# Corrigé du TP7

**Disclaimer :** Ce corrigé n'a pas pour but de vous fournir un détail des algorithmes répondant aux questions de l'énoncé, mais de vous résumer les notions clés dont vous aurez besoin pour résoudre ces exercices, les avertissements vis à vis de certaines erreurs classiques et les détails des passages les plus techniques, voire des précisions si vous souhaitez aller plus loin sur ces questions et plus généralement de synthétiser dans un même document la plupart des commentaires fait en TP afin qu'ils puissent profiter à tous. Pour obtenir un corrigé détaillé, il vous faudra donc vous référer à ce que vous avez pu faire en TP ou reprendre vous même les exercices à partir des notions présentées dans le présent corrigé pour écrire les algorithmes qui vous manquent.

En cas de difficultés, **n'hésitez pas à me contacter**, nous pourrons en discuter soit par mail soit en début de séance suivante. Ce corrigé n'a pour but que d'être une base de travail pour vous et de servir de complément par rapport à ce que vous avez fait en séance, il n'est en aucun cas fait pour se suffire à lui même.

## Écrire une classe

* Une classe est un type de variables constituées de plusieurs attributs (ou champs). Ces attributs peuvent être de type différent, contrairement aux tableaux ou vecteurs.

  Généralement, pour définir une classe, on utilise trois fichier que l'on appelle nomClasse.h (un header file) , nomClasse.cpp et le fichier main.cpp que vous utilisez habituellement.

  La grosse différence entre une classe et une structure est que les éléments de la structure sont protégés. C'est à dire que vous ne pouvez utiliser que les fonctions que vous avez définis sur les attributs de la classe, vous ne pouvez pas les manipuler comme des éléments classiques.

  Cette différence peut sembler inutilement complexe, mais c'est elle qui permet de faire de la programmation orientée objet et elle permet de limiter les erreurs dans de longs codes, surtout lorsque plusieurs personnes travaillent dessus.

* Pour créer le fichier .h il vous faut, lorsque vous créer un nouveau fichier, sélectionner le C/C++ header. Dans ce fichier doit figurer trois choses entre la deuxième et troisième ligne de code déjà présentes dans le fichier à sa création :

  Les bibliothèques que vous allez utiliser.

  La définition de la classe avec la syntaxe suivante :

  **class** nomClasse

  {

  **private :**

  type nomAttribut ;

  **public :**

  entête de fonctions membres ;

  } ;

  N'oubliez pas les ;

  Toutes les fonctions utilisant les parties des éléments de la classe doivent être à l'intérieur de la définition de la classe. On appelle ces fonctions des méthodes (ou fonctions membres).

  Vous pouvez également définir des fonctions en dehors de la classe, mais ces fonctions ne pourront pas utiliser autre chose qu'un élément de la classe et les fonctions membres. En particulier, vous ne pouvez pas utiliser les parties composant un élément de la classe.

  Dans la partie private figurent des éléments que vous ne pourrez utiliser que dans les méthodes en partie public. Dans la partie public figurent des éléments qui pourront utiliser les éléments de la partie private et pourront être utilisés dans la fonction main.

* Pour créer le fichier .cpp il vous faut, lorsque vous créer un nouveau fichier, sélectionner le C/C++ source. Dans ce fichier, vous définirez vos fonctions.

  Dans le fichier nomClasse.cpp il n'est pas nécessaire de séparer les fonctions membres faisant parti de la

définition de la classe et les fonctions définies en dehors de la classe.

Attention à bien écrire #**include"nomClasse.h"** au début de votre programme, sinon cela ne marchera pas.

* Pour que votre fonction main reconnaisse votre classe, vous devez écrire #**include"nomClasse.h"** au début de votre programme.

  Attention, vous n'avez pas besoin de mentionner le fichier .cpp , seulement le .h

### Manipuler les fonctions membres

* Les fonctions membres sont à définir dans la partie publique de la partie classe.
* Dans le fichier .cpp, vous pouvez utiliser les attributs de vos classes en utilisant la syntaxe : nomVariable.nomAttribut

  À l'inverse, si la fonction n'est pas incluse dans la définition de la classe, vous ne pourrez pas manipuler les attributs de la classe, sauf si vous avez créé une fonction membre spécifiquement pour ça (chose qu'il vaut mieux éviter tant que faire se peut).

### Surcharge des opérateurs

* Par défaut, les opérateurs +, −, ∗, /, mais aussi + =, − =, >> et <<, ne fonctionnent pas pour les éléments de la classe.

  Pour pouvoir les utiliser, il faut surcharger les opérateurs, c'est-à-dire les redéfinir pour les éléments de votre classe.
* Pour surcharger les opérateurs, commencez par définir l'entête de la surcharge d'opérateur dans la partie publique de la classe.

  La syntaxe classique est : type operator+(type nomVariable) ;

  Pour les opérateurs >> et <<, la syntaxe est différente, on considère pour variables des flux :
  ostream& operator<<(ostream &nomFlux, const type &nomVariable) ;
  istream& operator>>(istream &nomFlux, type &nomVariable) ;
* Vous pouvez alors définir le nouvel opérateur dans le fichier .cpp.

  Vous pouvez alors utiliser la syntaxe nomVariable.nomAttribut pour la variable à droite de l'opérateur et nomAttribut seulement pour la variable à gauche de l'opérateur.
* Il est possible dans le cas de la surcharge d'opérateur d'utiliser le pointeur **this**. Ce pointeur renvoie à l'objet lui même.

  On l'utilise donc souvent pour la surcharge des opérateurs +=

  Il suffit en effet d'effectuer les calculs sur l'objet avant de les renvoyer à l'aide de la commande :
  return ∗this ;

### Utiliser des constructeurs

* Pour définir un objet d'une classe, il va falloir utiliser des constructeurs. Un constructeur est une méthode qui est utilisée automatiquement dès que l'on définit un objet de la classe.
* Un constructeur est une méthode qui doit avoir le même nom que la classe et n'a pas de type de retour, c'est à dire qu'on ne met rien (même pas void) avant le nom du constructeur.
* Dans le fichier nomClasse.cpp, il y a deux façons de définir un constructeur :

```
nomClasse : : nomClasse()
{
nomAttribut1=valeurAttribut1parDefaut ;
nomAttribut2=valeurAttribut2parDefaut ;
nomAttribut3=valeurAttribut3parDefaut ;
}
```

```
nomClasse : : nomClasse() : nomAttribut1=valeurAttribut1parDefaut, nomAttribut2=valeurAttribut2parDefaut, nomAttribut3=valeurAttribut3parDefaut
{
}
```

* Il est également possible de surcharger le constructeur et de permettre ainsi à l'utilisateur de la classe de spécifier des valeurs pour définir les objets de la classe.

### Quelques conseils pour retravailler le TP

* Le gros intérêt des classes est de restreindre ce que vous pouvez faire sur les éléments de vos classes, vous êtes donc sûr que votre programme ne peut faire que ce que vous avez programmé et ne surinterprète pas.

  Cette façon de faire est contre intuitive et va vous demander du travail pour le maîtriser, mais il est important que vous maîtrisiez les classes, c'est une part importante du programme de ce semestre, il est donc important que vous maîtrisiez ce TP.
* Une grosse partie de la difficulté de la manipulation des fonctions membres vient de la difficulté à comprendre les classes, la syntaxe devrait venir rapidement avec la pratique.
* La surcharge d'opérateur est une compétence importante pour la suite du semestre.
* Par la suite, nous utiliserons beaucoup les classes, il est donc important de bien maîtriser ces compétences pour les tps suivants.