TP1 C++ : Première classe

# INTRODUCTION

Les objectifs de ce premier TP sont assez limités puisqu'il s'agit de la première prise en main des outils de programmation vue en cours. On a essayé ainsi d'appliquer des notions d'algorithmie, d'orienté objets et de technique spécifique tels-que l'allocation dynamique, le gcc ou le makefile. Ce TP nous a permit d'apprendre à bien structurer son code (respecter le guide de style INSA) et bien le commenté pour qu'il soit facile à lire.

# CAHIER DES CHARGES

L'application qu'on nous a demandée de concevoir doit manipuler une seule classe simple mais dynamique qui représente une collection d'objets quelconque. Cette classe doit ainsi contenir deux constructeur : un pour créer une collection vierge avec une capacité fixé à la création et un deuxième pour initialiser la collection avec un tableau d'objets donné et une méthode pour afficher son contenu. Cette classe doit contenir une méthode pour ajouter et une autre pour retirer un objet donné en paramètre dans la collection, une méthode pour ajuster la taille de la collection à la taille spécifié et une méthode qui sert à réunir deux collections d'objets. Il nous faut aussi implémenter le destructeur de la classe étant donné la nature dynamique de la classe collection.

# REALISATION

Le cahier des charges a été volontairement flou et ambigu. Il a donc fallu faire plusieurs choix pour pouvoir implémenter la classe collection. Étant donné qu'il nous a été interdis d'utiliser les templates, nous avons choisi de créer une classe simple nommée « dog » qui sera contenue par notre collection.

La classe dog nous permet de mettre en évidence le fait que le type de données contenu par la classe collection pourrai être de taille quelconque. Afin d'éviter des copies de données éventuellement de taille supérieure à celle d'un pointeur, on a choisi d'utiliser un tableau de pointeurs qui pointent sur des dogs comme structure de données interne à la classe collection. La classe collection va ainsi manipuler voir copier le tableau de pointeur au lieu des objets de type dog afin d'optimiser la mémoire et le temps d’exécution de l'application.

Un autre choix qui a été pris était d'ordonner les éléments de la collection dans l'ordre d'ajout.

Nous avons en outre choisit d'empêché le compilateur d'implémenter automatiquement le constructeur de copie et le « copy assignment operator » avec le mot clés « delete » car il nous était interdis de les implémenter et que la classe collection contient un pointeur qui ne devrait pas être copié par valeur lors de la copie de l'objet de type collection.

Concernant l'allocation de mémoire pour l'ajout d'objets dans la collection, on a préféré d'allouer de la mémoire par paquet c'est à dire : allouer plusieurs éléments vide à mettre dans la collection à la fois afin de réduire le cout d'ajout d'ajout d'objets à la collection.

# TESTS UNITAIRES

# CONCLUSION : AMELIORATIONS POSSIBLES