Travaux Dirigés de C++ n°4

Cours de programmation

—Licence d'informatique—

Les constructeurs et l'allocation dynamique

Dans ce TD, nous détaillons les différents constructeurs et nous présentons les opérateurs d'allocation `new` et de désallocation `delete`.

▸ **Exercice 1. (Allocation dynamique)** *Écrire un programme qui saisit un entier* `n` *et qui alloue dynamiquement un tableau de* `n` *caractères pour saisir ensuite une chaîne de caractères de cette taille. Pour cela, on utilisera l'opérateur* `new`*.*

Après avoir affiché cette chaîne, on libérera la mémoire associée à cette chaîne.

▸ **Exercice 2. (Allocation dynamique)** *Allouer un tableau d'entiers à deux dimensions représentant une matrice* `n` $\times$ `m` *où* `n` *et* `m` *sont des entiers entrés par l'utilisateur. On initialisera la matrice à* `0`*, on l'affichera, puis on libérera la place mémoire associée.*

▸ **Exercice 3. (Constructeurs)** *Dans l'exercice 2 de la feuille 3, lorsqu'on déclare :*

```
Point p;  // ou
Point* ptr = new Point;
```

quel est le mécanisme mis en œuvre ?

Transformer les méthodes `init` *de la classe* `Point` *en des constructeurs.*

Donner une définition explicite du constructeur par défaut. Quels problèmes présentent les initialisations par défaut des paramètres dans le cadre des constructeurs ?

▸ **Exercice 4. (Constructeurs par recopie)**

De la même façon que pour la classe `Point`*, transformer les méthodes* `init` *de la classe* `Rectangle` *en des constructeurs.*

Pour pouvoir utiliser directement un point, passé en argument d'un constructeur de rectangle, lors de l'initialisation des champs d'un rectangle, ajouter un constructeur par recopie dans la classe `Point`*.*

▸ **Exercice 5. (Variables et méthodes statiques)** *Ajouter, dans la classe* `Point`*, un champ privé* `nbCrees`*, de type* `int`*, déclaré comme étant* `static`*. Ce champ doit représenter le nombre total de fois ou un point à été crée. Il sera initialisé à* `0`*.*

Ajouter également à la classe `Point` *une méthode* `getNbCrees()` *qui retourne la valeur de ce champ. Cette méthode sera également déclarée* `static`*.*

Comment attribuer un numéro unique à chaque nouveau point créé ?

▸ **Exercice 6. (Liste d'initialisation)**

Avec les constructeurs de `Point` *et de* `Rectangle` *introduits dans les exercices précédents, que produit l'exécution du code suivant ?*

```
Point p1;
Point p2;
p2.print();
cout << "Nbre de points crees : " << Point::getNbCrees() << endl;
Rectangle r(p1,p2);
cout << "Nbre de points crees : " << p1.getNbCrees() << endl;
```

Modifier le constructeur à deux arguments de `Rectangle` *pour le définir par une liste d'initialisation. Quel serait alors l'affichage produit par l'exécution du code précédent ?*

▸ **Exercice 7. (Matrices)**

Définir une classe `Matrice` *permettant de représenter des matrices d'entiers à deux dimensions. La classe devra comporter :*

- *un constructeur acceptant la taille de chaque dimension ;*
- *un constructeur acceptant, en plus des tailles de chaque dimension, une valeur d'initialisation des éléments de la matrice ;*
- *un constructeur par recopie ;*
- *un destructeur ;*
- *une méthode* `get(unsigned i, unsigned j)` *acceptant un numéro de ligne* `i` *et un numéro de colonne* `j` *et retournant une référence à l'élément correspondant dans la matrice.*
- *une méthode* `print()` *affichant la matrice à l'écran.*

L'allocation de la place mémoire nécessaire au stockage des éléments de chaque matrice sera prise en charge dans les constructeurs ; sa libération sera effectuée dans le destructeur.