Flots ; Entrées/Sorties.

Buts

Cette série a pour but de vous faire pratiquer les entrées/sorties : lecture et écriture de fichier essentiellement.

Exercice 1 : Écriture dans un fichier (ofstream, niveau 1)

> Exercice n°28 (pages 74 et 239) de l'ouvrage *C++ par la pratique*.

Écrivez le programme `ecriture.cc` qui :

- lise depuis le clavier les **noms** et **âges** de différents individus;
- sauvegarde ces données dans un fichier nommé `data.dat`.

Votre programme devra en outre :

- vérifier que l'ouverture du fichier s'est correctement réalisée, et afficher un message d'erreur dans le cas contraire ;
- lire des valeurs tant que l'utilisateur n'indique pas qu'il a terminé la saisie, en appuyant sur les touches `CTRL+D` (ce qui correspond au caractère signalant la fin de fichier, testée par `eof()`).
  (Note : sous Windows, tapez `CTRL+Z` puis `Enter`)

Indications:

- n'oubliez pas de vous servir le cas échéant des fiches résumé du cours ou des « mini-références » ;
- pensez à vérifier que vos opérations d'extraction (entrée) se déroulent bien en testant `fail()` comme vu en cours ;
- n'oubliez pas de fermer le fichier à la fin des opérations d'écriture;
- et finalement, vous pouvez regarder le contenu du fichier en tapant, depuis un terminal, la commande

  ```
  cat data.dat
  ```

  (il faut d'abord vous rendre dans le répertoire approprié !).

Testez votre programme de sorte à obtenir le résultat suivant :

```
Jo      24
Marc    35
Ted     74
Andy     3
Werner  48
OldBob 103
```

Exemple d'exécution :

```
Entrez un nom (CTRL+D pour terminer) : Jo
âge : 24
Entrez un nom (CTRL+D pour terminer) : Marc
âge : Ted
Je vous demande un age (nombre entier positif) pas n'importe quoi !
Cet enregistrement est annulé.
Entrez un nom (CTRL+D pour terminer) : Marc
âge : 35
Entrez un nom (CTRL+D pour terminer) :
```

Exercice 2 : lecture depuis un fichier (ifstream + manipulateurs, niveau 1)

> Exercice n°29 (pages 75 et 240) de l'ouvrage *C++ par la pratique*.

Dans le fichier `lecture.cc`:

1. Affichez à l'écran le contenu du fichier créé lors de l'exercice précédant, et affichez de plus le nombre de personnes contenues dans ce fichier, ainsi que la moyenne et les extrêmes des âges.
   Vérifiez que l'ouverture du fichier s'est bien réalisée, et affichez un message d'erreur dans le cas contraire.
2. Modifiez ensuite votre programme de sorte qu'il réalise l'affichage en s'approchant le plus possible du format suivant :
   - affichage du nom sur 15 caractères, alignement à gauche;
   - affichage de l'âge sur 3 caractères, alignement à droite ;
   - affichage du nombre total de personnes sur 2 caractères ;
   - affichage de la moyenne sur 3 caractères au maximum ;
   - affichage des âges minimum et maximum comme les autres âges.

Exemple :

```
+-----------------+-----+
| jo              |  24 |
| marc            |  35 |
| ted             |  74 |
| andy            |   3 |
| werner          |  48 |
| bob             | 103 |
+-----------------+-----+
  âge minimum     :   3
  âge maximum     : 103
 6 personnes, âge moyen : 47.8 ans
```

Exercice 3 : Statistiques sur un fichier (fichiers, tableaux, niveau 2)

> Exercice n°30 (pages 76 et 242) de l'ouvrage *C++ par la pratique*.

On cherche ici à écrire un programme `stats.cc` qui calcule les statistiques sur les lettres contenues dans un fichier.

Ouverture du fichier

Écrire une fonction `demander_fichier`, qui prend un `ifstream` comme argument par référence et retourne un booléen indiquant si le fichier a pu être ouvert :

```
bool demander_fichier(ifstream& fichier);
```

Cette fonction, après avoir demandé à l'utilisateur d'entrer le nom du fichier à lire, ouvrira le fichier correspondant.

En cas d'erreur à l'ouverture, la fonction redemandera le nom du fichier, au maximum 3 fois. Au bout de 3 échecs le programme abandonnera...

Exemple d'exécution :

```
Nom du fichier à lire : stupid.name
 -> ERREUR, je ne peux pas lire le fichier stupid.name
Nom du fichier à lire : stupid2.name
 -> ERREUR, je ne peux pas lire le fichier stupid2.name
Nom du fichier à lire : evenmorestupid.name
 -> ERREUR, je ne peux pas lire le fichier evenmorestupid.name
=> j'abandonne !
```

Exemple positif :

```
Nom du fichier à lire : stupid.name
 -> ERREUR, je ne peux pas lire le fichier stupid.name
Nom du fichier à lire : data.dat
 -> OK, fichier data.dat ouvert pour lecture.
```

Récolte des statistiques

Définir le type `Statistique` comme un tableau d'entiers longs non-signés (libre à vous de choisir un tableau statique ou un tableau dynamique).

Écrire une fonction `initialise_statistique`, prenant comme argument une variable `Statistique` (et peut être d'autres arguments si nécessaire) et initialisant à 0 tous ses éléments.

Écrire une fonction `collecte_statistique`, de prototype

```
unsigned long int collecte_statistique(Statistique& a_remplir, ifstream& fichier_a_lire)
```

qui collectera le nombre de fois que chaque caractère compris entre l'espace (`' '` ou `char(32)`) et `'}'` (`char(125)`) apparaît dans le fichier `fichier_a_lire`.

Ainsi `a_remplir[0]` contiendra le nombre d'espaces contenus dans le fichier `fichier_a_lire`, et `a_remplir[93]` le nombre de `'}'` que contient `fichier_a_lire`.

Pour lire un fichier caractère par caractère, utilisez la fonction `get(char)` comme indiqué dans la « mini-référence » sur les flots.

La fonction `collecte_statistique()` retournera le nombre total de caractères enregistrés dans `a_remplir`, c'est-à-dire la somme de ses éléments.

Affichage des statistiques

Écrire pour finir une fonction `affiche` qui prend une `Statistique` en argument (plus d'autres arguments si nécessaire) et affiche les statistiques récoltées en valeurs absolues (les vrais nombres) et relatives (pourcentages).

Les valeurs absolues seront affichées alignées à droite sur 11 caractères et les pourcentages alignés à droite sur 5 caractères.

Attention ! Il ne faut pas afficher les caractères qui ne sont pas apparus dans le fichier (c.-à-d. ayant un compte de 0).

Exemple d'affichage :

```
STATISTIQUES :
  :           6 -  12.2%
A :           1 -  2.04%
C :           1 -  2.04%
D :           1 -  2.04%
E :           3 -  6.12%
I :           3 -  6.12%
...
```

Exercice 4 : QCM revisités (fichiers, structures, niveau 2)

> Exercice n°31 (pages 77 et 244) de l'ouvrage *C++ par la pratique*.
> (pages 77 et 245 dans la 2e édition).

Cet exercice reprend l'exercice 20 de la semaine 6 du MOOC (semaine 9 d'ICC). On veut maintenant que ce programme `qcm` soit indépendant d'un QCM particulier (séparation de l'algorithme et des données).

Pour cela on va donner la possibilité à ce programme de lire les QCM depuis un fichier.

Recopier le programme `qcm.cc` de la fois passée dans votre répertoire courant (serie12/) et éditez-le.

Commencez par y recopier la fonction `demander_fichier` de l'exercice précédent.

Puis changez la fonction `creer_examen` de la fois passée (celle qui créait le QCM) de sorte qu'elle crée le contenu du QCM à partir d'un fichier passé en argument :

```
Examen creer_examen(ifstream& fichier);
```

Le format du fichier à lire est le suivant :

```
Q: question
reponse1
->reponse2
reponse3

Q: question2
...
```

Le signe `->` en début de ligne indique la bonne réponse.
Le signe `#` en début de ligne indique une ligne de commentaire.

Pour indiquer une réponse commençant par "`->`" ou par "`#`", il suffit de ne pas mettre ce signe juste au début de ligne mais de mettre un espace. La procédure de lecture devra éliminer ces espaces initiaux.

Note : l'avantage de ce format est que l'ordre des réponses peut facilement être changé dans le fichier de configuration sans avoir à modifier l'indication de la bonne réponse.

Exemple de fichier :

```
Q: Combien de dents possède un éléphant adulte
32
-> de 6 à 10
beaucoup
24
2

Q: Quel signe est le plus étrange
 #
 ->
->->##<-
#b
a
```

La dernière question correspond à

```
Quel signe est le plus étrange ?
    1- #
    2- ->
    3- ->##<-
    4- a
```

et la réponse est `3`.

Créez, dans `exam1.txt` le fichier correspondant au questionnaire de la fois passée et testez votre programme.

Dernière mise à jour le 4 décembre 2024
Last modified: Wed Dec 4, 2024