

Programmation orientée objet

LIV 2025-2026

Travaux pratiques 1 - Première classe

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

Les exercices marqués de (©) sont optionnels.

Un fichier écrit en langage C++ se termine conventionnellement par .cpp.

Une commande de compilation est `g++ fichier_source1.cpp fichier_source2.cpp fichier_source3.cpp`.

Voici des options de cette commande.

- `-o nom_sortie` pour donner un nom au fichier de sortie (par défaut `a.out`).
- `-Wall` pour demander au compilateur d'afficher plus d'avertissements (warnings)
- `-Wextra` pour demander au compilateur d'afficher plus d'avertissements (warnings)
- `-std=c++20` pour compiler selon la norme c++20
- `-g -fsanitize=address` pour que l'exécutable puisse detecter certaines erreurs de mémoire (adresse inaccessible, fuite mémoire) (l'inconvenient est que l'exécutable est plus lent, `-g` pour avoir plus de précision).
- `-g` pour ajouter des informations de débogage, utile avec `debug` (exécutable plus lent).

Exemple : `g++ -Wall -Wextra fichier1.cpp -o monprogramme`

Exercice 1. Premier programme

Copier ou recopier le programme suivant compilez-le puis exécutez-le. (Le fichier est disponible sur le site du cours).

annexe01.cpp

```
1 #include<iostream>
2 #include<cstdio>
3
4 // ceci est un commentaire sur une ligne
5 /* ceci est un commentaire sur
6 plusieurs lignes*/
7
8 using namespace std; // permet d'éviter d'écrire systématiquement std::cin ou std::cout (à éviter)
9
10 int main(void)
11 {
12     int d; // une variable entière d
13     int i;
14
15     cout<<"****Premier programme****"<<endl;
16     std::cout<<"Saisissez un nombre puis appuyez sur entrée : ";
17     std::cin >> d;
18
19     i=1; // on met 1 dans i
20     while(i<=15)
21     {
22         if( d % i == 0)
23         {
24             cout << i << " divise "
25                 << d << endl;
26         }
27         else
28         {
29             printf("%d ne divise pas %d\n",i,d);
30         }
31         i=i+1;
```

```
32 }
33 }
```

La classe/le type `string` permet de manipuler plus aisément les chaînes de caractères. C'est un type de la bibliothèque standard C++ pour l'utiliser il faut ajouter une ligne `#include<string>` et soit utiliser le préfixe `std::` soit utiliser l'instruction `using namespace std;`.

```
std::string a="Bonjour ";
std::string b="à tous !";
std::string d=std::to_string(15);
a=a+" ";
std::string c=a+b;
char e=b[0]; // renvoie le première octet du codage de "à tous !"
std::cout << (c + d) << std::endl; // affiche Bonjour à tous !15
std::cout << (c=="au revoir à tous") << endl ;
// (c=="..." compare un string et une chaine de caractère)
```

Exercice 2. Modifié

En utilisant la classe/type `string` modifier le programme précédent pour qu'il affiche :

```
****Premier programme****
Saisissez un nombre puis appuyez sur entrée : 45
1 : divide(nt) 45 | : ne divide(nt) pas 45
1 : divide(nt) 45 | 2 : ne divide(nt) pas 45
1 3 : divide(nt) 45 | 2 : ne divide(nt) pas 45
1 3 : divide(nt) 45 | 2 4 : ne divide(nt) pas 45
1 3 5 : divide(nt) 45 | 2 4 : ne divide(nt) pas 45
...
...
1 3 5 9 : divide(nt) 45 | 2 4 6 7 8 10 : ne divide(nt) pas 45
...
..... 45 : divide(nt) 45 | ..... 44 : ne divide(nt) pas 45
```

On prend en compte un nombre en plus à chaque ligne.

Exercice 3. Première classe

On utilise la classe suivante (fichier disponible sur le site du cours) :

```
annexe02.cpp
1 #include<iostream>
2
3 using namespace std;
4
5 struct Point {
6     private :
7         float x;
8         float y;
9     public:
10         Point(float px,float py)
11         {
12             x=px;
13             y=py;
14         }
15         void affiche()
16         {
17             cout << "(" << x << "," << y << ")" << endl;
18         }
19         void rotate()
20             // effectue une rotation de +90 degrés ayant pour centre le point (0,0)
```

```

21     {
22         float tmp=x;
23         x=-y;
24         y=tmp;
25     }
26 };
27
28 int main()
29 {
30     Point p{4,5};
31     Point c{-4,1};
32     c.affiche();
33     c.rotate();
34     c.affiche();
35 }
36

```

Implanter la méthode `void translate(Point a)` qui effectue une translation donnée par le vecteur $\overrightarrow{Oa}$ à l'objet.

Exercice 4. *Modificateur*

1. Avec la classe `Point` de l'exercice précédent. Pourquoi le code suivant

```

.
.
Point a{3,4};
a.x=2;
.
.

```

ne compilera pas ?

2. Ajouter à la classe `Point` deux méthodes `void setX(float x)` et `void setY(float y)` qui permettent de modifier les attributs `x` et `y` du point.

Exercice 5. *Barycentre*

Implanter `Point barycentre(Point a)` une méthode de la classe `Point` qui renvoie le milieu du segment donné par le point `a` et l'objet. Exemple :

```

Point a{0,2};
Point b{0,6};
Point c=b.barycentre(a);
// le point c doit se trouver en (0,4) : milieu de [a,c]

```

Pour trouver le milieu on peut utiliser la formule $x = \frac{x_1+x_2}{2}$ et $y = \frac{y_1+y_2}{2}$