Examen de rattrapage du cours I04 (AMS 2018-2019)

Christophe Labourdette

Mai 2019

1 Indications

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

- 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,
- 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 placées dans un fichier "Readme".
- Le fichier Makefile fourni, permet de compiler question par question. Par exemple pour la deuxième question : "make phrase2". L'exécutable s'appelle alors phrase2.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 Rattrapage_I04_2018.tar
  ```

  Cela créera alors un répertoire Rattrapage_I04_2018 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 "Rattrapage_I04_2018_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 FICHIERS EXECUTABLES), par exemple avec la commande suivante :

  ```
  make clean;
  cd ; tar cpfz I04_2018_Nom_Prenom.tgz Rattrapage_I04_2018_Nom_Prenom
  ```

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

  (Christophe.Labourdette (at) cmla.ens-cachan.fr)

2 Des mots

On considère disposer d'un alphabet, autrement dit un ensemble de lettre (char). A l'aide duquel nous allons construire des mots. Dans un premier temps toute la classe sera déclarée dans une partie publique.

1. Proposez une classe Alphabet utilisant une liste (ordonnée en utilisant $<$ et sans doublon), comportant, dans un fichier phrase1.hpp, en plus des données (placées dans une partie privée) :

   — un constructeur par défaut (comprenant les lettres de l'alphabet),
   — un constructeur prenant en argument un vecteur de lettres quelconques (il pourrait y avoir des doublons)

   On surchargera également, à l'extérieur de la classe, l'opérateur "«" pour afficher l'alphabet.
   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 phrase2.hpp) :

   — un constructeur par recopie,
   — un opérateur d'assignation,
   — une fonction put_alphabet permettant de changer le ième caractère de l'alphabet,
   — une fonction get_alphabet permettant de récupérer le vecteur,
   — une fonction taille() qui renvoie la taille de l'alphabet.
   — On construira une fonction booléenne teste_mot prenant en argument une string et renvoyant vrai si le mot ne comprend que des lettres de l'alphabet et faux sinon.

   On n'oubliera pas de modifier l'opérateur "«" maintenant que les données sont privées. On testera à l'aide du fichier main2.cpp
3. Nous allons à présent travailler sur une classe Dico, comprenant des mots (string), composés de caractères provenant d'un alphabet donné. Dans un fichier phrase3.hpp, on construira la classe Dico autour d'une liste de string avec :

   — un constructeur par défaut (l'alphabet sera alors l'alphabet standard) vide,
   — un constructeur prenant en argument, un alphabet et un vecteur de string, on vérifiera que les mots ne contiennent que des caractères de l' alphabet donné, sinon ils ne seront pas inclus
   — une fonction aff_dico qui affiche la liste des mots contenus, avec un mot par ligne,
   — une fonction aff_alphabet, qui affiche l'alphabet

   On testera à l'aide du fichier main3.cpp
4. Pour cette question on ajoutera :

   — une fonction booléenne presence_mot prenant en argument une string et renvoyant vrai si le mot est présent et faux sinon
   — une fonction booléenne insere_mot prenant en argument une string et l'insérant à sa place (ordre alphabétique) si elle n'est pas déjà présente, elle renverra vrai en cas d'insertion et faux sinon,
   — une fonction booléenne efface_mot prenant en argument une string et la supprime si elle est présente, elle renverra vrai en cas de suppression et faux sinon,

   On testera cette question à l'aide du fichier main4.cpp