Fiche C++ n° 3 Membre static de classe

Pour l'instant, les classes que nous avons créées comportent des attributs et des méthodes. Ces éléments sont liées aux objets créés. Par exemple, la classe **Point** comporte un attribut ***x*** et une méthode ***déplace().*** Si on crée deux objets ***p1*** et ***p2***, ils auront chacun une abscisse ***x*** différente et ***p1*** pourra être déplacé sans que ne le soit ***p2***, c'est-à-dire que ***p1.deplace()*** ne modifie pas les coordonnées de p2.

Par contre, supposons que l'on veuille créer un attribut ***cpt*** qui est chargé de compter le nombre de points en mémoire. Il n'est **pas lié à un objet précis** mais à l'ensemble des objets, donc plus précisément **à la classe** elle-même. On pourrait utiliser une variable globale, mais comme il ne servirait que pour compter les points, cela pourrait poser des problèmes à cause des effets de bord. En effet, on n'est pas à l'abri d'une modification accidentelle de cette variable globale ailleurs dans le programme.

La solution est d'utiliser une variable statique de classe.

- Un membre donné déclaré avec l'attribut **static** est partagé par tous les objets de la même classe. Le mot clé **static** précède la déclaration des attributs ou méthodes (dans le .fichier .h uniquement).
- Il existe même lorsqu'aucun objet n'a été déclaré.
- Une donnée statique de classe est initialisée par défaut à 0 (comme toutes les variables statiques). Cette donnée peut néanmoins être initialisée à l'extérieur de la classe (même s'il est privé) en utilisant l'opérateur de résolution de portée (::) pour spécifier sa classe.
- Si un attribut est static, son accesseur le sera aussi.
- Depuis l'extérieur de la classe (le programme principal par exemple), l'accès à un attribut static public (ou une méthode statique) se fera sous la forme
  Classe::attributStatique ou ***Classe::methodeStatique()***
- En UML, un attribut ou méthode statique est souligné

Autre exemple

Prenons les cas d'une classe CRS232 qui comme son nom l'indique gère une liaison série de type RS232 (liaison série asynchrone). Parmi les attributs ou méthodes suivantes, dire si l'on doit les définir en **static** ou non (et dire pourquoi dans le cas où vous choisissez static

// Constructeur `CRS232(int NoPort=2);`	
`void initialiser(DWORD Vitesse=CBR_9600,BYTE NbDonnees=8 , BYTE NbStops=ONESTOPBIT ,BYTE Parite=NOPARITY);`	
// lit un octet sur le port `BYTE lire(void);`	
// Tableau des numéros de ports disponibles `vector<int> lesPortsDisponibles() ;`	
// Renvoie true si le port est déjà ouvert `bool portDejaOuvert() ;`	
// Renvoie le nombre de caractères pouvant être lus sur le port `int nombreCaracteresEnAttenteDeLecture();`	
// Renvoie false si aucun port série RS232 n'est supporté par le PC `bool presenceAuMoinsUnPortRS232();`	

Le programme illustre la syntaxe d'utilisation des attributs et méthodes static sur la classe Point.

```
//**** point.h *********************
// Déclaration de la classe point

class Point
{
public:
        void affiche(void);
        void deplace(float tx, float ty);
        static int getNbPoints();

        Point(float x1=0 , float y1=0);
        ~Point();

private:
        static int nbPoints;
        float y;
        float x;
};
```

Méthode static

Attribut static

```
//******* point.cpp ***************
// Définition des fonctions membres de la classe
#include <iostream>
#include <conio.h>
#include "point.h"

using namespace std;

int Point::nbPoints=0;

Point::Point(float x1 , float y1)
{
        x=x1;
        y=y1;
        nbPoints++;
}

Point::~Point()
{
        nbPoints--;
}

void Point::deplace(float tx, float ty)
{
        x+=tx;y+=ty;
}

void Point::affiche()
{
        cout <<"\nx="<<x<<"y="<<y<<"
        "<<nbPoints<<" en mémoire";
}

int Point::getNbPoints()
{
        return nbPoints;
}
```

Initialisation de la variable statique de classe.

```
// fiche 3.cpp
// Fonction principale
#include "point.h"

int main()
{
        Point *p1;
        int taille;

        cout <<"Nombre de points en mémoire = " << Point :: getNbPoints() << endl;

        p1=new Point(3,4);
        cout <<"Nombre de points en mémoire = " << Point :: getNbPoints() << endl;

        p1->deplace(1,2);
        p1->affiche();

        delete p1;
        cout <<"Nombre de points en mémoire = " << Point :: getNbPoints() << endl;

        cout << "\nRentre la taille du tableau ";
        cin >>taille;
        Point *tab = new Point[taille];
        cout <<"Nombre de points en mémoire = " << Point :: getNbPoints() << endl;

        delete[] tab;
        cout <<"Nombre de points en mémoire = " << Point :: getNbPoints() << endl;

        _getch();
        return 0;
}
```

J'appelle la méthode statique alors que je n'ai pas encore créé d'objet. Mémorisez bien la syntaxe :
Classe :: MethodeStatique()

Nombre de points en mémoire = 0

Nombre de points en mémoire = 1

Nombre de points en mémoire = 0

Nombre de points en mémoire = *taille*

Nombre de points en mémoire = 0