Exercice IX

Devoir Surveillé sur Table : programmation de classe et nombres aléatoires

Objectif

Développer en C++ un programme qui simule les affichages aléatoires d'images.
Pour ce faire, on partira d'un code existant disponible dans l'archive **ds2-make-slideshow.tar.gz**. (Voir fichier joint.)
L'archive contient un Makefile, au moins avec les cibles suivantes :

- '`make`' : lance toutes les compilations (source + test)
- '`make runtest`' : lance l'exécution des tests
- '`make run`' : lance l'exécution du programme principal
- '`make cleaner`' : lance l'effacement de tous les fichiers et dossiers générés

Pour rendre votre travail, vous devez fournir une archive contenant le dossier de travail après exécution de la commande '`make cleaner`'.

1. Contexte

Le code existant contient :

- `Makefile`
- les fichiers sources :
  - `src/SlideShow.cpp`
  - `src/main.cpp`
- les fichiers d'en-tête :
  - `inc/SlideShow.hpp`
  - `test/catch.hpp`
- le fichier de tests
  - `test/SlideShowTest.cpp`
- les fichiers de données :
  - `data/selection.txt`
  - `test/data/small.txt`

Le fichier `data/selection.txt` contient une liste de noms de fichier image, avec un nom par ligne.
Le fichier `test/data/small.txt` est un extrait de ce fichier avec que 10 noms pour effectuer des tests.

On souhaite que le programme principal tienne compte de l'argument passé à l'exécution - chemin vers le fichier contenant la liste de noms de fichier image.
L'affichage aléatoire - fonction membre de la classe `SlideShow` - affichera une ligne du fichier de façon aléatoire.
Le **code existant est incomplet**. Tous les tests ne passent pas avec succès et l'exécution du programme principal ne fait rien. il s'agira de compléter le code - là où c'est marqué "**TODO**" en commentaire - afin d'atteindre l'objectif du programme et avoir des tests qui passent avec succès.

2. Compilation et test du code existant

2.a. Lancez une compilation complète, puis lancez l'exécution des tests avec '`make runtest`'. Que constatez-vous ? Complétez l'implémentation de la fonction membre `init` de `SlideShow` pour que le "Test Number Of slides" passe avec succès.

2.b. Complétez l'implémentation de la fonction membre `getRandomNumber` de `SlideShow` pour que le "Test Random Number" passe avec succès. On souhaite que les nombres aléatoires retournés par cette fonction soit compris entre **1** et le **nombre de lignes** du fichier transmis en paramètre de construction de l'objet `SlideShow`. Modifiez le code source afin que ce soit le cas. Également, modifiez le nombre dans la macro `CHECK` du test pour que quelque soit le nombre retourné par `getRandomNumber`, il soit inférieur au nombre de lignes du fichier transmis à la construction de l'objet `SlideShow`.

Note: le code de `getRandomNumber` est fortement basé sur l'exemple fourni sur ce lien.

2.c. Expliquez dans les commentaires de la fonction membre `getRandomNumber` pourquoi on utilise le mot clé `static` pour les variables d'implémentation de cette fonction.

2.d. Expliquez dans les commentaires de `SlideShow.hpp` pourquoi le constructeur par défaut est mis ici en privé - `private`.

2.e. Complétez l'implémentation de la fonction membre `displayRandomSlide` de `SlideShow` pour que lors de son exécution, avec la sortie standard passée en argument, elle affiche aléatoirement le contenu d'une ligne du fichier transmis à la construction de l'objet `SlideShow`.

Note: l'objet de type ostream* (pointeur) passé en paramètre de la fonction permet d'utiliser soit la sortie standard dans le programme principal, soit un flux de chaîne de caractères - de type ostringstream - dans le programme de test, et tout ça avec la même fonction membre `displayRandomSlide`.

Modifiez le "Test Error Init" afin qu'il soit conforme à l'erreur qu'on souhaite valider.

2.f. Complétez les tests avec un nouveau `TEST_CASE` pour vérifier le bon fonctionnement de la fonction membre `displayRandomSlide` de `SlideShow` que vous avez implémentée.

3. Compilation et exécution du programme principal

3.a. Lancez l'exécution du programme principal avec '`make run`'. Que constatez-vous ? Complétez la fonction `main` afin de tenir compte du fichier passé en argument de l'exécutable **slide-show**.

3.b. Utilisez un objet `SlideShow` pour afficher aléatoirement 20 noms de fichier image - contenu de chaque ligne du fichier `data/selection.txt` - l'un à la suite des autres, sur la sortie standard.

3.c. Modifiez le code en utilisant la fonction sleep_until et affichez continuellement et aléatoirement un nom d'image toutes les 2s jusqu'à l'arrêt du programme par "CTRL+C".