INF1005C - PROGRAMMATION PROCÉDURALE
Travail dirigé No. 4
Les fonctions

Objectifs : Permettre à l'étudiant d'explorer la notion de fonctions.

Durée : Deux séances de laboratoire.

Remise du travail : Dimanche 26 octobre 2025, avant 23h30.

Travail préparatoire : Lecture des exercices et rédaction des algorithmes.

Documents à remettre: Sur le site Moodle des travaux pratiques, on peut remettre l'ensemble des fichiers .cpp compressés des exercices dans un fichier .zip en suivant **la procédure de remise des TDs.**
Retard : Les retards ne sont pas tolérés, un retard méritera la note 0.

Directives particulières

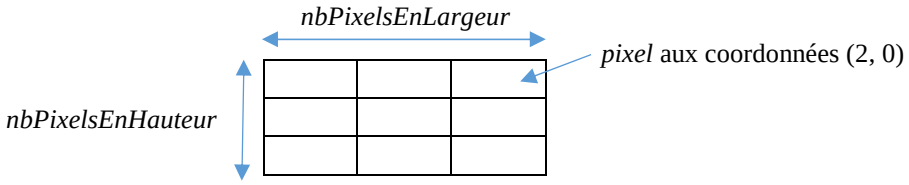
- Lisez le document *Boucle sur intervalle, cppitertools et span* sur le site Moodle du cours.
- Vous pouvez ajouter d'autres fonctions à celles décrites dans l'énoncé, ainsi que d'autres structures (et sous-structures), pour améliorer la lisibilité et suivre le principe DRY (Don't Repeat Yourself). À chaque endroit où vous remarquez une duplication de code (vous avez écrit les mêmes opérations plus d'une fois) et qu'il n'est pas possible de l'éliminer avec ce qui a été vu en cours, indiquez-le en commentaire.
- Pour votre information, le corps d'aucune des fonctions (incluant le *main*) du solutionnaire ne fait plus que 10 lignes, excluant les commentaires, lignes vides ou lignes ne contenant qu'une accolade, et excluant la fonction *autresTests* (ou lignes ajoutées au *main*, en plus de ce qui est demandé dans l'énoncé, pour vérifier que les fonctions donnent le bon résultat); la moyenne est autour de 5 lignes.
- Il est interdit d'afficher directement ou indirectement dans les fonctions décrites si la description n'indique pas d'affichage. Notez que les verbes retourner et indiquer ne disent pas qu'un affichage est permis. Les termes afficher, voir à l'écran, présenter à l'utilisateur, sont des indications qu'il faut afficher.
- Respecter le guide de codage, les points pertinents pour ce travail sont donnés en annexe à la fin.
- N'oubliez pas de mettre les entêtes de fichiers (guide point 33), et pour chaque fonction (guide point 89).
- Ne pas oublier de donner la description IN/OUT pour chacun des paramètres.
- Compiler avec /W4. Lors de la compilation avec Visual Studio 2022 (ou 2019), votre programme ne devrait pas afficher d'avertissements (« warnings » de « build »).

Gestion des Images

La classification d'images est une pratique assez fréquente dans le domaine de l'imagerie. Elle consiste à regrouper des images suivant un ou plusieurs critères donnés. A titre d'exemple, dans le domaine de l'imagerie médicale la classification automatique des scans des patients est très intéressante pour détecter les patients atteints d'une maladie donnée.

En tant que développeur stagiaire dans une compagnie de développement de logiciel d'imagerie, on vous demande de développer une partie d'un système de classification d'images.

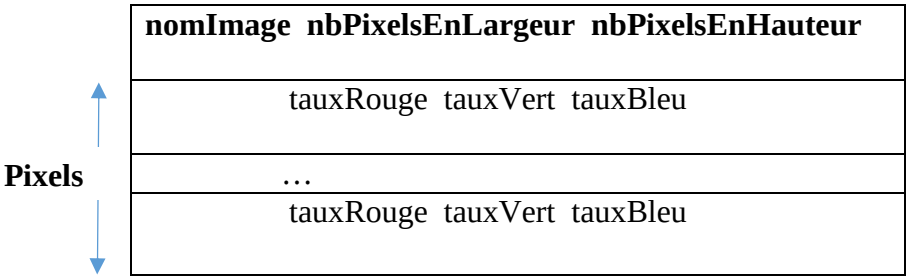
Dans le cadre de ce TD, chaque image est constituée d'un ensemble de pixels. Le terme « pixel » provient de la contraction de « picture element », qui constitue un point de l'image. Pour décrire la couleur d'un pixel, on utilise le modèle RVB (R : rouge, V : vert et B : bleu; en anglais dit RGB) dans lequel la couleur de chaque pixel est décrite par la concentration de ces trois couleurs primaires. A titre d'exemple, la couleur noire est donnée par les concentrations suivantes (R=0, V=0 et B=0). Une image peut être représentée sous forme d'un tableau à deux dimensions de pixels; voir la figure suivante.



Travail demandée :

Suivez les directives indiquées dans le fichier *TD4.cpp* pour implémenter les fonctions ainsi que le *main*. Chaque fonction à compléter contient un ou plusieurs « //TODO: » (voir la section « Méthodologie de correction d'erreurs et de programmation » dans le document « Programmer avec Visual Studio » sur le site Moodle). Le *main* est principalement des appels aux autres fonctions à écrire; on propose d'écrire les fonctions qui correspondent en même temps que les parties du *main*, pour tester vos fonctions à mesure que vous les écrivez.

Lisez-bien toutes les définitions de constantes et structures au début du programme pour les utiliser correctement. Aussi, vérifiez-bien que vous utilisez toutes les fonctions écrites. On vous fournit également le fichier *Images.txt* qui contient l'ensemble d'images d'un groupe d'image. Chaque image dans le fichier est structurée de la manière suivante :



Les pixels sont disposées dans le fichier *Image.txt* suivant l'ordre des lignes (on lit les pixels de la première ligne de l'image, de gauche à droite, puis de la deuxième ligne jusqu'à la dernière ligne de l'image).

Pour certaines fonctions c'est indiqué « Cette fonction ne doit avoir aucun ancien "for". », il faut alors utiliser seulement des boucles sur intervalles (aucun ancien « for ») et minimiser l'utilisation de « range » (préférer les « span » de C++20). Si ça vous aide, vous pouvez évidemment commencer par utiliser les anciens « for » avant de les transformer en nouveaux.

Notez qu'il est possible de faire le programme entier sans ancien « for » mais ce n'est pas demandé.

Le résultat de l’exécution du TD est le suivant :

```
**************************************************
  Type du groupe d'images: Images de tests
**************************************************
==================================================
 Nom de l'image: Image_Rouge
==================================================
   RRQRRR
   RBRRRR
   RRRRRR
   QR RRR
==================================================
==================================================
 Nom de l'image: Image_Verte
==================================================
   VVVV
    VQV
   VQVV
   BBBB
   BBBB
   BBBB
==================================================
==================================================
 Nom de l'image: Mario_http://pixelartmaker.com/art/7dd56f5049ccd8c
==================================================
   QQQQQQQQQQQQQQQQQ
   QQQQQQQQ    Q   QQ
   QQQQQQ  RRR  QQQ Q
   QQQQQ RRRRRR QQQ Q
   QQQQ RRR      QQ Q
   QQQ RRR        Q Q
   QQQ R  QQQQQQ   QQ
   QQ    QQQ Q Q RR Q
   QQ Q  QQQ Q Q RR Q
   Q QQ   QQQQQQQ R Q
   Q QQQ QQ QQQQQ R Q
   QQ QQQQ    QQ   QQ
   QQQ  QQQQ     R QQ
   QQQQ   QQQQQ RR QQ
   QQQ RRR      R QQQ
   QQ RRRRR  RR  QQQQ
   Q RRR   B RRR  QQQ
   Q RR QQQ B RRR QQQ
   Q R QQQQQ B     QQ
   QQ  QQQQQ QQBB  QQ
   QQQ  QQQ BQQB VV Q
   QQ VV   BBBB VVV Q
   Q  V BBBBBBB VVV Q
   Q VV BBBB    VVV Q
   Q VV B   QQQ VV  Q
   Q VV  QQQQQQQ   QQ
   QQ  QQQQQQQQQQQQQ
   QQQQQQQQQQQQQQQQQ
```

****

Annexe 1 : Points du guide de codage à respecter

Les points du **guide de codage** à respecter **impérativement** pour ce TD sont les suivants :
(voir le guide de codage sur le site Moodle du cours pour la description détaillée de chacun de ces points)

Nouveaux points depuis le TD3 :

2 : noms des types en UpperCamelCase
5 : noms des fonctions en lowerCamelCase
48 : aucune variable globale (les constantes globales sont tout à fait permises)
50 : mettre le & près du type
89 : entêtes de fonctions; y indiquer clairement les paramètres [out] et [in,out]

Points du TD3 :

3 : noms des variables en lowerCamelCase
21 : pluriel pour les tableaux (int nombres[];)
22 : préfixe n pour désigner un nombre d'objets (int nElements;)
24 : variables d'itération i, j, k mais jamais l
27 : éviter les abréviations (les acronymes communs doivent être gardés en acronymes)
29 : éviter la négation dans les noms
33 : entête de fichier
42 : include au début
46 : initialiser à la déclaration
47 : pas plus d'une signification par variable
51 : test de 0 explicite (if (nombre != 0))
52, 14 : variables vivantes le moins longtemps possible
53-54 : boucles for et while
58-61 : instructions conditionnelles
62 : pas de nombres magiques dans le code
67-78, 88 : indentation du code et commentaires
83-84 : aligner les variables lors des déclarations ainsi que les énoncés
85 : mieux écrire du code incompréhensible plutôt qu'y ajouter des commentaires