Travaux dirigés n°2

Xavier JUVIGNY

4 février 2022

1 Exercice 1

Écrire une classe `Point` qui représente un point dans l'espace $\mathbb{R}^3$. On pourra :

- Construire un point à partir de trois réels
- Construire un point à partir d'une liste d'initialisation
- Calculer la distance entre deux points (méthode ou fonction : choix d'interface)
- Un opérateur de converstion qui transforme le point en une chaîne de caractères représentant les données du point. Exemple, `"(1.,2.,-3.)"`

Écrire une classe `Vecteur` qui représente un vecteur en trois dimensions. On pourra :

- Construire un vecteur à partir de trois réels
- Construire un vecteur à partir d'une liste d'initialisation
- Construire un vecteur à partir de deux points ($\overrightarrow{u} = \overrightarrow{AB}$)
- Un produit scalaire utilisant l'opérateur `|` (pour écrire $(u|v)$)
- Un produit vectoriel utilisant l'opérateur `^`
- Une homothétie (qu'on pourra écrire sous la forme $v = \alpha.u$)
- Un opérateur de conversion en chaîne de caractère
- Pouvoir translater un point avec un vecteur

Dans tous les cas, chercher à obtenir l'interface la plus naturelle possible et tester vos fonctions dans un programme principal

2 Exercice 2 : Gestion d'un nuage de points

On reprendra les classes de l'exercice 1 pour répondre à cet exercice.

- Construire une classe NuageDePoints permettant d'instancer des nuages de N points en 3D, N pouvant varier d'un nuage à l'autre ;
- Mettre des affichages au début de chaque constructeur et destructeur pour tracer l'appel des constructeurs ;
- Tester votre classe à l'aide du programme suivant :

```
# include "nuage_de_points.hpp"
int main()
{
    NuageDePoints nuage1(10);
    NuageDePoints nuage2(nuage1);
    NuageDePoints nuage3;
    NuageDePoints nuage4(2, 5, nuage1);// Copie les points 2 à 5 du nuage1.
}
```

- Bonus : Gérer les erreurs lors de la construction d'un nuage à l'aide des exceptions ;
- Bonus : Pouvoir construire un nuage de la manière suivante :

```
NuageDePoints nuage5{ point{1.,0.},point{1.,1.},point{0.,1.} };
```

où point est un alias sur std ::array<double,2>

- Rajouter les opérateurs de copie/déplacement
- Accéder au i^e point p_i pour le lire/modifier ;
- Rajouter les opérateurs adéquats pour que le test suivant fonctionne :

```
NuageDePoints cop1, cop2;
      ...
NuageDePoints cop3 = cop1 + cop2;// cop3 = fusion de cop1 et cop2
Vecteur tr{1.,0.,0.};
cop3 += tr; // Translation des points par le vecteur tr
// Affiche nombre de points et les 1ers et derniers points...
std::cout << "cop3 : " << std::string(cop3) << std::endl;
// Sauvegarde le nuage de points :
ifstream fich("cloud.dat"); fich << cop3; fich.close();
```

- Rajouter une méthode donnant le nombre de points,
- Pouvoir itérer sur les points du nuage ;
- Calculer le point barycentre du nuage de point.

On s'attachera également à écrire une interface qui peut s'utiliser "naturellement".