Langage C++

- CIR2 -

ISEN ALL IS DIGITAL! NANTES | yncréa

TP2 : Création des classes en C++

Ayoub KARINE (ayoub.karine@isen-ouest.yncrea.fr)

Exercice 1

Partie 1 : méthodes et attributs publiques

Créer une classe "**Equation**" qui contient des attributs et des méthodes **publiques** permettant de résoudre une équation de second degré $ax^2 + bx + c$

On se restreint aux solutions (racines) réelles de cette équation.

- Les attributs de la classe sont les coefficients de l'équation (a, b et c)
- Les méthodes de la classe sont :
    - une méthode pour saisir les coefficients
    - une méthode de résolution de l'équation
    - une méthode pour l'affichage de l'équation :
      Exemple: 2x^2 + 5x + 8 = 0
    - une méthode pour le calcul de l'image d'une variable par cette équation (par ce polynôme).
      Cela permet aussi de vérifier si les racines trouvées par la méthode de résolution de l'équation sont correctes

Consignes :

- Définir la classe (attributs et prototypes des méthodes décrites ci-dessous) dans un fichier ".h"
- Définir les méthodes dans un fichier ".cpp"
- Pour la racine, utiliser la fonction `sqrt` de la bibliothèque `cmath`

Partie 2 : restriction d'accès aux attributs

Modifier la classe "**Equation**" pour restreindre l'accès aux attributs. Le but va être de s'assurer que l'équation est bien du second degré en vérifiant que la valeur de "a" est non nulle dans l'équation : $ax^2 + bx + c$

Pour cela, ajouter les getters et les setters dans la classe "**Equation**".

Partie 3 : test des fonctionnalités de la classe

Tester les différentes fonctionnalités de la classe "**Equation**" pour la partie 1 et la partie 2 sachant que les valeurs de "a", "b" et "c" sont saisies par l'utilisateur.

N'oublier d'inclure le fichier ".h" dans le fichier "main.cpp".

Exercice 2

Partie 1 : méthodes et attributs publiques

Créer une classe "**Point**" qui encapsule :

- les abscisses et les ordonnées (x,y) comme attributs publiques
- une méthode de calcul de distance entre deux points (le point courant de la classe et un autre point) → publique

```
double distance(Point P)
```

Pour le calcul de la racine utiliser la bibliothèque `#include <cmath>`

- une méthode de calcul du segment composé de deux points (le point courant de la classe et un autre point) ⟶ publique

```
Point milieu(Point P)
```

Partie 2 : test du fonctionnement de la classe en utilisant une allocation statique

Le résultat après l'exécution du programme est :

```
**SAISIE DU POINT A**
Tapez l'abscisse : 3.2
Tapez l'ordonnée : 1.4
L'abscisse vaut : 3.2
L'ordonnée vaut : 1.4

**SAISIE DU POINT B**
Tapez l'abscisse : 5
Tapez l'ordonnée : 6
L'abscisse vaut : 5
L'ordonnée vaut : 6

**MILIEU DE AB**
L'abscisse vaut : 4.1
L'ordonnée vaut : 3.7

** La distance AB vaut : 4.93964
```

Partie 3 : restriction d'accès aux attributs

Modifier la classe "**Point**" en :

- déclarant les attributs comme privés
- ajoutant les méthodes publiques suivantes :

```
void setX(double x); # le nom de l'argument et le même que l'attribut de la classe
void setY(double y); # le nom de l'argument et le même que l'attribut de la classe

double getX(); # pour récupérer X
double getY(); # pour récupérer Y
```

```
void saisir(); # pour permettre à l'utilisateur de saisir les coordonnées des points
void afficher(); # pour afficher les coordonnées des points
```

Partie 4 : Instanciation des objets via une allocation dynamique

1. Créer une nouvelle classe "**Point1**" en suivant les instructions suivantes :
    a. La déclaration des attributs (x et y) de la classes "**Point**" sera la même de la classe "**Point1**" (déclarer-les comme privés)
    b. La création des objets dans le "**main()**" sera, cette fois-ci, dynamique. En prenant ce constat en considération, proposer des nouvelles versions des prototypes et des définitions des fonctions développées dans les parties 2 et 3.
    c. N'oublier pas de supprimer les instances créées dynamiquement avec `delete`

Partie 4 : Création et suivi d'un trajet aléatoire

Consigne : **Toutes les fonctions programmées dans cette partie vont être intégrées dans le fichier "`main.cpp`" avant la fonction "`main()`"**

1. Pour générer une valeur aléatoire en C/C++, la fonction "`rand()`" de la bibliothèque "`cstdlib`"est souvent utilisée.

Insérer le code de la fonction ci-dessous dans votre code. Elle permet de générer un nombre aléatoire dans un interval [min, max] :

```
double randomDouble(double min, double max) {
   return (rand() / ((double)RAND_MAX)) * (max-min) + min;
}
```

2. La fonction `rand()` se base sur une valeur initiale. Pour changer cette valeur initiale à chaque exécution, il faut utiliser un entier qui change tout le temps comme argument de la fonction "`srand()`". Par exemple, on peut utiliser le timestamp UNIX qui correspond à l'heure courante, que l'on peut obtenir en utilisant la fonction time(). Cette fonction nécessite l'inclusion :

```
#include <ctime>
```

Ajouter, donc, dans la fonction "main()" :

```
srand( time(NULL) );
```

3. Proposer une fonction

```
Point1* createRandomPath(int nbSteps, double minX, double maxX,
double minY, double maxY){
 Point1* tabpoints = new Point[nbSteps];// création de nbSteps objet
 ....
}
```

qui construit un chemin aléatoire de nbSteps étapes (nbSteps point) de la façon suivante :

• alloue un tableau de nbSteps points dans le tas (allocation dynamique)

• affecte des abscisses aléatoires (minX, maxX) et des ordonnées aléatoires (minY, maxY) à chaque point du chemin

Pour créer un tableau d'objet, utiliser la syntaxe suivante :

Point

4. Proposer une fonction

```
void showRandomPath(Point1* pathPoints, int nbSteps)
```

qui permet d'afficher tous les coordonnées du chemin **`pathPoints`** qui est composé de **`nbSteps`**

Exemple d'affichage:

```
Le chemin aléatoire généré est :
{(1.37181,3.21904)(1.21564,4.19901)(1.1077,2.81546)(1.16429,4.16264)(1.42633,2.82834)}
```

5. Proposer une fonction

```
double followPath(Point1* point, Point1* path, int nbSteps)
```

qui déplace le point **`point`** le long du chemin **`path`** ayant **`nbSteps`** étapes, et qui renvoie la distance parcourue.

Exemple d'affichage:

```
Le chemin aléatoire généré est :
{(2.42298,2.1448)(1.71815,3.79864)(1.50033,2.5529)(1.30413,3.68595)(1.31598,4.96462)}
Les coordonnées du point E après le déplacement dans le chemin aléatoire
est : (1.31598,4.96462) et la distance parcourue est : 13.2224
```

Exercice 3 :

Proposer une classe "**JeuMorpion**" qui permet, via ses attributs et ses méthodes, de manipuler le jeu ti-tac-toe

Ci-dessous une description incomplète de cette classe :

```
#ifndef EXSUPP_JEUMORPION_H
#define EXSUPP_JEUMORPION_H

enum Content {VIDE, ROND, CROIX};

class JeuMorpion {
private:
   int grille[3][3];
   bool placer_coup(int ligne, int colonne, Content c);
public :
   void init();
   bool placer_rond(int ligne, int colonne);
   bool placer_croix(int ligne, int colonne);
   // à compléter
};


#endif //EXSUPP_JEUMORPION_H
```