

## TP4 / C++

### Exercice 1 :

Soit la code C++ suivant :

```
#include <iostream>
using namespace std;

class A {
public :
    A () { x = 0;                cout << "1"; }
    A (int px) { x = px;         cout << "2"; }
    A (const A& pa) { x = pa.x;  cout << "3"; }
protected :
    int x;
};

class B {
public :
    B (const A& pa=A()) : a(pa) { cout << "4"; }
    B (const A& pa, int py) { a = pa; y = py; cout << "5"; }
protected :
    A a;
    int y;
};

int main() {
cout << "-- A --\n";
A a0;          cout << endl;
A a1(3);        cout << endl;
A a2(a1);       cout << endl;
A a3 = a2;      cout << endl;
a3 = a1;        cout << endl;
cout << "-- B --\n";
B b0(a0,3);     cout << endl;
B b1(a1);       cout << endl;
B b2;           cout << endl;
}
```

- 1) Compiler, exécuter ce programme.
- 2) Interpréter, ligne par ligne, les sorties de la fonction `main()` ;

### Exercice 2 :

- 1) Implémenter les classes `Circle` vue en cours, la classe `Point`.
- 2) Implémenter les méthodes `getArea()` et `getPermieter()` pour cette classe.
- 3) Ecrire une seconde version pour l'estimation de la surface du cercle sans utiliser le nombre  $\pi$  (en utilisant les méthodes Monte Carlo, la somme alternée des inverses des impairs ou la somme des inverses des carrés). Utiliser le même algorithme pour écrire une fonction statique d'estimation du nombre Pi

### Exercice 3 :

1) développer une classe *Vector*, ainsi que les méthodes pour :

- construire le vecteur à partir de deux points,
- calculer la somme de deux vecteurs,
- calculer le produit par un réel,
- vérifier l'égalité entre deux vecteurs.

### Exercice 4

En utilisant *Point* et *Vector*, Ecrire une classe *Polygon* :

- Un tableau de points représente les sommets.
- Une méthode permet de retourner l'aire du polygone en utilisant la formule :

$$\frac{1}{2} \sum x_i y_{i+1} - x_{i+1} y_i$$

Vérifier votre méthode pour un triangle, un carré et un rectangle dans la fonction *main*.

### Aller plus loin:

- Une méthode permettant de retourner le périmètre du polygone (inspirée de la distance totale d'une trajectoire).
- Une méthode permet de savoir si le polygone est convexe.
- Une méthode permettant de savoir si un point est à l'intérieur du polygone.

Certains algorithmes sont disponibles sur la page :

<https://openclassrooms.com/courses/theorie-des-collisions/formes-plus-complexes>