# TP C++ 7 : Patrons

M1 Mathématiques Appliquées

2019-2020

Dauphine | PSL
UNIVERSITÉ PARIS

*Remarques préliminaires :*

- Il est fortement recommandé de valider les questions de manière incrémentale à l'aide d'un programme principal. Pour ce faire, il est conseillé de "tracer" son code en affichant des messages lors d'appels à des fonctions, en vérifiant que des valeurs devant être modifiées l'ont effectivement été, etc.
- On codera les classes dans des fichiers à part (.h pour la déclaration, .cpp pour la définition) et on utilisera un fichier de test par exercice.

## Exercice 1 : Tri et patron de fonctions

Dans cet exercice, on utilise une variante de l'algorithme du tri à bulles pour effectuer le tri d'un tableau par comparaisons successives.

L'algorithme utilisé suppose donc que l'on peut utiliser l'opérateur `>` pour comparer les éléments entre eux. Il procède de manière itérative en utilisant un paramètre $k$. Au début de l'algorithme, la valeur de $k$ est initialisée avec la taille du tableau. À chaque itération, l'algorithme détermine le plus grand éléments parmi les $k$ premières valeurs du tableau et place sa valeur dans la case d'indice $k-1$ du tableau[1], puis décrémente $k$ de 1. Lorsque $k = 1$, le tableau est entièrement trié.

1. Créer un patron de fonctions `tri` ayant pour arguments un tableau, sous la forme d'un pointeur sur un type paramétré, et un entier représentant la taille de ce tableau. Le corps de ce patron devra implémenter l'algorithme décrit en préambule, en supposant que l'opérateur `>` sera défini pour le type utilisé.
   Tester ce patron sur des types de bases (`int`,`float`,`char`).

2. Créer une classe `Point` telle que celle utilisée en cours, et y redéfinir l'opérateur `>` pour que les points soient comparés selon l'ordre lexicographique (càd $(a,b) > (c,d)$ si et seulement si $a > c$ ou $a = c$ et $b > d$). Vérifier ensuite que le patron de fonctions `tri` fonctionne bien lorsqu'il est instancié pour la classe `Point`.

[1] On rappelle que les indices commencent à 0 en C++.

## Exercice 2 : Coordonnées et patron de classes

1. Créer un patron de classes `Coord` modélisant un ensemble de valeurs du même type (paramétré), ou coordonnées, dont le nombre de valeurs sera également un paramètre du patron. Ce patron devra comporter les éléments suivants :

   i) Un constructeur prenant en paramètre un tableau d'éléments dont le type correspondra à celui des coordonnées, et dont on supposera que la taille correspondra au paramètre du patron[2] : ce tableau initialisera les valeurs utilisées par l'objet.

   ii) Le reste de la forme canonique, c'est-à-dire le destructeur, le constructeur de recopie et l'opérateur d'affectation.

   iii) Une surdéfinition de l'opérateur `+` pour effectuer l'addition de deux objets correspondant à la même instanciation de `Coord`. L'addition se fera coordonnée par coordonnée. L'opérateur ne devra pas modifier ses opérandes, mais renverra le résultat de l'addition (objet du type correspondant à la même instance de `Coord`). *Remarque : une telle fonction devra allouer un nouveau tableau de données.*

   iv) Une surdéfinition de l'opérateur `*` pour permettre le produit scalaire entre deux objets correspondant à la même instanciation de `Coord`. Le type de retour de l'opérateur sera le même que celui des coordonnées.

2. Définir ensuite un patron de fonctions `concat` prenant deux objets `Coord` ayant le même type mais pas forcément le même nombre de coordonnées, et créant un nouvel objet étant la concaténation de ces deux points. Ainsi, la concaténation d'un objet `Coord<int,2>` et d'un objet `Coord<int,3>` donnera un objet `Coord<int,5>` dont les 2 premières coordonnées correspondront à celles du premier objet, et les deux suivantes à celle du second objet. *Remarque : comme pour l'opérateur `+`, on allouera un nouveau tableau de données.*

3. Reprendre enfin la classe `Point` définie dans l'exercice 1. Surcharger l'opérateur `+` pour qu'il renvoie la somme de deux points, dont l'abscisse et l'ordonnée seront la somme des abscisses et ordonnées des deux points, respectivement. Surcharger de la même manière l'opérateur `*` pour qu'il renvoie le "produit" de deux points, dont l'abscisse et l'ordonnée seront le produit des abscisses et ordonnées des deux points, respectivement. Tester l'instanciation du patron de classes `Coord` correspondant à la classe `Point`.

[2] On pourra dans un second temps vérifier qu'il s'agit bien de la même taille.