Programmation Orienté Objet

L2 2022-2023

Travaux Pratiques 1

Site du cours : `https://defelice.up8.site/poo.html`
`g++ mon_fichier.cpp -std=c++17 -Wall -Wextra -o mon_programme`
Les exercices marqués d'un @ sont à faire dans un second temps.

Exercice 1.

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

annexe01.cpp

```
#include<iostream>
#include<cstdio>

// ceci est un commentaire sur une ligne
/* ceci est un commentaire sur
plusieurs lignes*/

using namespace std;  // permet d'éviter d'écrire systématiquement std::cin ou std::cout (à évi

int main(void)
{
   int d; // une variable entière d
   int i;

   cout<<"****Premier programme****"<<endl;
   std::cout<<"Saisissez un nombre puis appuyez sur entrée : ";
   std::cin >> d;

   i=1; // on met 0 dans i
   while(i<=15)
   {
      if( d % i == 0)
      {
         cout << i << " divise "
            << d << endl;
      }
      else
      {
         printf("%d ne divise pas %d\n",i,d);
      }
      i=i+1;
   }
}


```

La classe/le type `string` permet de manipuler plus aisement les chaines de caractères. C'est un type de la bibliothèque standard C++ pour l'utiliser il faut inclure `#include<string>` et soit utiliser le prefixe `std::` soit utiliser l'instrution `using namespace std;`.

```
std::string a="Bonjour ";
std::string b="à tous !;
std::string d=std::to_string(15);
std::string c=a+b;
char c=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.

En utilisant la classe/type string modifier le programme précédent pour qu'il affiche quelque chose comme

```
****Premier programme****
Saisissez un nombre puis appuyez sur entrée : 45
1 divise 45,
1 divise 45, 2 ne divise pas 45
1,3 divise 45, 2 ne divise pas 45
1,3 divise 45, 2,4 ne divise pas 45
..
```

Exercice 3.

On utilise la classe suivante :

annexe02.cpp

```
#include<iostream>

using namespace std;

struct Point {
    private :
       float x;
       float y;
    public:
       Point(float px,float py)
       {
         x=px;
         y=py;
       }
       void affichePoint()
       {
         cout << "(" << x << "," << y << ")" << endl;
       }
       void rotate()
          // effectue une rotation de +90 d. ayant pour centre le point (0,0)
       {
         float tmp=x;
         x=-y;
         y=tmp;
       }
   };

int main()
{
   Point p(4,5);
```

```
31    Point c(-4,1);
32    c.affichePoint();
33    c.rotate();
34    c.affichePoint();
35  }
```

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

Exercice 4.

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(int x)` et `void setY(int y)` qui permettent de modifier les attributs `x` et `y` du point.

Exercice 5.

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 $y = \frac{y_1+y_2}{2}$