Haute École Bruxelles – Brabant

HE2B ESI

Informatique de gestion
Informatique et systèmes

Année 2020 – 2021
5 janvier 2021

DEV3 : examen machine oral

M.-A. : ABS, BEJ, *NVS*, PBT & PHA

Vecteurs

Durée : 3h00.

Directives

Les directives d'ordre général sont données en fin d'énoncé. Veuillez les lire et respecter. En particulier celle relative à l'obligation de *commit* / *push* dans votre dépôt gitlab tout au long du déroulement de l'examen.

Ce qui vous est fourni

L'archive compressée `m_vector.7z` renferme deux répertoires :

- le dossier `src` où se trouvent :
  - le dossier `random` contenant le fichier `random.hpp` où sont implémentés des fonctions et modèles de fonctions pour la génération de nombres pseudo-aléatoires, utilisés par la fonction `data()` ;
  - les fichiers `data.h` et `data.cpp` où est définie l'énumération fortement typée `Randomness` et où est implémentée la fonction `data` produisant des données brutes à utiliser au cours de l'examen ;
  - le fichier d'en-têtes `m_vector_incomplete.h` ainsi que son homologue sans documentation `m_vector_incomplete_without_doc.h` dont l'un des deux doit être complété lors de cet examen ;
- le dossier `html` où se trouve la documentation de l'énumération `Randomness`, de la fonction `data()`, des fonctions du fichier `random.hpp` et de la classe `m_vector` et des fonctions y attachées.

Au cours de l'examen, parmi les fichiers fournis, `m_vector_incomplete.h` *ou* `m_vector_incomplete_without_doc.h`, selon votre choix, est *le seul* que vous avez le droit de modifier, et ce uniquement aux endroits mentionnés. Si vous les modifiez ailleurs

que là où autorisé ou si vous apportez des modifications à d'autres fichiers fournis, vous en êtes pénalisé.
Les deux fichiers d'en-têtes `m_vector_xxx.h`[1] ont même contenu pour le (pré-)compilateur C++. Ils diffèrent uniquement par leur contenu pour doxygen.

Ce qui vous est demandé

1. (2 pts) Créez un projet console C++17 au sein de Qt Creator[2].
   1) Nommez ce projet m_vector_xxxxx où vous remplacez xxxxx par votre numéro d'étudiant.
   2) Copiez dans le répertoire du projet le contenu du répertoire `src` extrait de l'archive `m_vector.7z`.
   3) Ajoutez à votre projet le fichier source `data.cpp` et les fichiers d'en-têtes `data.h`, `random.hpp` et *soit* `m_vector_incomplete.h`, *soit* `m_vector_incomplete_without_doc.h`[3].
   4) Dans le fichier source qui contient la définition de la fonction principale, incluez le fichier `m_vector_xxx.h` que vous avez choisi d'utiliser[1]. Lors de la compilation, trois erreurs sont produites. Elles apparaissent aux endroits marqués `// FIXME : erreur de compilation ici`. Vous pouvez, dans un premier temps, mettre ces trois définitions de méthodes et fonction en commentaire, de sorte que le projet compile et ainsi ne pas être bloqué dans les tests du code que vous écrivez.
      Au cours de la suite de l'examen, vous devez, entre autres choses, résoudre ces erreurs de compilation, mais aussi vous débarrasser des très nombreux avertissements du compilateur. Idéalement, le projet remis compile et ne génère aucun avertissement.

m_vector (55 pts)

2. (1 pt) Renommez le fichier `m_vector_xxx.h` que vous avez choisi d'utiliser[1] en `m_vector.h`.
   Déplacez la classe `m_vector` et les fonctions qui y sont associées depuis l'espace de nommage `he2b::nvs` vers l'espace de noms `he2b::g`*xxxxx*, où *xxxxx* est votre numéro d'étudiant.

3. (54 pts) Complétez le fichier `m_vector.h` aux vingt-cinq endroits marqués par le commentaire `// TODO`, dont trois étiquetés également `// FIXME`.
   Veillez à ce que votre code respecte les fonctionnalités telles que décrites dans la documentation fournie.

Fonction principale (28 pts)

4. (15 pts) Invoquez la fonction `data()`. Instanciez un `std::vector` de `m_vector`. Utilisez chaque `std::tuple` retourné par `data()` pour peupler ce `std::vector` en respectant les indications de la documentation de la fonction `data()` et en

1 C'est-à-dire *soit* `m_vector_incomplete.h`, *soit* `m_vector_incomplete_without_doc.h`.
2 Ou à défaut si vous n'utilisez pas Qt Creator, un ensemble de fichiers d'en-têtes et de fichiers sources ainsi qu'un fichier `makefile` compatible C++17. Sans ce dernier, si vous ne fournissez pas de fichier `pro` pour Qt Creator, votre travail n'est pas pris en compte.
3 Il ne faut *pas* donc inclure *les deux* fichiers `m_vector_incomplete.h` et `m_vector_incomplete_without_doc.h`, *mais* uniquement *un des deux*, à votre bon choix ; ils sont équivalents pour ce qui concerne le contenu C++.

considérant et comptant dans l'ordre indiqué ci-après les erreurs possibles dans chaque `std::tuple` :

1) erreur dans le format de la `std::string` censée représenter un entier non signé ;
2) la `std::string` indique zéro composante ;
3) le nombre de composantes indiqué par la `std::string` diffère de celui du `std::vector`.

Seuls les `std::tuple` sans erreur sont utilisés pour garnir le `std::vector` de `m_vector`.

5. (2 pts) Affichez sur la sortie standard :
   1) le nombre d'erreurs de format de `std::string` dénombrées au point précédent ;
   2) le nombre de `std::string` renseignant zéro composantes comptées au point précédent ;
   3) le nombre de `std::string` indiquant un nombre de composantes différent de celui du `std::vector` recensées au point précédent ;
   4) le nombre de `std::tuple` sans erreur comptabilisés au point précédent.

6. (2 pts) Affichez sur la sortie standard le contenu du `std::vector` de `m_vector` garni précédemment, avec un titre adéquat.

7. (3 pts) Créez un `std::vector` de pointeurs *ou* de `std::reference_wrapper` de `const m_vector` et faites en sorte que le premier élément de ce nouveau `std::vector` pointe / référence le premier élément du `std::vector` de `m_vector`, le deuxième le deuxième, etc., jusqu'au dernier élément du nouveau `std::vector` pointant / référençant le dernier élément du `std::vector` de `m_vector`.
   Il doit être impossible de modifier les `m_vector` pointés / référencés à travers les pointeurs / `std::reference_wrapper` nouvellement créés.

8. (4 pts) Triez le nouveau `std::vector` de pointeurs / `std::reference_wrapper` en majeur dans l'ordre croissant du nombre de composantes et en mineur dans l'ordre croissant de la norme des `m_vector` pointés / référencés.
   Pour une explication du tri en majeur / mineur, rendez-vous ici : https://openclassrooms.com/forum/sujet/trier-un-tableau-en-majeur-28231.

9. (2 pts) Parcourez le `std::vector` trié de pointeurs / `std::reference_wrapper` et affichez sur la sortie standard les `m_vector` pointés / référencés, précédé d'un titre adéquat.

Bon travail !

Directives

- *Vous devrez effectuer un* commit *et un* push *à chaque fois que vous passez d'une question à une autre*. Par exemple, si vous passez à la question 7. après la 5. en sautant la 6.
  Les remises qui ne respectent pas cette consigne ne sont pas corrigées.
- Tous les fichiers sources et d'en-têtes que vous utilisez lors de cet examen doivent obligatoirement se trouver dans le répertoire ou un sous-répertoire du projet.
- Le code que vous rédigez doit être conforme à la norme C++17 (ISO/IEC 14882:2017) telle que prise en charge dans les laboratoires.
- Il importe avant tout de bien lire l'énoncé et de respecter rigoureusement ce qui y est demandé.
- L'examen est bien évidemment individuel ! Toute tentative de fraude, par quelque moyen que ce soit, est sanctionnée.
- Vous pouvez disposer de vos notes. Vous pouvez accéder à l'internet en lecture et en écriture. Cependant, les seuls textes que vous pouvez rédiger en surfant sont destinés à des machines, les moteurs de recherche, pour spécifier les termes de celle-ci, jamais à des êtres humains.
- En cas de problème, c'est la dernière version *pushée* dans votre dépôt *avant* l'heure de fin d'examen qui sert à l'évaluation.