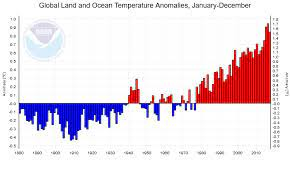
# SAE S2.02 : Exploration algorithmique

Il s’agit dans cette SAE d’essayer de constater le phénomène de réchauffement climatique à partir des données recueillies lors de votre précédente SAE. Afin d’exploiter au mieux ces données qui sont considérables en quantité, une étude sur la représentation de plusieurs types de structures de données sera préalablement initiée. L’objectif est de déterminer quelle structure semble être la plus adaptée pour répondre efficacement à un certain nombre de questions comme :

* Le réchauffement climatique est-il global ?
* Le réchauffement est-il similaire dans toutes les stations ?
* Est-ce que ce réchauffement est le même sur tous les mois ou faut-il les différencier ? Est-ce que la saison est importante ?
* Y-a-t-il des années exceptionnelles ? L’interprétation des résultats est-elle la même si on ne prend pas en compte ces années exceptionnelles ?
* …



Vous devrez bien entendu rendre à la fois votre solution Visual épurée des fichiers non nécessaires ainsi qu’un compte-rendu le plus complet possible avec en première page la liste de toutes les fonctionnalités que vous aurez eu le temps d’implémenter. Ensuite sur les pages suivantes vous répondrez aux questions demandées au fur à mesure du déroulé de ce sujet. Enfin dans une dernière partie vous exposerez les principales difficultés rencontrées et la conclusion à votre travail d’analyse des données.

## Partie 1 : Étude de structures de données : listes chaînées, arbres binaires

1. Préambule

Pour chacune des fonctions membres de vos structures de données, il est demandé d’évaluer (sans démonstration) la complexité de celles-ci. Vous indiquerez celles-ci en commentaires dans les fichiers d’entêtes comme dans les 2 premiers exemples donnés pour *Liste.h*. Bien évidemment, vous compléterez ces estimations de complexités au fur et à mesure de votre implémentation.

On vous rappelle les principales classes de complexité que nous rencontrons en informatique :

O(1) : complexité constante ne dépendant pas de n (quelques opérations)

O(n) : n opérations à un facteur multiplicatif près ( ainsi 0(n)=O(3n)=O(n/2)… ) (typiquement une boucle sur les n éléments)

O(log(n)) : lorsque la taille n du problème est divisée par 2 à chaque étape (par exemple quand vous jouez au jeu de la recherche d’un nombre entre 1 et 1000 et que l’on vous répond plus petit ou plus grand : vous proposez 500 pour éliminer la moitié des nombres…)

O(n²) : typiquement une boucle imbriquée dans une autre (exemples : tris simples)

O(n log(n)) : (exemples : tris performants)

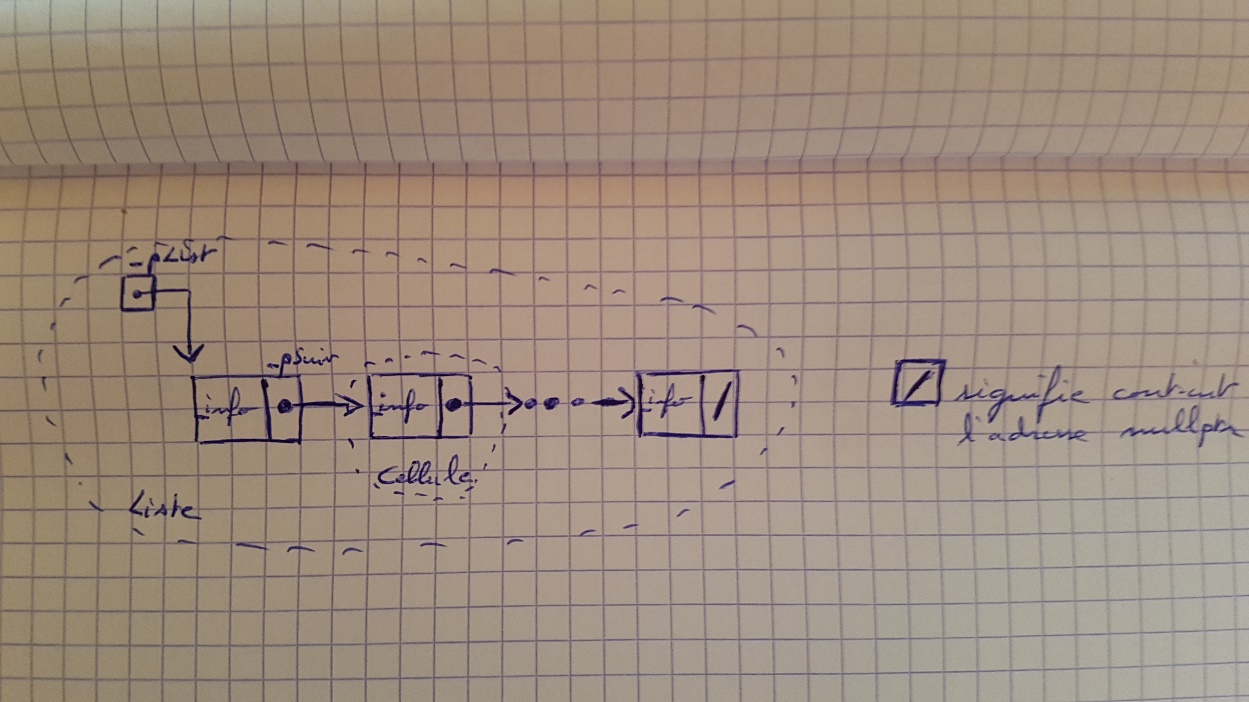
Lorsque n devient grand, on a (<< veut dire « très inférieur ») :

**O(1) << O(log(n)) << O(n) << O(n log(n)) << O(n²) …**

1. Structures de données
   1. Liste chaînée simple

*La classe Liste vous est fournie. Seule la fonction membre ajoute\_cellule\_en\_queue* est encore à implémenter par vos soins. Il est important de comprendre la structure abstraite d’une liste et les schémas qui suivent sont là pour vous aider et doivent être mis en parallèle avec les lignes de code associées, en particulier pour la suppression.

* Représentation schématisée d’une liste chaînée simple



* Suppression dans une liste chaînée simple

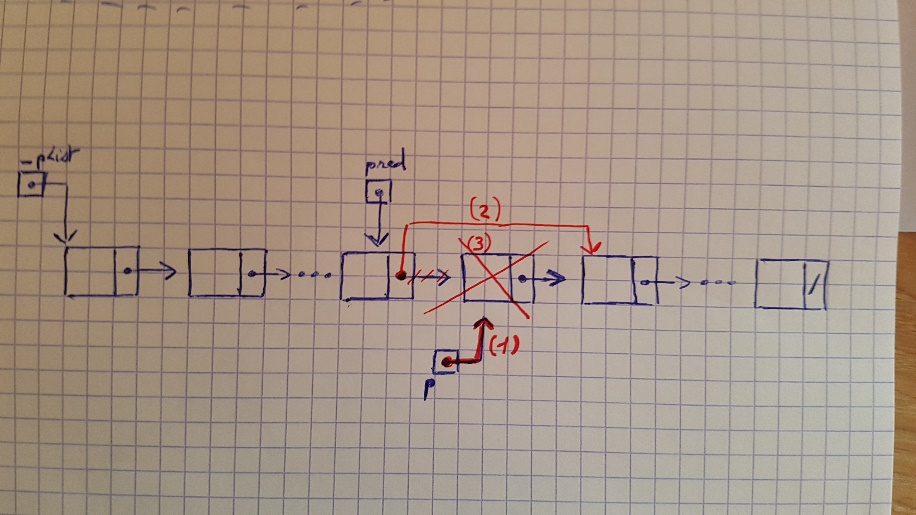


Figure 1 : Suppression cas général

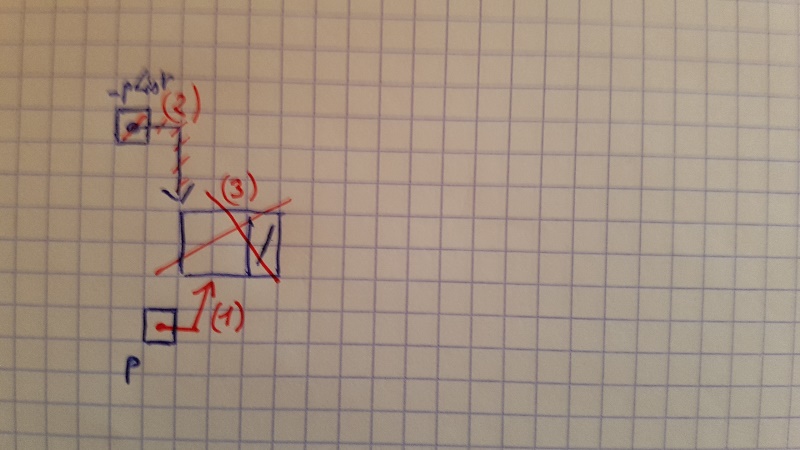


Figure 2 : cas particulier : suppression en tête de liste

* 1. Liste chaînée bi-directionnelle
* Représentation schématisée d’une liste chaînée bi-directionnelle

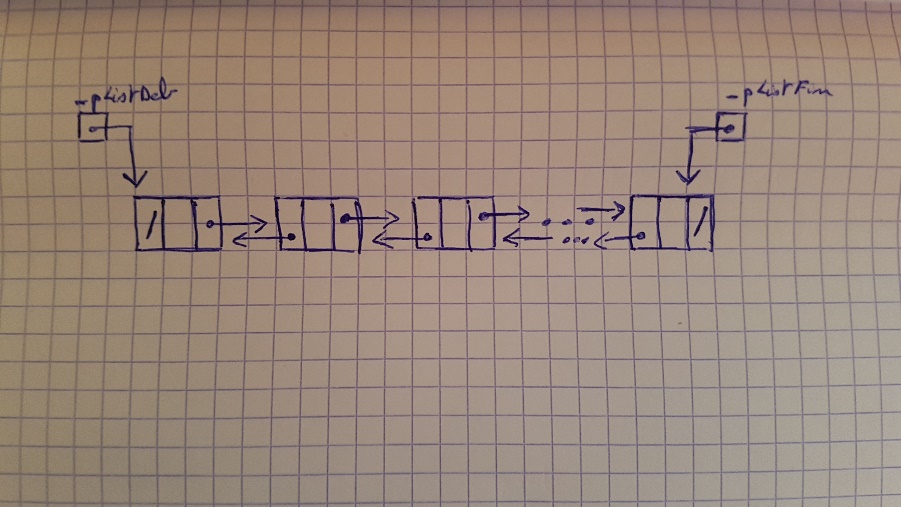


Figure 3: Liste bi-directionnelle

En vous inspirant de la classe *Liste* , implémenter les fonctions membres de la classe ListeB (liste bi-directionnelle). Faire les schémas comme sur l’exemple de la suppression (fig 1 et 2) pour les fonctions *ajoute\_cellule\_en\_tete, ajoute\_cellule\_en queue et supprime\_cellule*.

c. Arbre binaire

* Représentation schématisée d’un arbre binaire

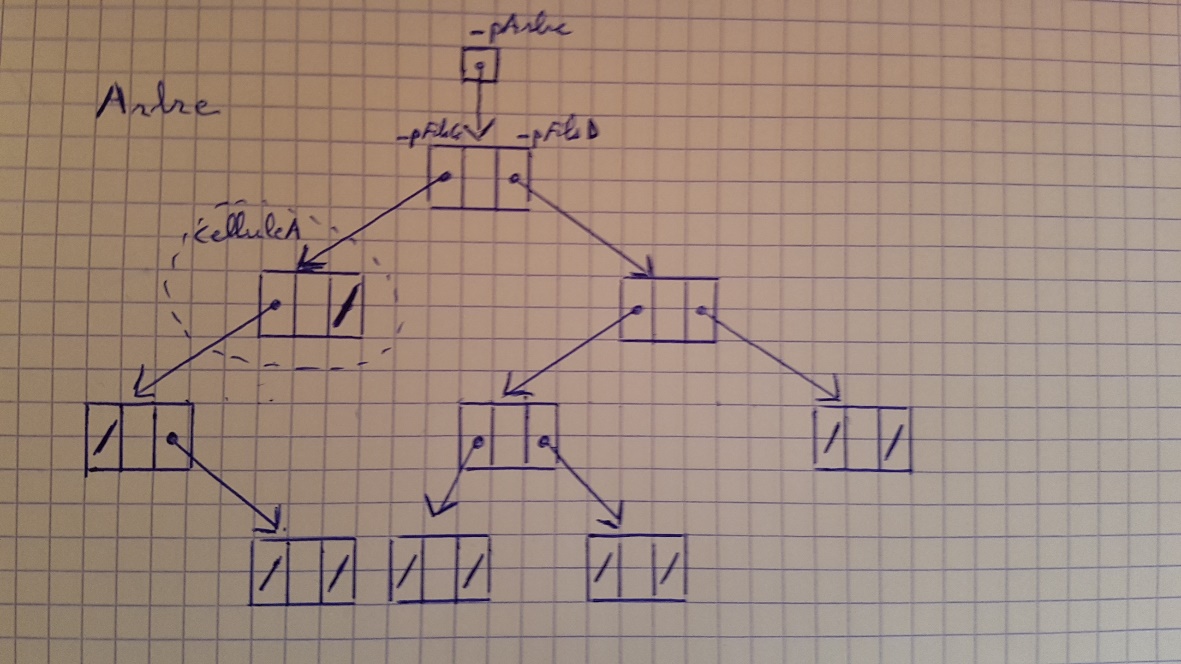


Figure 4 : Un exemple d'arbre

La classe Arbre représentera des arbres binaires de recherche (ABR). On commencera par comprendre à l’aide d’internet ce qu’est un ABR, un parcours préfixé, infixé, postfixé et on implémentera ensuite les fonctions membres non déjà implémentées (n’oubliez pas d’évaluer les complexités : cf Préambule).

La fonction membre donnée ajoute\_cellule est à commenter.

Certains pseudo-algorithmes ou indications sont donnés ci-après pour vous aider. Les fonctions récursives sont souvent utilisées avec les arbres car souvent plus faciles à écrire…

Fonction *nbelem* :

*Si arbre vide alors retourner 0*

*Sinon retourner 1 + nbelem(arbregauche)+nbelem(arbredroit)*

Fonction *nbfeuille* :

*Un nœud est une feuille si celui-ci n’a ni fils\_gauche, ni fils\_droit.*

Fonction *hauteur* :

*La hauteur d’un arbre vide vaut 0*

*sinon elle vaut 1 + max(hauteur(arbregauche), hauteur(arbredroit))*

Vous écrirez bien évidemment un programme principal permettant de tester les fonctions et structures de données. N’oubliez pas de remplir votre compte-rendu avec pour chaque structure de données les complexités de toutes les fonctions.

Si vous ne réussissez pas à implémenter la liste doublement chaînée rapidement, consacrez vos efforts sur l’arbre çà rapportera plus de points.

## Partie 2 : Exploitation de la base de données

Bien évidemment l’attribut *\_info* des structures vues dans la 1ère partie n’est pas restreint à être un entier et peut être de type quelconque… Seule est nécessaire une relation d’ordre pour pouvoir exploiter les ABR.

Utilisez maintenant le squelette de la solution de la 2ième partie dans laquelle vous exploiterez la base de données météo fournie. L’implémentation de la structure de données suit le schéma suivant :

struct CelluleStation

{

int \_ID; // numero d’identification de la station

std::array<Arbre, 12> tab\_arbres; // 1 arbre binaire de recherche par mois de l’annee

CelluleStation\* \_pSuiv;

CelluleStation\* \_pPred;

} ;

struct CelluleArbre {

float \_deltaT; //deltaT entre \_annee et \_annee+1 (T\_annee+1 - T\_annee) pour le mois considere

int \_annee; // année du mois en cours

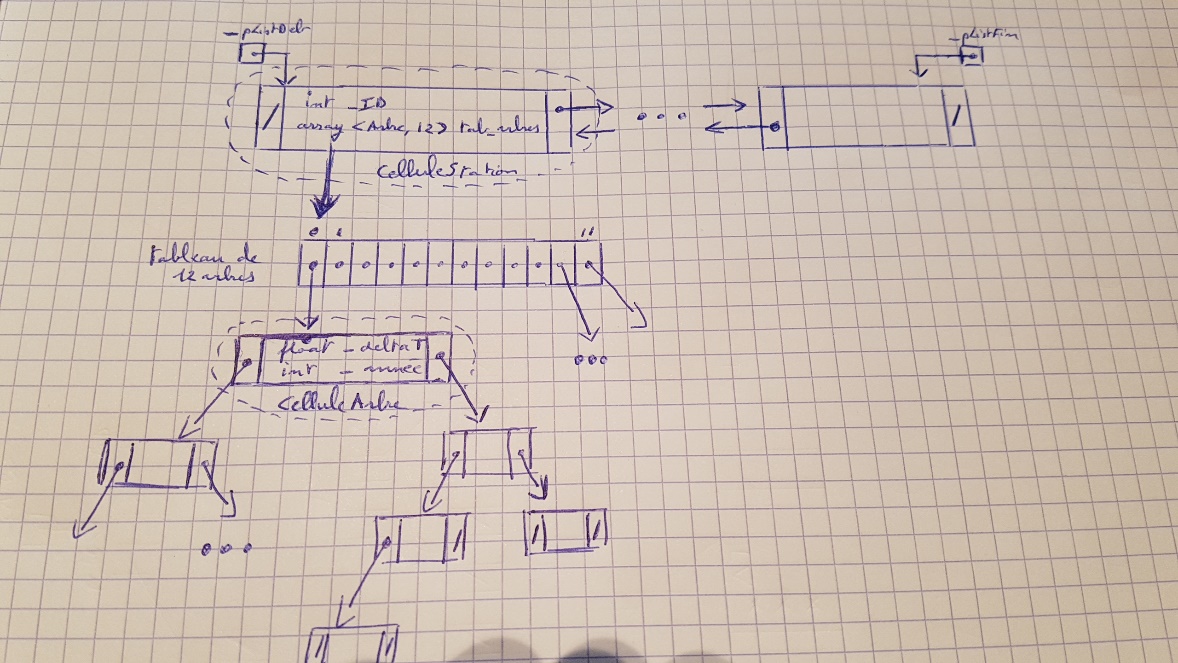
float \_tempMois; // température moyenne du mois en cours

float \_tSigmaMois; // écart type moyen de la température du mois en cours

CelluleArbre\* \_pFilsG;

CelluleArbre\* \_pFilsD;

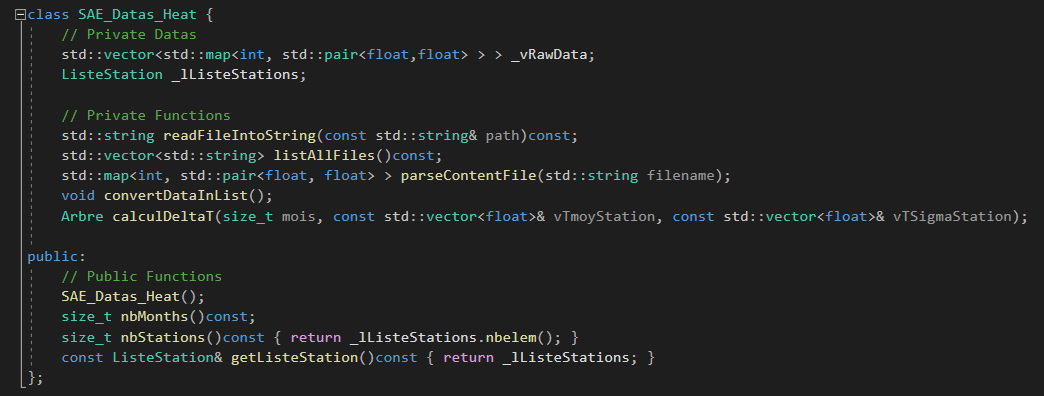
};



Chaque arbre sera un ABR ordonné par rapport à la clé *\_deltaT*, qui est la différence de température entre 2 années consécutives pour un mois fixé.

En mathématique ce deltaT va correspondre à une approximation de la dérivée. Et donc si une dérivée est positive et croissante la fonction va elle-même croître de plus en plus vite.

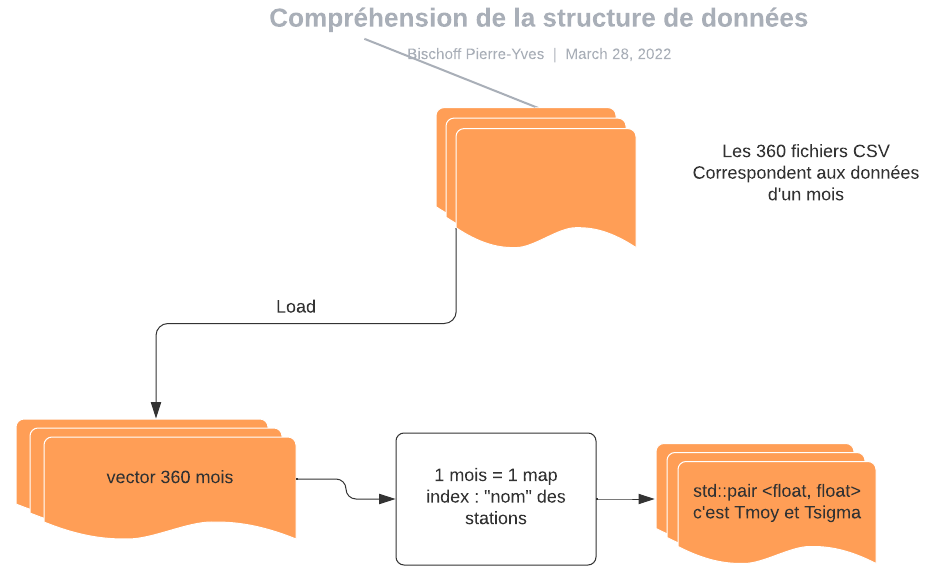
## Partie 2-1 : Construction de la structure arborescente :

Remarque : si vous n’avez pas réussi à implémenter correctement la liste bi-directionnelle dans la partie 1, vous pourrez utiliser une liste chaînée simple à la place. Le fichier d’entête de la classe SAE\_Data contiendra au minimum ceci :

Il s’agit maintenant de remplir cette structure. La fonction parseContentFile(std::string filename) permet de lire les données d’un fichiers CSV de la SAE 2.4. Chaque fichier correspond à un mois de données. La première ligne contient les entêtes des colonnes que la fonction enlève. Les autres lignes contiennent les données proprement dites.

Vous aurez donc un vecteur de 360 éléments ( std::vector<std::pair<float,float>> ) contenant toutes les données pour tous les mois (nous ne nous intéresserons aux températures moyennes du mois et à l’écart-type fournis dans la paire). Ensuite un élément est représenté en C++ par une map indexée par un index correspondant à l’ID de la station (un int par exemple 7005) et pour chaque index vous aurez un vecteur de paires contenant les données.

Voici un schéma je l’espère plus explicite :



Les fonctions de remplissage de la structure sont commencées, la base est épurée et ne contient plus que 21 stations complètes. Vous devez parcourir ces stations pour remplir la liste bi-directionnelle. Vous aurez à implémenter la fonction remplirListe() et la fonction calculDeltaT() qui construit les arbres avec les valeurs de deltaT. Je vous rappelle que le vecteur vStationTmoy contient les données des 360 mois et que pour trouver l’écart de température entre 2 années il suffit de faire la différence entre l’indice i+12 et l’indice i (Attention de ne pas sortir du vecteur).

## Partie 2.2 : exploitation de la structure arborescente :

Maintenant que votre structure de données est remplie, vous allez pouvoir définir vos propres fonctions d’exploitation de la base dans *Traitement.cpp* afin d’en tirer un maximum d’informations. Vous expliquerez en commentaires ce que sont censés faire chacune de vos fonctions, et les résultats attendus. N’oubliez pas de remplir le compte-rendu avec des exemples illustrés !!!

Voici quelques exemples de questions pouvant être traitées, et donc implémentez les fonctions correspondantes (pas forcément dans l’ordre) :

Q1 : Quelle est la température moyenne de l’ensemble des stations sur un mois donné ?

Q2 : Quelle indication vous donne la somme des deltaT (sur un arbre, sur une station, sur toute la base) ?

Q3 : Les stations météos sont placées sur tout le territoire français pas forcément la France métropolitaine. En calculant le réchauffement par station et en excluant les valeurs un peu extrêmes que l’on attribuera aux DOM-TOM, calculez le réchauffement en France métropolitaine.

Q4 : En utilisant que les données de la France métropolitaine calculez le réchauffement moyen pour chaque mois.

Q5 : En météorologie on s’intéresse beaucoup aux saisons : hiver (décembre année précédente, janvier, février) ; printemps (mars, avril, mai) ; … Calculez le réchauffement pour chaque saison et concluez sur le caractère régulier sur l’année du réchauffement climatique. (vous pouvez supposer que le mois de décembre 1991 qui manque est égal au mois de décembre 1992).

Q6 : Quelles indications donnent les deltaT négatifs ?

Q7 : Vous pourrez être amené à vous demander des choses comme : que traduit le fait que, lors d’un parcours infixé de mon arbre de recherche, l’affichage des années correspondantes se retrouvent en ordre croissant (ou quasi croissant) ? (etc…)

Q8 : On parle parfois en météorologie de fonction en crosse de hockey. Quelle est la définition mathématique d’une telle fonction. Quel est le rapport avec nos deltaT ?

Q9 : De manière analogue essayez d’exploiter l’attribut tSigma des arbres.

…

Pour l’interprétation des résultats il faut savoir qu’en météorologie on travaille au minimum sur des durées de 10 ans pour atténuer les variations annuelles qui sont assez importantes. Il nous semble nécessaire d’utiliser un tableau intermédiaire (indexé suivant les années) pour répondre aux questions suivantes :

Q10 : Quelles sont les 3 années les plus chaudes et les 3 années les plus froides ? Quelles conclusions en tirez-vous ?

Q11 : Quels sont les 3 hivers les plus froids et les 3 étés les plus chauds ? Quelles conclusions en tirez-vous ?

Q12 : En comparant les moyennes sur 10 ans, interprétez le réchauffement pour chaque décennie ?

…

## Partie 3 : Base de données non intègre :

Il se trouve que les données de la base de données ont été préalablement traitées par nos soins afin que la base soit intègre (sans valeurs aberrantes, ou absentes). Mais hélas il ne reste que 21 stations météos qui ont les données complètes.

On vous demande de créer un nouveau projet qui reprendra l’ensemble de votre code précédant et qui exploitera les données de la base 2 (base non intègre).

Réinterprétez par exemple la question Q1 sur le mois de mai.

Il vous est demandé de rendre intègre la base. Vous pourrez donc être amené à supprimer de votre structure des arbres (ayant des valeurs de températures aberrantes) ou même des postes de votre liste. Il est important de retirer ces erreurs afin de ne pas perturber fortement vos résultats et leurs interprétations.

Voici les règles définies pour épurer la base de données :

On considérera qu’une valeur deltaT est aberrante si la valeur absolue du deltaT>25.0

* Si un arbre possède 1 ou 2 valeurs deltaT aberrantes on enlève ces nœuds de l’arbre.
* Si un arbre (pour un mois donné) possède 3 ou plus valeurs aberrantes on supprime l’arbre.
* Si on doit enlever 2 ou plus arbres à une station, on supprime la station.

En conclusion est-ce que les résultats redeviennent plus conforme aux résultats de la partie 2 ?