Bases du Dev. Logiciel

Exercice 1

Soit deux tableaux de nombres entiers V1 et V2 contenant chacun N éléments.

1. Donnez un algorithme permettant de créer un tableau W contenant 2*N éléments tels que les éléments d'indice paire de W contiennent les éléments de V1 et les éléments d'indices impairs de W contiennent les éléments de V2.
2. Donnez le code C++ de la fonction implémentant cet algorithme.
3. Listez les erreurs possibles qu'un utilisateur pourrait commettre en utilisant votre fonction. Amendez votre code pour détecter les cas d'erreurs possibles et émettre un message d'erreur le cas échéant.

Exercice 2

Soit un tableau de nombres réels V contenant chacun N éléments.

1. Donnez un algorithme permettant de calculer la moyenne des éléments de V de manière itérative.
2. Donnez un algorithme permettant de calculer la moyenne des éléments de V de manière récursive.
3. Donnez le code C++ des fonctions implémentant ces deux algorithmes.
4. En utilisant la fonction `std::chrono`, mesurez le temps d'exécutions des deux algorithmes pour des vecteurs de tailles croissantes de 1000 à 10000000 éléments. Que constatez vous ?

Exercice 3

Un format d'image simple consiste à stocker dans un tableau à deux dimensions la suite des valeurs de chaque pixels. Chaque pixel représente un niveau de gris encodé par un entier valant 0(noir) à 255 (blanc).

- Définissez un type de donnée abstrait `Image` permettant de stocker l'ensemble des informations nécessaires à la récupération des données images issu d'un fichier PPM.
- Implémenter ce type en C++ en utilisant le type `vector` pour le stockage des données. Vous trouverez ici les informations nécessaires à la définition d'un type abstrait en C ++ :

  https://en.wikipedia.org/wiki/C%2B%2B_classes#Aggregate_classes

  Restez simple, l'utilisation de struct suffit pour ces exercices.

- Donner un algorithme et une implantation d'une fonction `afficheImage` qui affiche à l'écran une image en niveau de gris stockées dans une `Image` en utilisant des caractères différents pour représenter les différents niveaux de gris :
  - Un espace pour les valeurs de pixel entre 0 et 15
  - Un point (.) pour les valeurs de pixels entre 16 et 31
  - Un ° pour les valeurs de pixels entre 32 et 47

- Un * pour les valeurs comprises entre 48 et 63
- Un X pour les valeurs entre 63 et 95
- Un # au delà

Veuillez à respecter la hauteur et la largeur de vos images lors de l'affichage.

A partir du type de données défini ci-dessus, nous allons définir quelques manipulations d'images :

- Donnez l'algorithme et implémentez la fonction `InverseImage` qui, à partir d'une image, inverse la valeur de chacun de ces pixels (0 devient 255, 1 devient 254, etc…)

- Donnez l'algorithme et implémentez la fonction `Transpose` qui, à partir d'une image de hauteur H et de largeur L, effectue une rotation à 90° de cette image, créant ainsi une image de hauteur L et de largeur H contenant les mêmes pixels.

Le floutage est une opération classique en traitement d'image. Elle consiste, pour chaque pixel d'une image d'entrée, de calculer la moyenne des 8 pixels voisins autour de ce pixel et d'en stocké le résultat dans une image de sortie. Ainsi pour chaque pixel P(i,j), on calcule le pixel résultat R(i,j) comme :

```
R(i,j) = (   P(i-1,j-1) + P(i,j-1)+ P(i+1,j-1)
           + P(i-1,j) + P(i,j) + P(i+1,j)
           + P(i-1,j+1) + P(i,j+1) + P(i+1,j+1)
         ) / 9
```

- En ignorant les bords de l'image, donnez l'algorithmes effectuant ce calcul
- Implémentez cet algorithme et testez-le sur une image adéquate permettant de vérifier vos résultats.
- Quelles modifications seraient nécessaire afin de traiter les bords ?