JUNIA Grande école d'ingénieurs

HEI · ISEN · ISA

TP3 / C++

Exercice 1 :

Soit la classe *Point3D* définie comme suit (dans *Point3D.hpp*) :

```
class Point3D {
	private:
		float x,y,z;	// private attributes

	public:
		// constuctors
		Point3D(); // fill X Y Z with random values (from 0 to 100)
		Point3D(const float &newx, const float &newy, const float &newz);
// fill XYZ values

		// Setters and getters
		void setXYZ(const float &newx, const float &newy, const float
&newz);
		void setX(const float &newx);
		void setY(const float &newy);
		void setZ(const float &newz);
		float getX();
		float getY();
		float getZ();

		// other methods
		void print();	// prints the coordinates of the point
		float distanceTo(const Point3D &otherPoint3D);
};
```

1) Développer l'ensemble des méthodes dans le fichier *Point3D.cpp*.

2) Tester les différentes méthodes dans la fonction *main()*.

Remarque : Pour la génération de nombres aléatoires, faire appel aux fonctions *rand()* et *srand()* de la bibliothèque *<cstdlib>* (ou les nouvelles fonctions de la bibliothèque « randow » propre au C++).

Exercice 2 :

Soit la classe *Trajectory* définie comme suit (dans *Trajectory.hpp*) :

```
#include "Point3D.hpp"

constexpr size_t numberOfPoints = 10;
class Trajectory{
	private:
		Point3D points[numberOfPoints];
	public:
		void print(); // print the coordinates of all points
```

```
        Point3D & getPoint(const int &n); // returns the reference of point n
        float getTotalDistance();
};
```

1) Développer l'ensemble des méthodes dans le fichier *Trajectory.cpp*.

2) Tester les différentes méthodes dans la fonction *main()*.

3) Modifier l'ensemble des points pour qu'ils soient sur une droite. Comparer la distance totale avec la distance entre le premier et le dernier point.

3) Modifier la classe pour que le nombre de points soit dynamique. Le constructeur prend obligatoirement le nombre de points en paramètre et alloue l'espace nécessaire. Il faut aussi développer un destructeur pour la classe.

Exercice 3 :

Il existe la norme du C++ une classe générique « vector » que nous étudierons un peu plus tard. Dans cet exercice, il est demandé de développer une classe C++ « My_vector » qui tente de « mimer» son comportement et dont le prototype est le suivant :

```
#include <cstdlib>
typedef float my_type;

class My_vector{
private :
  size_t size;
  my_type *tab;
public:
  My_vector();
  My_vector(const size_t &size);
  My_vector(const My_vector&);
  ~My_vector();
  const size_t &get_size() const;
  void set_an_element(const size_t &index, const my_type &val);
  const my_type &get_an_element(const size_t &index) const;
  void push(const my_type &val);
};
```

Implémenter l'ensemble des méthodes sachant que:

- la taille maximale allouée pour tab est de 1GO ;
- en utilisant le constructeur par défaut size est égal à 0;
- tab est initialisé à 0;
- quand le rang d'un élément du tableau dont la valeur change (set_an_element) est supérieur (ou égal) à « size », « tab » devra être réalloué et les valeurs nouvellement créées devront être initialisé à;
- l'index dans la fonction « get_an_element » devra être conforme avec la taille du tableau, si ce n'est pas le cas la fonction devra afficher un message d'erreur et retourner «0»;
- la fonction « push » ajoute un élément au tableau.