origine anthropique.

Les dernières projections du GIEC (Groupe d'experts intergouvernemental sur l'évolution du climat) sont que la température de surface du globe pourrait croître de 1,1 à 6,4 °C supplémentaires au cours du xxi^e siècle.

Cette évolution certaine des températures va impacter grandement toutes les espèces animales et végétales mais aussi les mécanismes planétaires comme les écosystèmes terrestre et marins, comme le montre la carte suivante :



Le changement des différents biomes entraîne un déplacement ou encore une disparition totale des espèces y vivant !

Ces espèces ne pouvant plus vivre dans leur environnement sont voués à en trouver un nouveau ou à s'éteindre, les principales espèces en voie de disparition sont celles qui sont le plus en dépendance de leur environnement, d'une alimentation spécifique ou celles qui résistent le moins aux variations de température.

Voici quelques exemples des espèces les plus affectées par le réchauffement climatique et qui risquent de disparaître prochainement : Le poisson limes à taches orange, le faux dragonnier, l'ours polaire, le manchot, la morue de l'atlantique du nord etc....

Mais lors de notre étude nous allons principalement nous intéresser à l'augmentation des températures depuis le 20^{ème} siècle à aujourd'hui et de son impact sur les ours polaire.

La survie des ours polaire dépend de la banquise qui se forme de plus en plus tard en automne pour disparaître de plus en plus tôt au printemps. La banquise arctique est le lieu de chasse de ce grand prédateur mais étant donné qu'elle disparaît à cause de l'augmentation des températures cela provoque des situations cocasses :



Pour plus d'information sur les ours polaire :

https://fr.wikipedia.org/wiki/Ours_blan

3. Identification et Hypothèses

Cette partie va nous permettre d'appliquer une méthodologie scientifique à la problématique du réchauffement climatique.

La première étape étant d'identifier les différents problèmes causés par le réchauffement climatique, voici le storyboard que nous proposons :



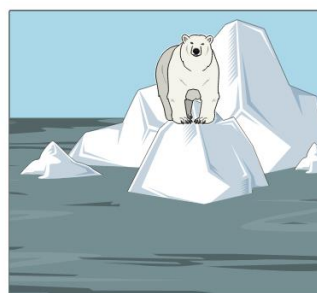
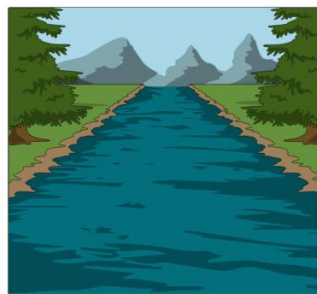
Réchauffement climatique

**L'impact du réchauffement sur les être vivants, l'écosystème, et sur le climat.
La planète subit des changements drastiques et irréversibles**

Depuis plusieurs années, nous observons un dérèglement climatique qui impacte et altère l'écosystème sur toute la surface du globe.

Les modifications de l'habitat naturel des différentes espèces provoquent leur disparition partielle ou totale.

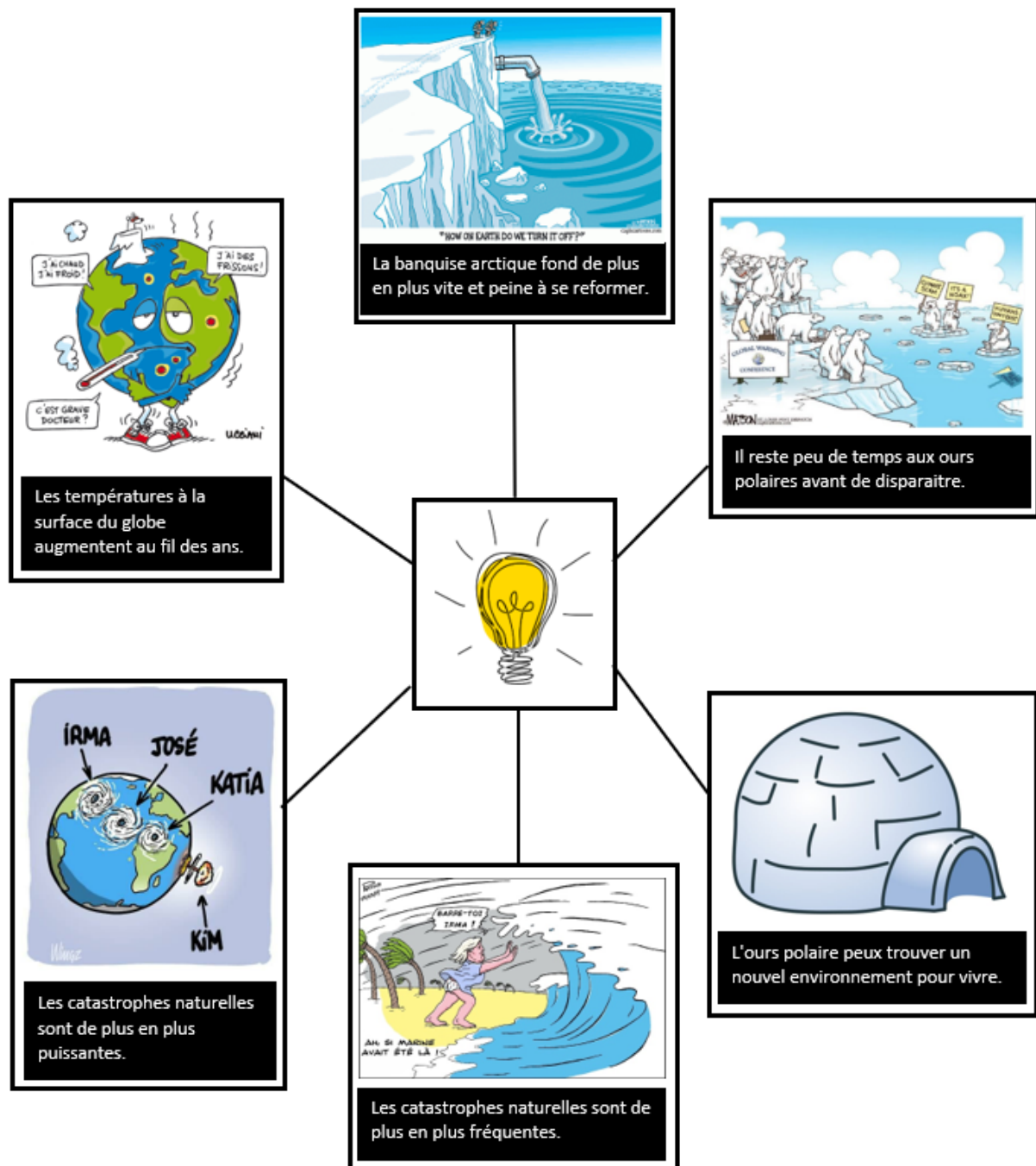
Le dérèglement climatique engendre des catastrophes naturelles plus fréquentes et surtout plus puissantes. Ce qui peut provoquer un effet boule de neige à l'échelle mondiale.



La problématique du réchauffement climatique soulève moult questions comme :

- Les températures à la surface du globe augmentent t'elles ?
- Le réchauffement climatique change t'il notre environnement ?
- Combien de temps reste-il aux espèces avant de disparaître ?
- Existe-il un refuge pour ces espèces ?
- Les catastrophes naturelles deviennent-elles plus fréquentes et puissante ?

Et j'en passe, mais intéressons-nous principalement à ces questions et formulons les hypothèses qui nous permettront de prévoir une réponse à chacune de ces questions.



4. Expérimentation

La partie expérimentale consiste à noter des observations mais surtout de collecter des données pour pouvoir les traiter et confirmer ou non nos hypothèses.

Pour réaliser une expérience et tester nos hypothèses il est **très important** d'obtenir des données importantes (big data), c'est-à-dire beaucoup de données pour accroître la pertinence de nos résultats mais surtout qu'elles soient correctes et en adéquation avec les hypothèses que nous avons formulé.

Note : Il est très difficile de trouver ce genre de données sur internet car elles représentent le fruit d'un travail difficile, long et très coûteux ! Ce qui fait que pour avoir accès à des données techniques il faut prévoir d'acheter la base de données à une entreprise ou un organisme.

Il est néanmoins possible après des heures de recherches peu passionnantes et souvent peu efficaces de trouver la perle rare : des données, bien qu'incomplètes mais exploitables.

Pour répondre à notre première hypothèse : "Les températures à la surface du globe augmentent au fil des ans" nous allons réaliser l'expérience suivante :

4.1. Expérience : Augmentation des températures.

Dans l'idéal, pour réaliser cette étude nous avons besoin de récolter les températures en chaque point à la surface du globe au fil du temps. Ces données sont inaccessibles voire n'existent pas. Il faudrait également remonter le plus loin dans le temps pour concrétiser le tout.

Mais plus on remonte dans le temps et plus l'homme n'était pas en possession d'outils nécessaires à la mesure de température. D'autant plus que les résultats étaient moins précis et non stockables dans des bases de données.

Ces facteurs peuvent fausser notre étude.

Notre étude : Nous avons donc décidé de comparer les moyennes des températures actuelles à la surface du globe avec les moyennes du passé.

La question est : "Comment récupérer ce type de données ?" sachant qu'il est difficile de mettre la main sur les données du passé.

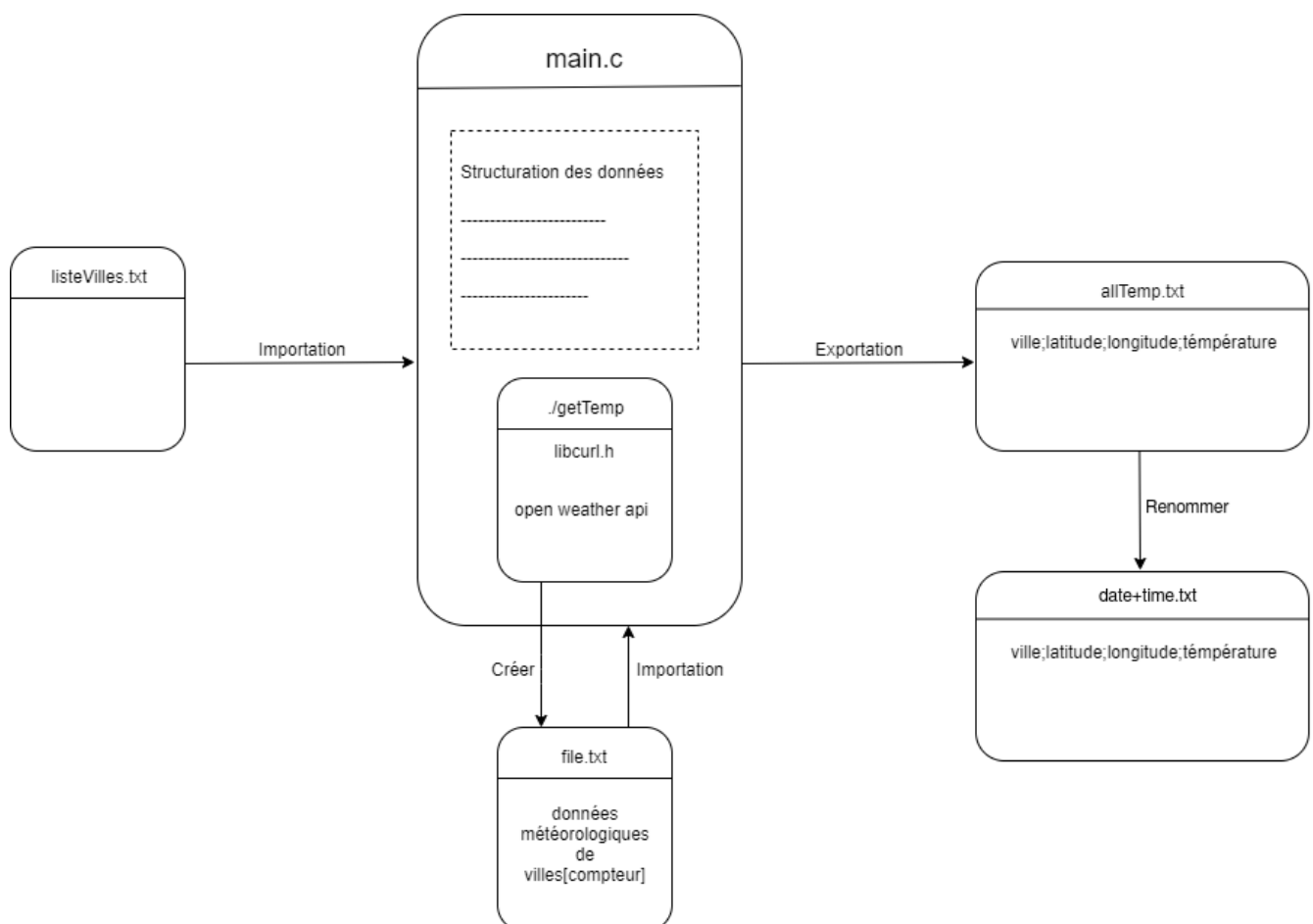
Solution :

1. Pour récupérer les températures actuelles à la surface du globe nous utiliserons une API.
2. Pour récupérer les températures moyenne à la surface du globe, nous avons réussi à trouver une base de données "Absolute" de la CRU (Climate Research Unit) qui nous a permis d'obtenir la moyenne des températures de 1961-1990 à la surface du globe.

4.1.1. Récupération de données en utilisant une API

Nous avons réalisé un programme stockant la longitude, la latitude et la température de chaque capital du monde dans un fichier texte.

Pour une meilleure compréhension de ce qui va suivre on a réalisé un schéma résumant comment est construit le programme.



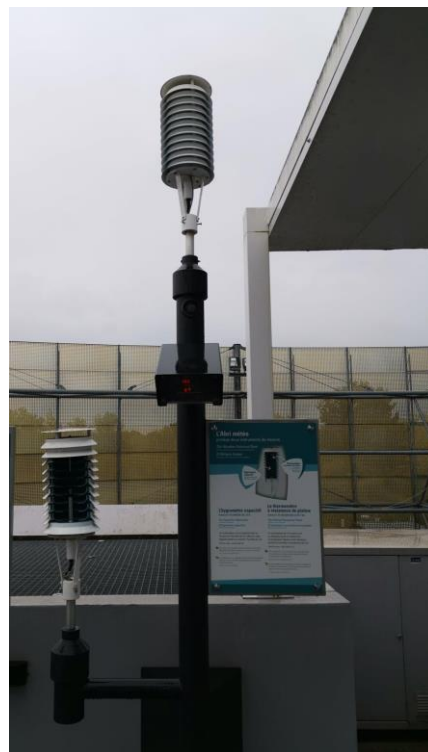
Les API :

API est l'acronyme d'**Application Programming Interface**, que l'on traduit en français par **interface de programmation applicative** ou **interface de programmation d'application**. L'API peut être résumée à une solution informatique qui permet à des applications de communiquer entre elles et de s'échanger mutuellement des services ou des données. Il s'agit en réalité d'un ensemble de fonctions qui facilitent, via un langage de programmation, l'accès aux services d'une application. Nous, nous avons utilisé une API REST qui signifie Representational State Transfer. C'est une API réalisant des requêtes HTTP.

Openweather :

Nous dans notre cas nous avons utilisé comme API openweather elle permet de récupérer des données météorologiques précises, et en temps réel d'une ville donnée en paramètre.

Ces données météorologiques sont récoltées grâce à divers capteurs récoltant ses données :



C'est deux photos sont des appareils servant à récolter des données. Celui de gauche est un pluviomètre qui va mesurer les précipitations et celui de droite est un abri météo à l'intérieur on trouve deux capteurs, un thermomètre à résistance de platine qui va mesurer la température de l'air et un hygromètre captatif qui va mesurer l'humidité de l'air.

Openweather est une API REST, ici vous pouvez voir comment est composé notre requête http :

<http://api.openweathermap.org/data/2.5/weather?q=Paris&units=metric&APPID=b7c635aa6d4f700b3ffd7a54f01b4958>

Ville Unité Identifiant

Ville correspond à la ville dont nous voulons les données.

Unité permet de choisir en quel unité est exprimé la température.
Identifiant : nous permet d'avoir accès à la base de données.

Cette requête nous renvoi :

```
{ "coord": { "lon": 2.35, "lat": 48.86 }, "weather": [ { "id": 501, "main": "Rain", "description": "mode  
rate  
rain", "icon": "10d" } ], "base": "stations", "main": { "temp": 7.01, "pressure": 1002, "humidity": 87, "temp_min": 6.11, "temp_max": 8, "visibility": 8000, "wind": { "speed": 7.7, "deg": 230 }, "rain": { "1h": 3.56 }, "clouds": { "all": 100 }, "dt": 1575878746, "sys": { "type": 1, "id": 6540, "country": "FR", "sunrise": 1575876702, "sunset": 1575906848, "timezone": 3600, "id": 2988507, "name": "Paris", "cod": 200 }
```

Si vous voulez en savoir plus sur le fonctionnement de cette api :

<https://openweathermap.org/current>

Libcurl :

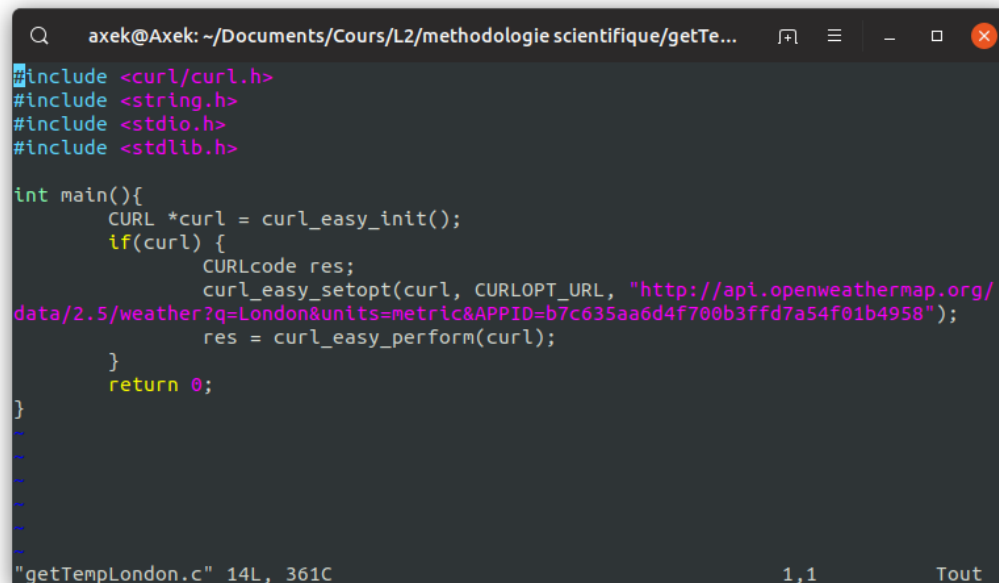
Pour obtenir ses données depuis un programme en C nous avons dû utiliser la librairie libcurl elle permet de réaliser des requêtes HTTP. Mais c'était plutôt complexe à la faire fonctionner.

Le premier problème que nous avons eu été que cette bibliothèque n'était pas installée de base sur nos machines. Il a donc fallu que nous téléchargions la bonne version. Dans notre cas nous avons télécharger la version curl-7.67.0.zip. Une fois le dossier curl-7.67.0 extrait. Puis on a récupéré le dossier curl pour le mettre dans usr/include (a noté que ce programme est codé sous Ubuntu).

EXAMPLE

```
CURL *curl = curl_easy_init();  
if(curl) {  
    CURLcode res;  
    curl_easy_setopt(curl, CURLOPT_URL, "http://example.com");  
    res = curl_easy_perform(curl);  
    curl_easy_cleanup(curl);  
}
```


Nous avons récupéré cet exemple sur le site de libcurl, il renvoi le résultat d'une requête dans la sortie de la console. Pour tester si le code fonctionner, nous avons d'abord essayé de le faire fonctionner avec une ville.

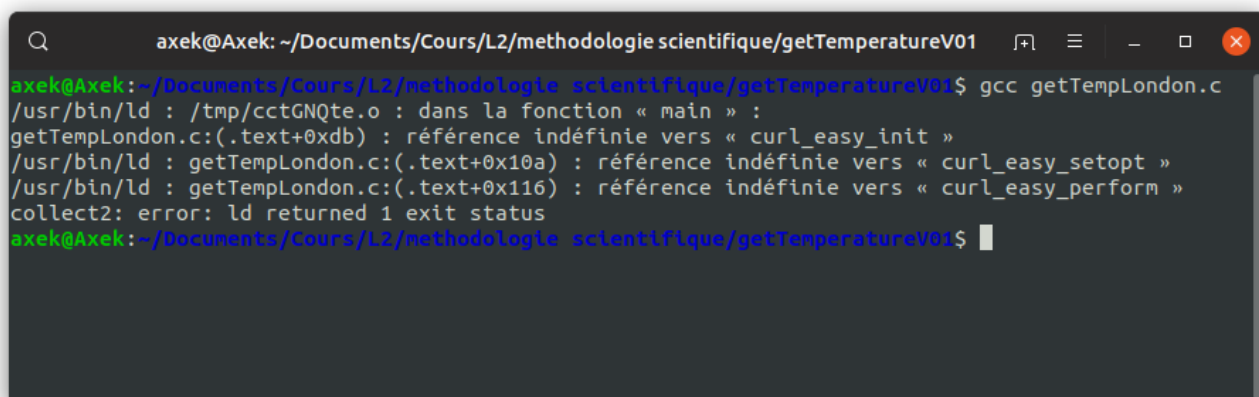


```
axek@Axek: ~/Documents/Cours/L2/methodologie scientifique/getTe...
#include <curl/curl.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

int main(){
    CURL *curl = curl_easy_init();
    if(curl) {
        CURLcode res;
        curl_easy_setopt(curl, CURLOPT_URL, "http://api.openweathermap.org/
data/2.5/weather?q=London&units=metric&APPID=b7c635aa6d4f700b3ffd7a54f01b4958");
        res = curl_easy_perform(curl);
    }
    return 0;
}

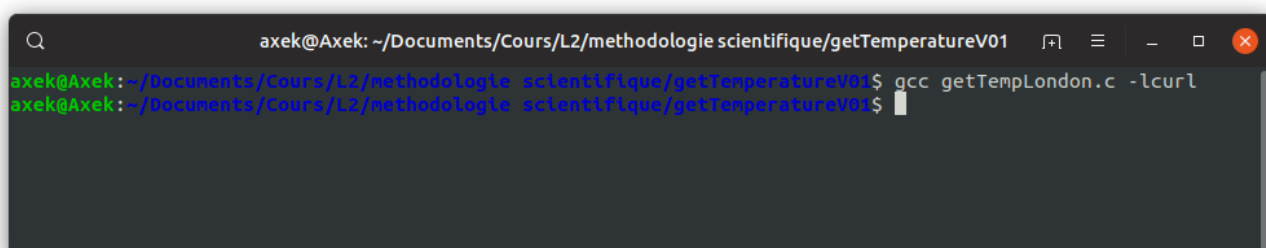
"getTempLondon.c" 14L, 361C 1,1 Tout
```

Lorsqu'on compile on obtient :



```
axek@Axek: ~/Documents/Cours/L2/methodologie scientifique/getTemperatureV01$ gcc getTempLondon.c
/usr/bin/ld : /tmp/cctGNQte.o : dans la fonction « main » :
getTempLondon.c:(.text+0xdb) : référence indéfinie vers « curl_easy_init »
/usr/bin/ld : getTempLondon.c:(.text+0x10a) : référence indéfinie vers « curl_easy_setopt »
/usr/bin/ld : getTempLondon.c:(.text+0x116) : référence indéfinie vers « curl_easy_perform »
collect2: error: ld returned 1 exit status
axek@Axek:~/Documents/Cours/L2/methodologie scientifique/getTemperatureV01$
```

Mais on a une erreur ! Lorsque vous utilisez libcurl il faut penser à utiliser l'argument -lcurl à chaque fois que vous compilez votre code (cet oubli d'argument nous aura bien fait galérer).
Et si on le lance on a :



```
axek@Axek:~/Documents/Cours/L2/methodologie scientifique/getTemperatureV01$ gcc getTempLondon.c -lcurl
axek@Axek:~/Documents/Cours/L2/methodologie scientifique/getTemperatureV01$
```

```
axek@Axek: ~/Documents/Cours/L2/methodologie scientifique/getTemperatureV01
axek@Axek:~/Documents/Cours/L2/methodologie scientifique/getTemperatureV01$ ./a.out
{"coord":{"lon":-0.13,"lat":51.51},"weather":[{"id":501,"main":"Rain","description":"moderate rain","icon":"10d"}],"base":"stations","main":{"temp":8.94,"pressure":1012,"humidity":81,"temp_min":7.78,"temp_max":10},"visibility":4900,"wind":{"speed":8.2,"deg":210,"gust":13.9},"clouds":{"all":75},"dt":1575985033,"sys":{"type":1,"id":1414,"country":"GB","sunrise":1575964485,"sunset":1575993108},"timezone":0,"id":2643743,"name":"London","cod":200}axek@Axek:~/Documents/Cours/L2/methodologie scientifique/getTemperatureV01$
```

C'est bien le résultat attendu.

Maintenant que nous savons que libcurl fonctionne bien on peut s'attaquer au code du programme.

La documentation pour plus d'informations : <https://curl.haxx.se/libcurl/c/allfuncs.html>

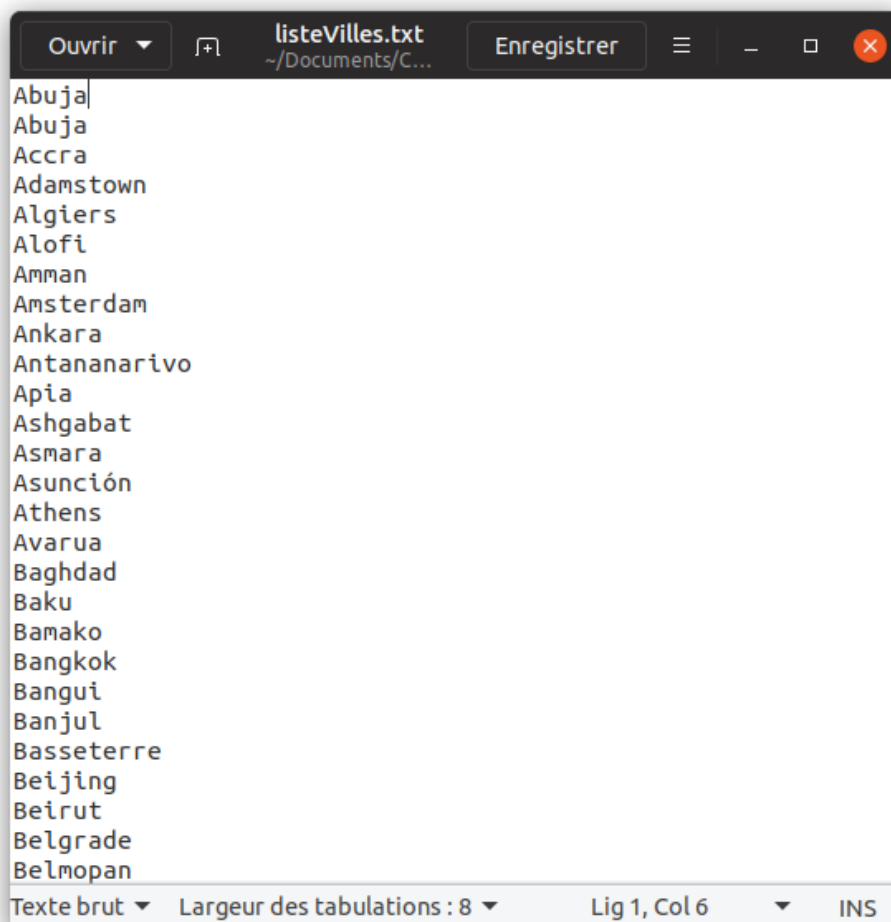
Listes des villes :

Tout d'abord il nous faut un fichier texte dans lequel il y aura la liste de toutes les capitales du monde. Pour faire cela, on est allé sur :

https://fr.wikipedia.org/wiki/Liste_des_capitales_du_monde.

Nous avons copier-coller le tableau dans un fichier texte puis nous avons mis en forme les données afin d'avoir un nom de ville suivi d'un retour à la ligne, et ça pour chaque ville.

Ce qui nous donne :



```
Ouvrir  listeVilles.txt  Enregistrer
~/Documents/C...

Abuja
Abuja
Accra
Adamstown
Algiers
Alofi
Amman
Amsterdam
Ankara
Antananarivo
Apia
Ashgabat
Asmara
Asunción
Athens
Avarua
Baghdad
Baku
Bamako
Bangkok
Bangui
Banjul
Basseterre
Beijing
Beirut
Belgrade
Belmopan

Texte brut  Largeur des tabulations : 8  Lig 1, Col 6  INS
```

Une fois le fichier texte mise en forme il faut l'importer.

```
/*----- Importation des villes -----*/
char villes[200][100];
int nb = 0;
FILE *fp;

fp = fopen("listeVilles.txt", "r");
while(fscanf(fp, "%s\n", villes[nb]) != -1)
    nb++;
fclose(fp);
```

Pour l'importation nous nous sommes servi de :

- fopen() avec comme argument « r » qui permet de lire un fichier
- fscanf() qui permet de récupérer le contenu d'un fichier
- fclose() qui permet de fermer notre fichier listesVilles.txt

Cette partie de code va donc récupérer chaque ligne de notre fichier listeVilles.txt pour les mettre dans un tableau villes.

AllTemp :

Ensuite il faut récupérer les données de chaque ville. Pour cela nous avons d'abord réalisé un programme qui affiche dans la console les données météorologiques d'une ville donnée en argument.

```
#include <curl/curl.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]){
    CURL *curl = curl_easy_init();
    if(curl) {
        char url[500] = "http://api.openweathermap.org/data/2.5/weather?q=";
        strcat(url, argv[1]);
        printf("%s\n", argv[1]);
        strcat(url, "&units=metric&APPID=b7c635aa6d4f700b3ffd7a54f01b4958");
        CURLcode res;
        curl_easy_setopt(curl, CURLOPT_URL, url);
        res = curl_easy_perform(curl);
    }
    return 0;
}
```

Pour récupérer les arguments de notre programme il faut utiliser les paramètres argc et argv. Argc permet de savoir combien d'argument sont en paramètre et argv stock les arguments dans un tableau (sachant qu'en position 0 se trouve le nom du programme).

Pour ce code nous utilisons strcat() qui concatène deux chaînes de caractères. Nous réalisons une première concaténation, elle est composée du début de notre url et notre argument puis nous en réalisons une deuxième qui concatène la partie d'avant avec la fin de notre url. Si nous mettons « Madrid » en argument cela nous donne donc une url sous cette forme :

<http://http://api.openweathermap.org/data/2.5/weather?q=Madrid&units=metric&APPID=b7c635aa6d4f700b3ffd7a54f01b4958>

Lorsque qu'on exécute le programme on obtient bien les données de Madrid :

```
axek@Axek: ~/Documents/Cours/L2/methodologie_scientifique/getTemperatureV03
axek@Axek:~/Documents/Cours/L2/methodologie_scientifique/getTemperatureV03$ ./getTemp Madrid
Madrid
{"coord":{"lon":-3.7,"lat":40.42},"weather":[{"id":800,"main":"Clear","description":"clear sky","icon":"01d"}],"base":"stations","main":{"temp":8.36,"pressure":1027,"humidity":71,"temp_min":6,"temp_max":10,"visibility":7000,"wind":{"speed":0.5,"deg":130},"clouds":{"all":0},"dt":1575989157,"sys":{"type":1,"id":6443,"country":"ES","sunrise":1575962807,"sunset":1575996500,"timezone":3600,"id":3117735,"name":"Madrid","cod":200}}axek@Axek:~/Documents/Cours/L2/methodologie_scientifique/getTemperatureV03$
```

Récupération des données de l'API :

Maintenant nous devons récupérer ses données pour chaque capitale. Nous avons réalisé une boucle qui se terminera uniquement quand nous aurons fini de s'occuper de chacune de nos capitales.

Pour chaque ville on réalise cette commande (grâce à la fonction system() qui permet de réaliser des commande Unix) : ./getTemp villes[compteur] > file.txt
Puis on en récupère le contenu avec une importation fichier.

```
while(compteur < nb)
{
    /*----- Récupération des données de l'api openweather -----*/
    //system("clear");
    strcpy(result, "./getTemp ");
    strcat(result, villes[compteur]);
    strcat(result, " > file.txt");
    system(result);
    /*----- Importation du fichier -----*/
    fp = fopen("file.txt", "r");
    while(feof(fp) == 0)
    {
        fscanf(fp, "%c", &c);
        data[i] = c;
        i++;
    }
    fclose(fp);
    printf("%s\n", data);
}
```

Mise en forme des données :

Ensuite, nous devons traiter l'information pour récupérer uniquement la température, la longitude et la latitude. Nous avons cette partie du code qui se trouve juste après les « récupérations des données de l'API » qui se trouve aussi dans la même boucle.

```
/*----- Tri des données -----*/

/* temperature */
strcpy(ret, strstr(data, "\"temp\":"));
i = 7;
j = 0;
while((ret[i] >= '0' && ret[i] <= '9') || ret[i] == '.' || ret[i] == '-')
{
    temp[j] = ret[i];
    j++;
    i++;
}
printf("la temperature est de %s\n", temp);

/* longitude */
strcpy(ret, strstr(data, "\"lon\":"));
i = 6;
j = 0;
while((ret[i] >= '0' && ret[i] <= '9') || ret[i] == '.' || ret[i] == '-')
{
    lon[j] = ret[i];
    j++;
    i++;
}
printf("la longitude est de %s\n", lon);

/* latitude */
strcpy(ret, strstr(data, "\"lat\":"));
i = 6;
j = 0;
while((ret[i] >= '0' && ret[i] <= '9') || ret[i] == '.' || ret[i] == '-')
{
    lat[j] = ret[i];
    j++;
    i++;
}
printf("la latitude est de %s\n", lat);
```

Pour récupérer uniquement la température, on utilise tout d'abord la fonction strstr() qui va chercher dans nos données « "temp": ». Puis une fois trouvé on stock « "temp": » et tout ce qui se trouve devant dans une variable appelée ret.

Exemple de ce qu'on trouve dans ret avec Madrid :

```
"temp":8.53,"pressure":1027,"humidity":76,"temp_min":6.11,"temp_max":10},"visibility":7000,"wind":{"speed":1},"clouds":{"all":0},"dt":1575988992,"sys":{"type":1,"id":6443,"country":"ES","sunrise":1575962807,"sunset":1575996500},"timezone":3600,"id":3117735,"name":"Madrid","cod":200}
```

Puis on se positionne juste après les deux points de temp en disant que i vaut 6. A l'aide d'une boucle on récupère les chiffres, les points et les signes négatif s'ils sont présents. Cela nous permet donc de récupérer la température.

Et pour finir le traitement des données on réalise la même chose pour la longitude et la latitude.

Remarque :

Pour le traitement des données nous avons eu un bug dont nous n'avons pas compris la cause, pendant une longue période le programme mettait des caractères spéciaux devant les valeurs temp, lon et lat. Le problème c'est résolu tout seul sans que nous comprenions comment.

Exportation des données :

Nous sommes maintenant à l'étape où il faut exporter nos données ci-dessous vous pouvez voir le code de l'exportation qui se trouve lui aussi dans notre boucle.

```
/*----- Exportation des données -----*/
fp = fopen("allTemp.txt", "a");
fprintf(fp, "%s;%s;%s;%s\n", villes[compteur], lat, lon, temp);
fclose(fp);

memset(temp, 0, sizeof(temp));
memset(lon, 0, sizeof(lon));
memset(lat, 0, sizeof(lat));
i = 0;
j = 0;
compteur++;
}
```

Comme vous pouvez le voir on fopen() mais on met en paramètre « a », qui permet de créer le fichier s'il n'existe pas il sera créé et s'il existe on ajoute du texte.

Et on utilise fprintf() qui permet d'écrire du contenu dans un fichier.

Nos données sont exportées sous cette forme : ville;latitude;longitude;température
Puis nous fermons le fichier.

A la fin de la boucle on réinitialise temp, lon et lat, on remet à 0 i et j, et on passe à la ville suivante.

Toutes ses données sont stockées dans un fichier appelé allTemp.txt

Changement du nom du fichier :

Nous sommes désormais à la dernière étape, cette étape consiste à renommer le fichier allTemp.txt en un fichier date+time.txt. Exemple, on exécute le programme il se termine le 10 décembre 2019 à 17:42:09, il s'appelle donc « 121019174209.txt » 12 étant le mois, 10 le jour, 19 l'année, 17 l'heure, 42 les minutes, 09 les secondes.

Voici le code :

```
/*----- Nommage du fichier allTemp.txt -----*/
system("date +%D > date.txt");
system("date +%T >> date.txt");
fp = fopen("date.txt", "r");
fscanf(fp, "%s", date);
fscanf(fp, "%s", time);
fclose(fp);
strcat(date, time);
i = 0;
j = 0;
while(i < strlen(date)){
    if(date[i] >= '0' && date[i] <= '9')
    {
        nameFile[j] = date[i];
        printf("%c ", nameFile[j]);
        j++;
    }
    i++;
}
strcat(nameFile, ".txt");
strcat(command, nameFile);
system(command);
system("rm file.txt date.txt");
return 0;
}
```

Tout d'abord on stock la date dans un fichier date.txt sous cette forme mm/jj/aa (mm étant le mois, jj le jour et aa l'année) avec un retour à la ligne, puis on ajoute l'heure dans date.txt sous cette forme hh:mn:ss (hh étant l'heure, mn étant les minutes et ss étant les secondes).

Après on récupère le contenu de date.txt pour le mettre dans une variable appelé date de type chaîne de caractères. On a donc dans variable quelques chose de ce type

« mm/jj/aahh:mn:ss ».

Et pour terminer on traite la variable date pour récupérer uniquement les chiffres et on met le contenu une variable appelé nameFile. Si on reprend l'exemple plus haut on se retrouve avec comme contenu dans la variable « 121019174209 » puis on utilise un strcat() pour ajouter un « .txt » et un autre strcat() pour concaténer « mv allTemp.txt » et

« mmjjaahhmss.txt » cette commande est ensuite lancer ce qui va avoir pour effet de renommer allTemp.txt dans le format voulu. Et avant la fin du programme les fichiers inutiles sont supprimés.

4.1.2. Exploitation et traitement des données Absolute

C'est à la suite d'heures de recherches que nous sommes tombés sur des données intéressantes : "absolute.nc" du CRU

Absolute	<u>NetCDF</u> (1MB)
Absolute temperatures for the base period 1961-90 on a 5° by 5° grid (Jones et al., 1999). Note that in this file, latitudes run from North to South.	

<https://crudata.uea.ac.uk/cru/data/temperature/>

Note importante : Des difficultés ont été rencontrées lors du téléchargement de ces données. Effectivement nous nous sommes aperçus que la donnée téléchargée était sous le format NetCDF (format que nous ne connaissions pas).

Nous avons donc obtenu le fichier absolute.nc, fichier avec une extension .nc (pour NetCDF) que nous ne savions exploiter. Il a fallu faire des recherches pour trouver des logiciels qui permettent d'exploiter ce type de données et de savoir comment les utiliser.

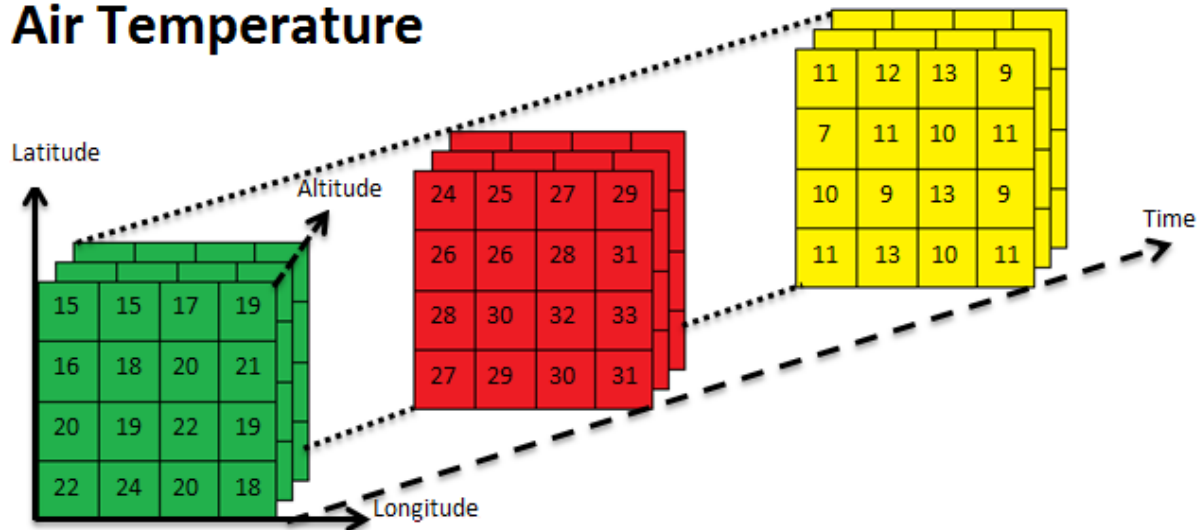
Il a donc été nécessaire de se documenter sur cette technologie voici un court récapitulatif de cette découverte :

NetCDF (Network Common Data Form) est constitué, d'une part, d'un ensemble de bibliothèques logicielles et d'autre part, d'un format de données « auto-documenté », indépendant de l'architecture matérielle qui permet la création, l'accès et le partage de données scientifiques stockées sous la forme de tableaux.

Pour plus d'info : <https://fr.wikipedia.org/wiki/NetCDF>

Mais pour faire plus simple il s'agit d'un cube de données voici comment les données sont représentées donc notre cas :

Air Temperature

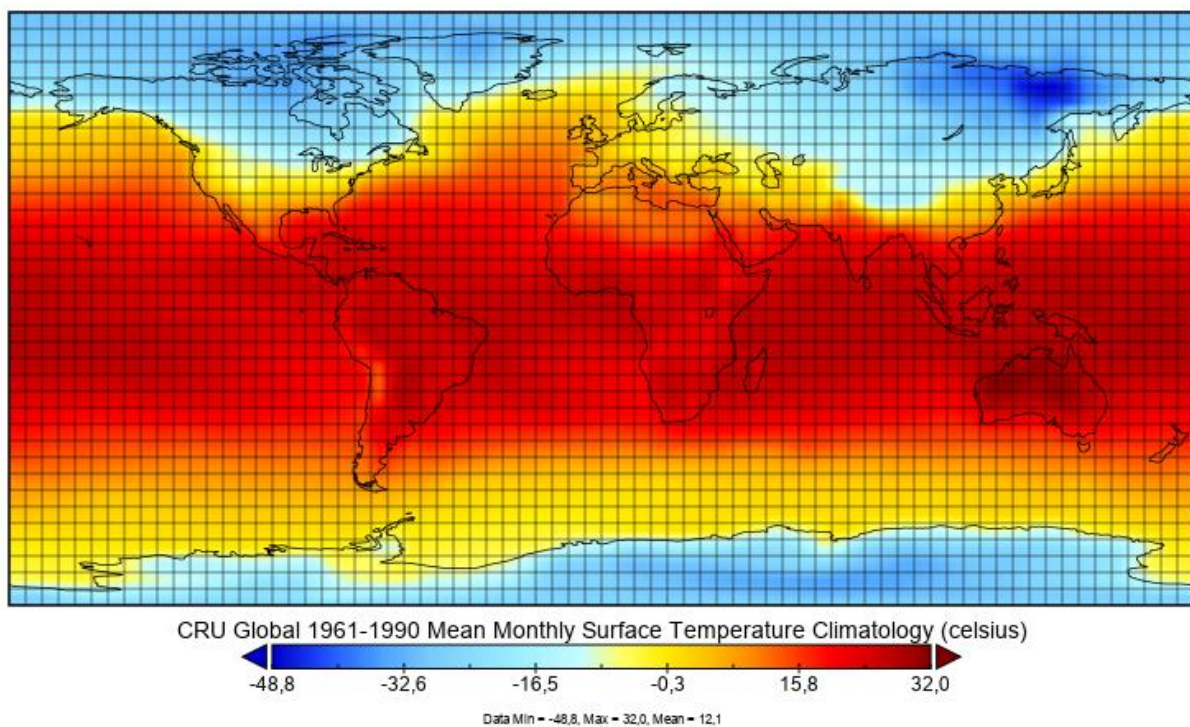


absolute.nc va donc être constitué des moyennes des température à des coordonnées précises au fil des mois pendant une période allant de 1961 à 1990.

Les coordonnées sont définies avec une longitude et une latitude sur une grille allant de 5° en 5°.

Il est possible d'exploiter et de visualiser ces données en utilisant un logiciel développé par la NASA, il s'agit de : "Panoply". C'est grâce à ce logiciel que nous avons pu extraire les données de absolute.nc sous forme de fichier texte (pour pouvoir les exploiter pour réaliser notre expérience). Et de profiter au passage pour visualiser ces données :

CRU Global 1961-1990 Mean Monthly Surface Temperature Climatology



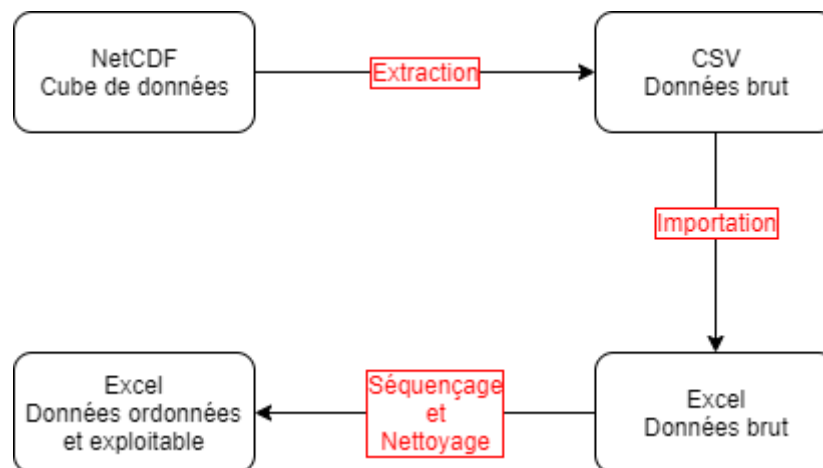
Ce relevé de température correspond au mois de Janvier et nous avons la température à chaque point de la grille.

Passage de NetCDF à Excel :

Pour réaliser notre expérience nous avons décidé d'utiliser Excel car c'est un logiciel à notre portée du fait de sa simplicité d'utilisation.

Nous avons donc extrait les données de absolute.nc sous format csv pour les mettre en forme sur Excel. Un travail pénible car nous avons une grande quantité de données et il a fallu les séquencer et les nettoyer pour les mettre en forme.

Pour réaliser cette tâche nous avons effectué le processus suivant :



A ce stade nous avons théoriquement la moyenne des températures dans toutes les villes du monde ainsi qu'une feuille Excel qui nous fournit une moyenne de température entre 1961 et 1990 au fil des mois à des points de coordonnées précis.

Mais une **nouvelle problématique** fait son apparition :

Comment comparer des températures qui n'ont pas les mêmes coordonnées ?

- Temp = moyenne des températures mensuel dans les villes du monde (actuel).
- Temp-1 = moyenne des températures mensuel dans les villes du monde entre 1961 et 1990

C'est techniquement impossible ! Je m'explique, une ville à des coordonnées précise et une certaine température à un instant t. Pour savoir si la température en ce point a augmenté à cause du réchauffement climatique au fil du temps. Je dois comparer la moyenne actuelle (**temp**) avec celle du passé (**temp-1**). Or il n'est pas possible de connaître temp-1 car ces données ne sont pas disponibles sur internet et certaines villes actuelles n'existaient pas à l'époque ou n'étaient pas équipées capteur de température.

Il a donc été nécessaire d'approximer à l'aide des données de absolute.nc la valeur de temp-1 pour chaque ville.

Exemple :

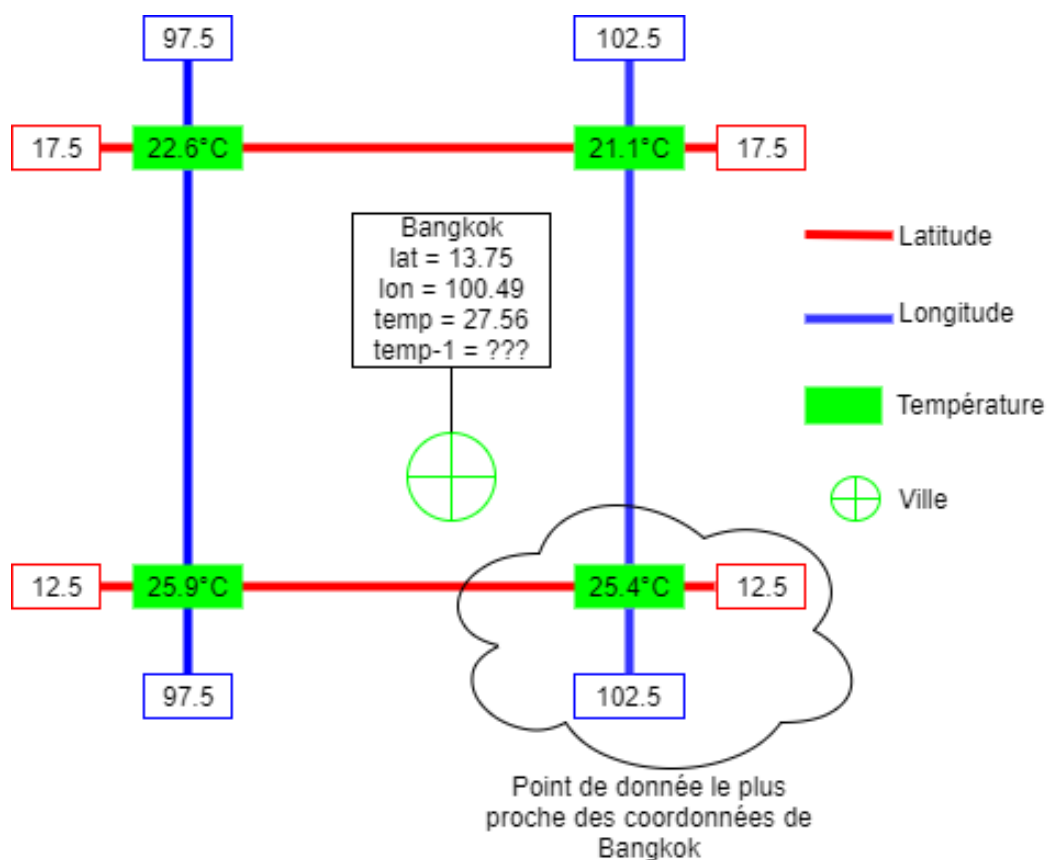
A l'aide de l'API voici les données que nous récupérons :

19	Bangkok	13,75	100,49	27,56
----	---------	-------	--------	-------

Dans l'ordre : La ville, la latitude, la longitude et la température actuelle. Ces données sont très précises

Or ces coordonnées ne sont pas disponibles dans ma base de données absolue donc je n'ai pas la valeur de temp-1 pour faire mon analyse de comparaison.

Voici la représentation du problème :



Il est donc nécessaire de calculer une approximation de la valeur de temp-1 (la température moyenne de Bangkok entre 1961 et 1990 durant le mois de janvier).

Solution proposée :

Il serait techniquement possible avec de solides connaissances en mathématique de faire un lissage linéaire pour déterminer une approximation linéaire de temp-1 en chaque point de coordonnée. Mais faute de temps et d'expertise une solution plus simple est proposé.

De plus, aucune théorie n'affirme que la température fluctue de manière linéaire, donc la marge d'erreur est potentiellement identique pour les 2 méthode.

Notre solution consiste à nous référer à chaque point de donnée le plus proche de la ville pour donner une approximation de sa température moyenne de l'époque.

Dans l'exemple vu plus haut le point de donnée le plus proche de Bangkok est le point de coordonnée **X(lat=12.5, lon=102.5, temp-1=25.4)**. Nous pouvons donc approximer la valeur temp-1 à 25.4°C avec une erreur correspondant à la variation de température moyenne en fonction de la distance d entre les deux points. Cependant cette erreur est infime.

Il a donc été nécessaire d'implémenter un algorithme capable de faire matcher les coordonnées de chaque ville avec le point de donnée le plus proche de celui-ci

Une macro Excel en VBA a été développée pour réaliser cette tâche qui serait très pénible de faire à la main.

Le code est disponible en **Annexe**.

Voici un résultat type que l'algorithme va permettre de définir :

19	Bangkok	13,75	100,49	27,56	25,40
----	---------	-------	--------	-------	-------

Ici, on s'aperçoit que l'algorithme permet de définir temp-1 de Bangkok. De plus la valeur est en adéquation avec l'exemple vu plus haut, il n'y a pas d'erreur.

Notre algorithme va calculer temp-1 pour chaque ville présente dans la feuille Excel et s'arrêter automatiquement dès qu'il n'y en aura plus.

Observation :

Dans le cas de Bangkok, nous observons qu'il y fait moyennement plus chaud aujourd'hui qu'entre 1961 et 1990.

Toutes les pièces sont maintenant rassemblées pour réaliser une analyse de données qui nous permettra de confirmer ou non notre hypothèse.

5. Analyse des résultats et Conclusion

1^{er} Hypothèse :

Pour répondre notre hypothèse qui était de savoir si la température à la surface du globe augmentait au fil des ans nous pouvons maintenant, comparé la moyenne des températures dans nos villes.

A l'aide de la fonction MOYENNE() d'Excel, nous avons calculer la moyenne de temp et de temp-1.

Pour temp :

```
=MOYENNE(D2:D190)
```

14,947

La moyenne de température actuelle est 14.947°C

Pour temp-1 :

```
=MOYENNE(E2:E190)
```

13,63

La moyenne de température entre 1961 et 1990 était 13.63°C

Si nous calculons la différence de température :

$14.947 - 13.63 = 1.317$ soit une augmentation d'environ 1.3°C pendant un période de 45ans.

Si nous calculons le taux d'accroissement :

$((14.947 - 13.63) / 13.63) * 100 = 9,66\%$

Soit une augmentation d'environ 10% de la température à la surface du globe en 45 ans.

C'est une augmentation importante et alarmante de la température à la surface du globe.

Cette analyse de données nous permet de confirmer notre hypothèse : Il y a bien un réchauffement climatique.

2ème Hypothèse :

La banquise arctique fond de plus en plus vite.

Nous pouvons confirmer cette hypothèse en appliquant une certaine logique. Comme la première hypothèse a été vérifiée et qu'il fait de plus en plus chaud sur la terre (accroissement de la température de 10%). Nous savons que, plus il fait chaud plus la glace a tendance à fondre. Donc effectivement, la banquise arctique fond de plus en plus vite.

3ème Hypothèse :

L'ours polaire peut trouver un nouvel environnement pour vivre.

Après des recherches sur internet nous avons défini un seuil : L'ours polaire a besoin de vivre dans un environnement avec une température moyenne de 0°C.

Nous avons donc questionné notre analyse de donnée avec la requête suivante :

```
=SI(D2<0;"ours oui";"ours non")
```

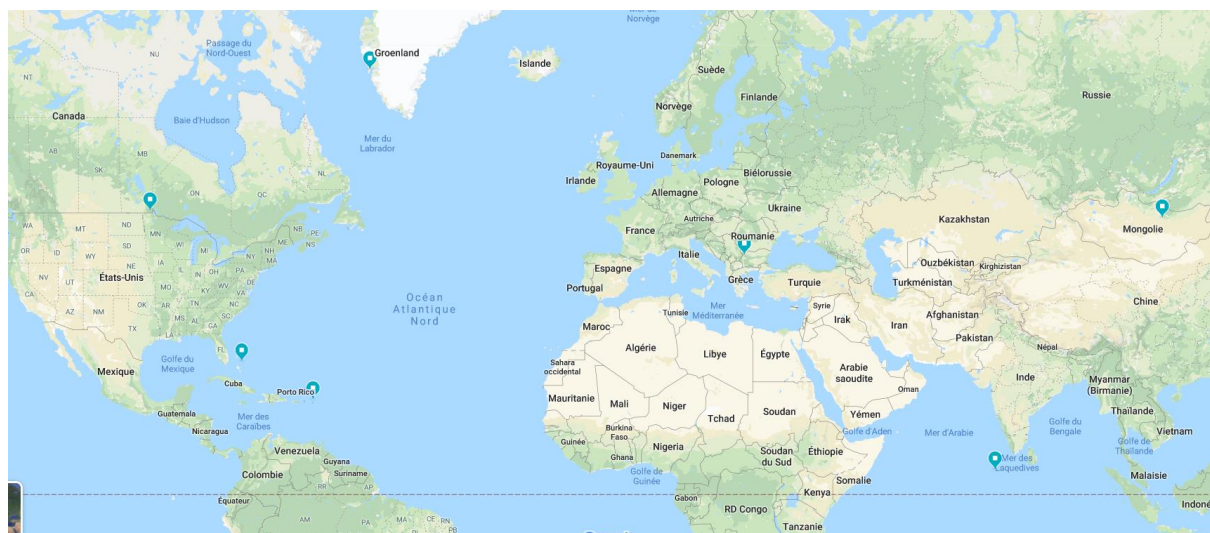
Si la température moyenne est inférieure à 0°C alors, créer une variable "ours oui" sinon créer une variable "ours non"

Ces variables vont permettre de définir les zone où l'ours peut vivre

Grace à une fonction de tri nous pouvons récupérer ces lieux :

Ville accueillant les ours						
Malé	46,35	10,91	-0,77	7,50	-8,27	ours oui
Montevideo	44,94	-95,72	-5,49	-8,50	3,01	ours oui
Nassau	46,81	15,34	-1,47	4,60	-6,07	ours oui
Nuuk	64,17	-51,74	-6,06	-10,50	4,44	ours oui
Philipsburg	46,33	-113,29	-3	-6,00	3,00	ours oui
Roseau	48,85	-95,76	-15,54	-15,00	-0,54	ours oui
Sofia	42,7	23,32	-1,4	2,30	-3,70	ours oui
Ulaanbaatar	47,92	106,92	-10	-21,50	11,50	ours oui

Voici les zones, il est possible de définir un certain périmètre autour de ces points car la température ne devrait pas fluctuer de manière brusque.



Annexes

Code de getTemp.c :

```
#include <curl/curl.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]){
    CURL *curl = curl_easy_init();
    if(curl) {
        char url[500] =
"http://api.openweathermap.org/data/2.5/weather?q=";
        strcat(url, argv[1]);
        printf("%s\n", argv[1]);
        strcat(url, "&units=metric&APPID=b7c635aa6d4f700b3ffd7a54f01b4958");
        CURLcode res;
        curl_easy_setopt(curl, CURLOPT_URL, url);
        res = curl_easy_perform(curl);
    }
    return 0;
}
```

Code de main.c :

```
#include <stdio.h>

#include <string.h>

#include <stdlib.h>

int main(){

    /*----- Déclaration des variables -----*/

    char villes[200][100];

    int nb = 0;

    int compteur = 0;

    FILE *fp;

    char data[1000];

    char c;

    int i = 0;

    int j = 0;

    char ret[500];

    char temp[10];

    char lon[10];

    char lat[10];

    char result[500];

    char nameFile[50];

    char date[50];

    char time[50];

    char command[100] = "mv allTemp.txt ";

    /*----- Importation des villes -----*/
```

```
fp = fopen("listeVilles.txt", "r");
while(fscanf(fp, "%s\n", villes[nb]) != -1)
    nb++;
fclose(fp);
while(compteur < nb)
{
    /*----- Récupération des données de l'api openweather -----*/
    //system("clear");
    strcpy(result, "./getTemp ");
    strcat(result, villes[compteur]);
    strcat(result, " > file.txt");
    system(result);
    /*----- Importation du fichier -----*/
    fp = fopen("file.txt", "r");
    while(!feof(fp))
    {
        fscanf(fp, "%c", &c);
        data[i] = c;
        i++;
    }
    fclose(fp);
    printf("%s\n", data);

    /*----- Tri des données -----*/

    /* temperature */
    strcpy(ret, strstr(data, "\"temp\":"));
    i = 7;
    j = 0;
    while((ret[i] >= '0' && ret[i] <= '9') || ret[i] == '.' || ret[i] == '-')
        j++;
    ret[j] = '\0';
}
```

```
{
    temp[j] = ret[i];
    j++;
    i++;
}
printf("la temperature est de %s\n", temp);

/* longitude */
strcpy(ret, strstr(data, "\"lon\":"));
i = 6;
j = 0;
while((ret[i] >= '0' && ret[i] <= '9') || ret[i] == '.' || ret[i] == '-')
{
    lon[j] = ret[i];
    j++;
    i++;
}
printf("la longitude est de %s\n", lon);

/* latitude */
strcpy(ret, strstr(data, "\"lat\":"));
i = 6;
j = 0;
while((ret[i] >= '0' && ret[i] <= '9') || ret[i] == '.' || ret[i] == '-')
{
    lat[j] = ret[i];
    j++;
    i++;
}
printf("la latitude est de %s\n", lat);
```

```
/*----- Exportation des données -----*/

fp = fopen("allTemp.txt", "a");
fprintf(fp, "%s;%s;%s;%s\n", villes[compteur], lat, lon, temp);
fclose(fp);

memset(temp, 0, sizeof(temp));
memset(lon, 0, sizeof(temp));
memset(lat, 0, sizeof(temp));

i = 0;
j = 0;
compteur++;

}

/*----- Nommeage du fichier allTemp.txt -----*/

system("date +%D > date.txt");
system("date +%T >> date.txt");
fp = fopen("date.txt", "r");
fscanf(fp, "%s", date);
fscanf(fp, "%s", time);
fclose(fp);
strcat(date, time);

i = 0;
j = 0;
while(i < strlen(date)){
    if(date[i] >= '0' && date[i] <= '9')
    {
        nameFile[j] = date[i];
        j++;
    }
    i++;
}
```

```
    strcat(nameFile, ".txt");  
    strcat(command, nameFile);  
    system(command);  
    system("rm file.txt date.txt");  
    strcpy(command, "gedit ");  
    strcat(command, nameFile);  
    system(command);  
    return 0;  
}
```


Macro Excel (VBA) temp :

```
Sub temp()  
Dim longitude As Single  
Dim latitude As Single  
Dim ligne As Integer  
Dim temp As Single  
Dim ligne1 As Integer  
Dim column1 As Integer  
ligne = 2  
Sheets("tempVille").Select  
Do While (Not IsEmpty(Cells(ligne, 1)))  
    column1 = 3  
    ligne1 = 3  
  
    Sheets("tempVille").Select  
    latitude = Cells(ligne, 2)  
    longitude = Cells(ligne, 3)  
    Sheets("Janvier").Select  
    Do While (longitude > Cells(2, column1))  
        Do While (latitude < Cells(ligne1, 2))  
            ligne1 = ligne1 + 1  
        Loop  
        column1 = column1 + 1  
    Loop  
    temp = Cells(ligne1, column1)  
    Sheets("tempVille").Select  
    Cells(ligne, 5) = temp  
    ligne = ligne + 1  
Loop  
End Sub
```