Explications de la classe Vector (namespace pancar) :

Ce pdf a pour objectif d'expliquer en détails l'intérêt et le fonctionnement de la classe *Vector* du namespace *pancar*. En effet le code peut sembler à première vue un peu flou voire abstrait et l'intérêt d'une telle classe pourrait échapper au correcteur.

Pourquoi cette classe ?:

L'objectif de cette classe est de remplir les mêmes fonctions que les collections de Java ou les conteneurs de la bibliothèque standard du langage C++. En effet, malgré mes recherches je n'ai pas trouvé de bibliothèque qui remplisse les mêmes fonctions en Arduino. En fin de compte il serait prétentieux de la comparer aux conteneurs sans itérateurs ni foncteurs donc cette classe se rapproche davantage des collections. Un dessin de pancake est en réalité une collection d'un nombre indéfini de points, positionnés par l'utilisateur avec un logiciel Java préalablement programmé.

Cette classe peut aussi être héritée par d'autres classes, par exemple pour les groupes de moteurs pas-à-pas…

Comment cette classe fonctionne ?:

Cette classe fonctionne par une récursivité sur des pointeurs de tableaux de pointeurs de référence de template. Les templates sont une fonctionnalité offerte par le langage C++ pour avoir des types génériques. Par exemple, si on veut une collection d'entiers, on va écrire *pancar::Vector<int> maCollectionDEntiers*, pour des String *pancar::Vector<String> maCollectionDeStrings* etc. Une même classe peut servir pour tous les types souhaités ! Les templates n'étant pas vraiment compatibles avec la compilation séparée, j'ai dû mettre les déclarations **et** les instanciations dans le fichier d'entête.

Contrairement aux conteneurs de la SL C++, cette classe travaille non pas avec les valeurs mais avec les références des variables. C'est en effet nécessaire dans certains cas notamment avec les moteurs. Par contre notre classe n'est pas responsable des objets qu'elle collectionne. Si par exemple le développeur modifie une variable après l'avoir stockée dans le Vector, la valeur stockée se verra aussi modifiée. De plus on ne peut pas passer de valeur directement :

```
pancar::Vector<int> maCollectionDEntiers;
int entier=10;
```

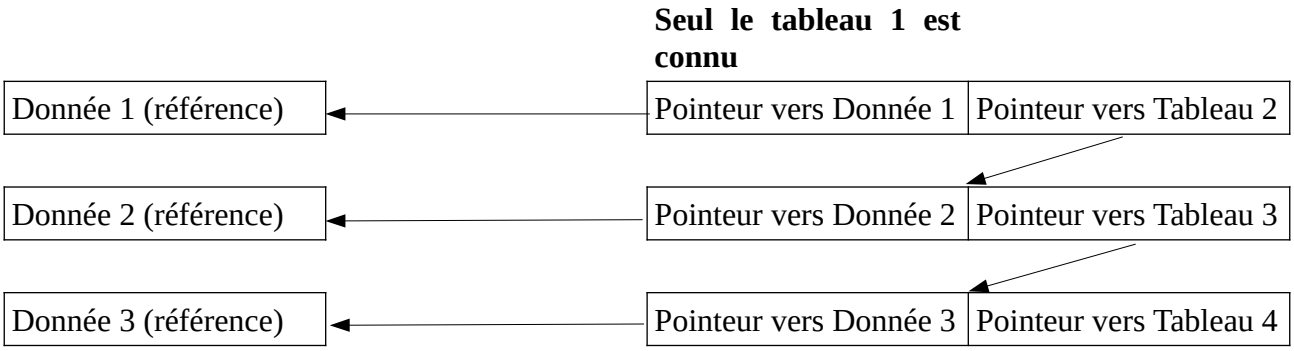
maCollectionDEntiers.add(entier); // OK

maCollectionDEntiers.add(10); // NON

maCollectionDEntiers.add((new int(10))); // OK*

La récursivité de tableaux de pointeurs :

Comment stocker un nombre indéfini de données dans un seul et même tableau ? Une solution serait de créer un tableau « trop grand » pour n'en utiliser ensuite qu'une partie. Cependant cette solution n'est pas très « propre » et elle ne convient pas vraiment pour nos points parce qu'il peut y en avoir vraiment beaucoup ! L'idée est donc de créer des tableaux de 2 pointeurs : le premier pointant vers la donnée intéressante et le deuxième vers le deuxième tableau. Ainsi seul le premier tableau peut être connu de la classe, d'où la récursivité.



ETC…

Pour ajouter un élément au Vector, il faut créer un nouveau tableau de pointeurs et faire pointer la case 2 du dernier tableau vers le « nouveau dernier tableau ». Pour enlever un élément au Vector, il faut faire pointer la deuxième case du tableau précédent vers le tableau suivant.