Examen du cours I04 (AMS 2016-2017)

Marc Tajchman et Christophe Labourdette

14 novembre 2016

1 Indications

Cet examen a pour but de vous évaluer à la fin du cours I04, voici quelques points qui seront particulièrement important.

- Dans la mesure du possible on essaiera d'écrire du C++ moderne utilisant par exemple des fonctionnalités du standard C++11,
- On privilégiera toujours la libraire standard et la STL
- Les programmes doivent compiler et s'exécuter, un programme qui ne compile pas ne sera a priori pas noté.
- Les programmes peuvent et doivent être commentés.
- Les documents et la consultation sur internet est autorisée mais attention, la recopie flagrante ou le plagiat d'un programme existant sera sanctionnée, de même que les communications avec un autre candidat ou une personne extérieure.
- Les remarques et les instructions particulières seront plaçées dans un fichier "Readme".
- Le fichier Makefile fourni, permet de compiler question par question. Par exemple pour la deuxième question du carré : "make carre2". L'exécutable s'appelle alors carre2.exe.
  Le makefile permet également de supprimer les exécutables : "make clean".
- Les fichiers sont fournis sous forme d'archive, pour l'extraire vous pouvez, une fois copiée l'archive dans votre répertoire de travail, exécuter la commande :

  ```
  tar xvf Examen_I04_2016.tar
  ```

  Cela créera alors un répertoire Examen_I04_2016 que vous pourrez renommer selon la consigne suivante.
- Les fichiers réponses à votre examen doivent se trouver dans un répertoire, celui-ci devra être dans votre répertoire de travail et s'appeler "Examen_I04_2016_Nom_Prenom".
  (En remplaçant bien entendu Nom et Prenom par les votres)

- Lorsque vous aurez terminé vous devrez créer une archive (APRES AVOIR SUPPRIME LES EXECUTABLES), par exemple avec la commande suivante :

  ```
  cd ; tar cpfz I04_2016_Nom_Prenom.tgz I04_2016_Nom_Prenom
  ```

  Ensuite vous devrez copier cette archive sur la clé du surveillant et envoyer ce fichier par mail, a Marc Tajchman et Christophe Labourdette.

2 Des carrés

On considère disposer d'un carré (matrice), composé de valeurs binaires ou de flottants entre 0 et 1. La taille du carré est un paramêtre. Dans un premier temps toute la classe sera déclarée dans une partie public.

1. Proposez une classe Carre, template, comportant, dans un fichier carre1.hpp :
   — un constructeur par défaut (la taille sera 1),
   — un constructeur prenant la taille du carré,
   On surchargera également, à l'extérieur de la classe, l'opérateur "«" pour afficher le carré.
   On utilisera main1.cpp pour tester cette question.
2. A partir de cette question, on placera les données dans une partie privée.
   Ajouter (on utilisera à présent un fichier carre2.hpp) :
   — un constructeur par recopie,
   — un opérateur d'assignation,
   — une fonction put permettant de changer la valeur en i,j,
   — une fonction get permettant de récupérer la valeur en i,j,
   — une fonction taille() qui renvoie la taille du carré.
   On n'oubliera pas de modifier l'opérateur "«" maintenant que les données sont privées. On testera à l'aide du fichier main2.cpp
3. Dans un fichier carre3.hpp on ajoutera les surcharges des opérateurs +,-,*, qui effectuent les opérations entre deux carrés élément par élément.
   On testera à l'aide du fichier main3.cpp
4. Implémentez une fonction membre moyenne qui prend pour chaque point du carré la moyenne du point avec les 4 voisins les plus proches (dans le cas entier on prendra l'entier le plus proche). Pour les bords on proposera deux méthodes pour remplacer les données manquantes sur les bords, "miroir", on prend la valeur du point symétrique, "tore", on prend la valeur en considérant le bord comme collé au bord opposé.
   La méthode par défaut sera "miroir"
5. Dans un fichier carre5.hpp, ajouter la surcharge des opérateurs * et *= , par un élément de type T.
   On testera ensuite avec le fichier main5.cpp.

3 Etude de texte

On considère disposer d'un petit texte dans un fichier (texte_93.txt et texte_machine.txt).
On essaiera toutes les questions avec les deux fichiers textes fournis. A chaque étape le fichier .cpp est une incrémentation du fichier de la question précédente.

1. Dans le fichier texte1.cpp, écrire une petite fonction qui lit le fichier et place les mots dans une structure de type std :: vector<std::string>
   std :: vector<std::string>& lire_fichier(std :: string nom_fichier) Ecrire un programme main qui utilise la fonction précédente et affiche le texte lu avec un mot par ligne.
2. Dans un fichier texte2.cpp, écrire une petite fonction std :: vector<string>& minus(std::vector<string>&) qui transforme un texte en minuscule.
   Ajouter au main, une partie qui affiche le texte lu en minuscule avec un mot par ligne
3. Dans un fichier texte3.cpp, écrire une petite fonction std :: vector<string>& deponctue(std::vector<string>&) qui remplace toute ponctuation par un espace.
   Ajouter dans le main une partie qui teste la fonction prècèdente sur le texte lu, transformé en minuscule et qui affiche un mot par ligne
4. Dans un fichier texte4.cpp, écrire une fonction qui remplit des structures std :: map<std::string,int> et std :: map<char,int> pour compter les occurences des mots et des caractères du texte
   void compte(std::vector<std::string>&, std::map<std::string,int>&, std::map<char, int>&) Ajouter au main une partie qui affiche le compte des mots et des caractères trouvés (en minuscule et sans ponctutation).