Projet : étape 2 (semaine 3 du semestre)

Buts

Nous allons cette semaine commencer à utiliser la compilation séparée (eh oui, ça devait arriver !..).

Préliminaires :

Je vous rappelle qu'il est prévu une répartition des tâches du projet au sein du binôme. Je considère que chacun doit faire environ la moitié du travail lié au projet. Essayez donc de rapidement vous organiser, concevoir correctement la répartition et commencez à faire un `Makefile` (sujet du jour).

Par ailleurs, **avant** de vous lancer directement dans la réalisation de la nouvelle partie du projet, je vous conseille de vous assurer de bien avoir compris les concepts présentés en cours, par exemple en faisant **au préalable quelques exercices** si nécessaire.

Cette première partie du projet (de la semaine passée à la semaine prochaine) est assez progressive de sorte à vous permettre de vous organiser et prendre vos marques dans ce projet. Profitez-en donc pour faire les choses correctement dès le départ (cela permet de gagner du temps par la suite) et évitez absolument de prendre du retard... Je vous conseille d'avancer le plus possible en synchronisation avec le planning prévu, ou alors avec une semaine de décalage pour bien assimiler le cours, mais certainement pas plus.

Pour terminer avec les conseils et remarques préliminaires, n'oubliez pas de revenir de temps en temps consulter **la page de description générale du projet** et **la page d'administration** afin de situer votre travail courant par rapport au projet dans son ensemble.

[2*] Exercice P3 : Modularisation

Il est donc temps de se mettre à la compilation séparée... Voici un exercice censé vous aider à le faire.

L'idée est d'avoir un fichier séparé pour chaque classe, plus un fichier pour le programme principal (et un/des fichier(s) pour les tests des classes).

Commençons donc avec la classe `Vecteur`.

Si vous ne l'avez pas encore séparé en trois fichiers, copiez le fichier `Vecteur.cc` en `Vecteur.h`. Et copiez-le encore une fois en `testVecteur.cc`. Vous avez donc maintenant trois fois le même fichier (même contenu) mais avec trois noms différents.
Évidemment, nous allons maintenant modifier chacun de ces fichiers séparément.

Dans le fichier `testVecteur.cc` :

1. supprimez toutes les définitions de la classe `Vecteur` et de ses méthodes ;
2. ne gardez que le `main` et les `#include` nécessaires *ici*, c.-à-d. nécessaires à ce *seul* fichier ;
3. ajoutez `#include "Vecteur.h"`.

Dans le fichier `Vecteur.h` :

1. ne gardez que la définition de la classe `Vecteur` ;
   supprimez aussi le `#include <iostream>` (sauf, comme évoqué plus haut, si vous l'utilisez dans *ce* fichier, ce qui ne devrait pas être le cas à ce stade, mais peut être avez vous déjà pris de l'avance sur le projet) et le `using namespace` (ça c'est obligatoire !) ;
   Comme expliqué en cours, **IL NE DOIT JAMAIS Y AVOIR DE `using namespace` dans un fichier `.h` !** (malus)
2. supprimez les définitions des méthodes (elles resteront dans le `.cc`)
   (Remarque avancée : sauf celles en 1 seule ligne, maximum, que vous pouvez garder ici, si jamais) ;
3. ajoutez

   ```
   #pragma once
   ```

   en **tout début** de fichier ;
   ceci sert à éviter qu'un même fichier `.h` soit inclu par mégarde plusieurs fois dans une même compilation (ça arrive ... surtout quand votre projet va augmenter de taille !).

Dans le fichier `Vecteur.cc` :

1. ne gardez que les définitions des méthodes ;
2. « extériorisez » (hors de la classe) ces définitions en ajoutant `Vecteur::` devant le nom de la méthode ;
3. ajoutez `#include "Vecteur.h"` et supprimez tous les `#include` non strictement nécessaire à ce fichier ci.

Créez maintenant le fichier `Makefile` permettant de construire (make) votre projet à partir de ses constituants.

Je vous donne ci-dessous un exemple simple, que vous pouvez **télécharger en suivant ce lien**, mais je vous recommande néanmoins d'aller voir le **tutoriel sur les Makefile** pour en savoir plus :

```
CXX = g++
CC  = $(CXX)

CXXFLAGS = -std=c++11  # C++11, ou autre suivant vos préférences

# Partie commentée : choisissez les options que vous voulez avoir
#                    en décommentant la/les lignes correspondantes
#
CXXFLAGS += -pedantic -Wall         # pour les pur(e)s et dur(e)s
# CXXFLAGS += -g                    # pour debugger
# CXXFLAGS += -O2                   # pour optimiser la vitesse

all: testVecteur

Vecteur.o: Vecteur.cc Vecteur.h

testVecteur.o: testVecteur.cc Vecteur.h

testVecteur: testVecteur.o Vecteur.o
```

Une fois tout ceci fait, tapez simplement `make` (dans un terminal se trouvant dans bon répertoire (`~/myfiles/cpp/projet`)) et...
..oh miracle ! Ça marche. (enfin j'espère pour vous !)
Vérifiez en lançant votre programme : `./testVecteur`

Dernière mise à jour le 7 février 2025
Last modified: Fri Feb 7, 2025