TP 2

1 Anatomie d'une classe

Finir avec soin cet exercice du TP1. Il est recommandé d'ajouter l'affichage d'une ligne vide entre chaque création d'objet dans le *main*, pour bien voir quelles opérations sont appelées de manière automatique. N'oubliez pas d'écrire vos réponses (sur une feuille ou en commentaire dans votre fichier source).

2 Constructeurs, destructeurs et affectation des classes dérivées

Dérivez la classe *LaClasse* de l'exercice précédent en *LaClasseSpecialisee* et munir la classe obtenue de constructeurs, destructeur et opérateur d'affectation laissant une trace d'affichage. Regardez ce qu'il se passe :

- quand vous omettez de définir des constructeurs dans *LaClasseSpecialisee*,
- quand vous omettez de définir un constructeur par copie,
- puis quand vous oubliez de recourir aux listes d'initialisation dans la définition des constructeurs autres que le constructeur par défaut (si jamais vous en avez absolument besoin, n'hésitez pas à passer en *protected* les données membres de la classe de base).

Est-ce bien cohérent avec ce que vous avez vu en cours ?

Amusez-vous à tester les possibilités d'`upcast` et `downcast` de pointeurs et de références au sein de votre hiérarchie de classe *LaClasse*.

3 Un module *String* fait main

Coder avec soin le module *String* proposé au TP1 en vous servant de la classe *LaClasse* pour modèle de syntaxe des éléments qu'on peut trouver dans une classe. Assurez vous de votre gestion saine de la mémoire avec `valgrind`. Bien entendu le but n'est pas que vous recouriez à un conteneur de la STL, mais que vous gériez vous même l'allocation dynamique de la mémoire nécessaire au stockage.

4 Comparatif Java et C++

Avez-vous fini cet exercice ? Quelles sont vos conclusions en terme d'efficacité ? Essayez de comprendre ce qu'il se passe en mémoire en réalisant un dessin de l'occupation mémoire d'une Image en JAVA et en C++ respectivement.