

Programmation Orienté Objet

2022-2023 Travaux Pratiques 3

Site du cours : <https://defelice.up8.site/poo.html>

`g++ mon_fichier.cpp -std=c++17 -Wall -Wextra -fsanitize=address -o mon_programme`

Les exercices marqués d'un @ sont à faire dans un second temps.

Exercice 1. *Modélisation d'intervalles bornés*

A corriger en TD.

Le but de cet exercice est d'implanter une classe pour représenter les intervalles fermés bornés.

Intervalle (fermé borné)

Un intervalle fermé borné est un sous-ensemble des nombres réels que l'on note souvent $[a; b]$ où a et b sont des nombres réels avec $a < b$. Les valeurs a et b sont appelées les *extrémités* de l'intervalle. Par exemple, $[-2; 4]$ et $[-1,3; 4,98]$ sont deux intervalles (fermés bornés). L'ensemble vide est un intervalle et $[3; 3]$ est un intervalle qui ne contient qu'un seul élément : 3.

Un nombre n appartient à l'intervalle...

- s'il est égale à l'une des extrémités, ou bien
- s'il se situe entre les extrémités.

Exemple :

- . 3,5 appartient à $[2; 4]$,
- . 2 appartient à $[2; 4]$,
- . 4 appartient à $[2; 4]$,
- . mais 4,5 n'appartient pas à $[2; 4]$.

Les intervalles étant des ensembles on peut considérer l'intersection (\cap) ou l'union (\cup) d'intervalles. Une intersection d'intervalle est toujours un intervalle ce n'est pas le cas pour l'union. Par exemple $[1; 3] \cap [2; 4] = [2; 3]$. Mais $[-2; -1] \cup [1; 2]$ n'est pas un intervalle.

But de l'exercice

Implanter en C++ une classe **Interv** dont chaque objet (i.e instance) modélise un intervalle fermé borné, pour les extrémités on utilisera des valeurs de types **double**.

La classe **Interv** doit contenir les méthodes suivantes :

- Un constructeur avec deux paramètres **double** (les deux extrémités de l'intervalle),
- un constructeur sans paramètre (qui construit l'ensemble vide),
- un constructeur avec un paramètre **a** (qui construit $[a; a]$),
- une méthode **appartient(double e)** qui renvoie 1 (ou **true**) si le nombre **e** est dans l'intervalle, 0 sinon,
- une méthode **Inter intersec(Interv const b)** qui construit et renvoie l'intersection de deux intervalles,
- une méthode **estVide(void)** qui renvoie 1 si l'intervalle est l'ensemble vide,
- une méthode **union** qui renvoie l'union de deux intervalles seulement si cette union est elle-même un intervalle. Dans le cas contraire on peut interrompre le programme par exemple avec l'instruction **throw 1;**.

Exercice 2. *Compte copie*

Construire la classe **CompteInstance** qui possède une méthode publique **static int compter(void)** renvoyant le nombre d'objets de la classe qui existent actuellement.

Exemple :

```
CompteInstance a;
{
    CompteInstance b;
    CompteInstance c;
```

```

    CompteInstance d;
    CompteInstance::compter(); // renvoie 4 (car 4 objets a b c d)
}
CompteInstance::compter() // renvoie 1 (seule a existe ici)

```

Exercice 3. *Suite pseudo-aléatoire*

On souhaite créer une classe `alea_t` qui génère une suites de nombres ressemblant à une suite aléatoire. Exemple d'utilisation

```

int graine=56;
alea_t genA{graine};
alea_t genB{graine};
int a=genA.rand(); // renvoie un nombre aléatoire
int b=genB.rand();
// a==b ici
genA.srand(0);

```

Pour la génération on peut par exemple utiliser la formule $n = a * 1103515245 + 12345$ où n est la prochaine valeur renvoyée par la méthode `rand()` et a l'ancienne valeur. Sinon vous pouvez toujours utiliser la méthode de la fonction `rand()` de la bibliothèque standard du C (voir `$man 3 rand`)

Exercice 4. *Somme*

Écrire une classe `intreliee_t` qui possède au moins les méthodes suivantes.

- un constructeur qui produit un objet mémorisant une valeur entière.
- `set(int a)` qui modifie cette valeur.
- `int get()` qui renvoie la valeur.
- `int somme()` qui renvoie la somme de toutes les valeurs entières mémorisée dans un objet.

Indice : La méthode `somme` est-elle forcément liée à un objet ? **Précision :**

- A priori la méthode `somme()` renvoie une valeur différente avant ou après qu'un objet soit détruit.
- on suppose ici qu'un utilisateur de la classe ne peut pas utiliser `=` entre deux objets de la classe.