INF1010
Programmation orientée-objet

Travail pratique 4
Classes et Fonctions Génériques

Département de génie informatique et logiciel

Polytechnique Montréal
09 mars 2020

Objectifs

Permettre à l'étudiant de se familiariser avec les classes et fonctions génériques tout en pratiquant un peu plus les notions de cours couvertes depuis le début de la session.

Remise du travail

- Date : **17 mars 2020 à 23h55**
- **Une note de 0 sera attribuée aux équipes qui remettent leur travail en retard.**
- Remettre uniquement **tous** les fichiers .cpp et .h sous une archive **.zip**
- Nom du fichier zip : **matricule1_matricule2_groupe.zip** où matricule1 < matricule2. Exemple : 1234566_1234567_1.zip

Références

- Notes de cours sur Moodle
- Livre Big C++ deuxième édition
- https://en.cppreference.com/w/

Directives

- Les travaux s'effectuent obligatoirement en équipe de deux personnes faisant partie du même groupe.
- Les entêtes de fichiers sont obligatoires.
- Les fonctions doivent être documentées.
- **Le guide de codage sur Moodle doit être suivi.**

Conseils

- Lisez tout le document avant de commencer!
- Ayez lu vos notes de cours!
- Profitez de la présence des chargés de lab. pour répondre à la plupart de vos questions !

Faites attention si vous partagez votre travail, car le plagiat sera pénalisé par **une note de 0**.

Venez poser vos questions sur Slack! Vous devez utiliser votre adresse @polymtl.ca.
Lien pour faire son compte : https://join.slack.com/t/inf1010-h20/signup
Lien du workspace : https://inf1010-h20.slack.com

Informations préalables

Le présent TP constitue une suite du programme **`CinéPoly`** réalisé lors des précédents TPs, il aura pour objectif d'utiliser les notions apprises en cours sur les fonctions et classes génériques afin de contribuer à l'amélioration des fonctionnalités du logiciel.

L'équipe multimédia de l'entreprise **`CinéPoly`** a maintenant besoin de votre aide afin d'implémenter quelques algorithmes de traitement d'images. Elle en a besoin pour l'interface graphique de son application. Pour implémenter ces algorithmes, elle vous demande d'utiliser des classes génériques pour être le plus flexible possible par rapport aux images sur lesquelles on les utilise. Donc, vous décidez de simuler votre propre classe Image tout en essayant de la garder la plus indépendante possible du fonctionnement des algorithmes.

Dans le monde du multimédia, une image contient des pixels. Elle est donc représentée par une matrice de pixels. Cependant, vu que l'entreprise **`CinéPoly`** vient juste de se lancer, son équipe de développement n'a toujours pas défini la façon dont elle va implémenter les pixels de l'image. On veut donc que notre classe Matrice soit flexible pour accepter différents types d'éléments avec certains critères.

Dans ce TP, on va tout d'abord commencer par l'implémentation d'une classe simple Pixel, qui correspond à des intensités de rouge, vert et bleu, et qui définit les éléments de notre matrice de pixels. Par la suite, on implémentera une classe Matrice qui sera en relation d'agrégation avec la classe Image. Enfin, on passera à l'implémentation de la classe **AgrandirMatrice** qui permet de redimensionner une Image avec une interpolation par plus proche voisin, et de la classe **PivoterMatrice** qui a pour but de faire pivoter une Image dans une direction précise.

Pour vous aider, on va bien vous détailler le fonctionnement de chacun des algorithmes et comment les implémenter. On vous fournit également quelques méthodes et classes pour que vous vous inspiriez. Assurez-vous de bien lire l'énoncé et de faire de votre mieux pour assister la séance du laboratoire.

ATTENTION :

- **Tout au long du TP, n'utilisez que les pointeurs intelligents.**
- **Vous serez pénalisés pour les utilisations inutiles du mot-clé *this*. Vous n'allez pas avoir besoin de l'utiliser pour ce TP.**
- **Il est fortement conseillé de suivre l'ordre d'implémentation proposé dans la section ci-dessous.**
- **Veuillez ne pas modifier les fichier « main.cpp », « Image.h », « def.h » et « debogageMemoire.h » !**
- **Les tests fournis sont là pour vous guider ! La note des tests n'est pas nécessairement votre note finale du TP.**

Remarque : Faites-vous plaisir de nous envoyer vos commentaires ! On apprécierait tout feedback que vous pouvez avoir par rapport au TP !

Travail à réaliser

On vous demande de compléter les fichiers qui vous sont fournis pour pouvoir implémenter le système décrit ci-dessous.

`Classe Pixel`

Cette classe représente un pixel.

Cette classe contient les attributs suivants :

- `rouge_ (uint8_t)`
- `vert_ (uint8_t)`
- `bleu_ (uint8_t)`

Les méthodes suivantes doivent être implémentées :

- `operator=(const Pixel& pixel)`
  - L'opérateur doit copier les éléments du paramètre `pixel` dans l'objet.
- `operator<< (std::ostream& os, Pixel pixel)`
  - L'opérateur permet d'afficher tous les attributs de l'épisode.
  - L'opérateur doit pouvoir être appelé en cascade.
  - L'affichage des attributs doit se faire en hexadécimal tel que montré dans la dernière section de l'énoncé.
  - L'opérateur ne doit pas être déclaré `friend`.
  - Pensez à utiliser les fonctions `std::setw`, `std::setfill` et `std::uppercase`.
  - `Operator<< Pixel(10, 16, 15)` doit afficher : **#0A 10 0F**
- `operator>> (std::istream& is, Pixel& pixel)`
  - L'opérateur permet d'initialiser tous les attributs de l'épisode.
  - L'opérateur doit pouvoir être appelé en cascade.
  - L'opérateur ne doit pas être déclaré `friend`.
  - Utiliser les Setters de la classe.
- Les setters
  - Rappelez-vous que les valeurs que les attributs peuvent prendre sont entre 0 et 255, car codés sur 8 bits.
  - L'appel à la méthode `setRouge(-155)` doit affecter une valeur de 0 à l'attribut `rouge_`.
  - L'appel à la méthode `setBleu(1998)` doit affecter une valeur de 255 à l'attribut `bleu_`.
  - **Vous aurez un Point de Bonus si vous n'utilisez aucune instruction « if ».**

`Classe Matrice<T>`

Cette classe représente une Matrice. C'est une classe générique qui contient des éléments de type T.

Cette classe contient les attributs suivants :

- `elements_ (std::vector<std::vector<T>>`
- `height_ (size_t)`
- `width_ (size_t)`

Les méthodes suivantes doivent être implémentées :

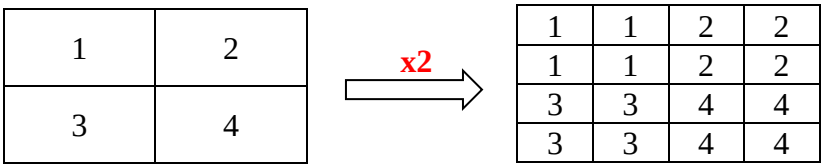
- Le constructeur par défaut
  - Faites attention à bien initialiser l'attribut `elements_` en utilisant la constante `CAPACITE_MATRICE`
  - Initialisez les attributs `height_` et `width_` à `0`.

- `operator()(const size_t& posY, const size_t& posX) const`
  - Retourne l'élément se trouvant à la ligne `posY` et la colonne `posX` du tableau `elements_`.
  - Si `posY` est supérieure à `height_` ou `posX` est supérieure à `width_`, retourne un élément par défaut (`T()`).
- `ajouterElement(T element, const size_t& posY, const size_t& posX)`
  - Ajoute l'élément passé en paramètre dans la position correspondante.
  - Retourne `true` si l'ajout a été effectué.
- `lireElement(const std::string& elementFichier, const size_t& posY, const size_t& posX)`
  - Lit l'élément à partir de la chaine de caractères `elementFichier` puis l'ajoute dans la matrice.
  - Si la lecture a été bien faite, utilisez la méthode `ajouterElement` pour ajouter l'élément lu dans la matrice.
  - Retourne `false` si la lecture a échoué ou si l'ajout de l'élément a échoué.
  - Utilise l'opérateur >> pour lire l'élément.
- `chargerDepuisFichier(const std::string& nomFichier)`
  - Charge la matrice à partir du fichier passé en paramètre.
  - Utilise la méthode `lireElement`.
  - Voir la façon dont les fichiers `txt` fournis sont structurés pour mieux comprendre comment faire la lecture du fichier.
- `clone()`
  - Retourne un pointeur vers une copie de la matrice.
- Les setters
  - L'appel à la méthode `setHeight(CAPACITE_MATRICE+10)` affecte une valeur de `CAPACITE_MATRICE` à `height_`.
  - **N'utilisez aucune instruction « `if` ».**

`Classe AgrandirMatrice<M>`

Cette classe est une classe générique qui implémente l'agrandissement d'une matrice avec interpolation au plus proche voisin. Lisez attentivement le paragraphe suivant pour bien comprendre cet algorithme.

Lorsqu'on agrandit une image par un facteur d'échelle donné (dans l'exemple ci-dessous le facteur est égal à 2), de nouveaux pixels sont créés. Il s'agit donc de trouver les bonnes valeurs à placer dans ces pixels. C'est qu'on appelle une interpolation. La technique la plus simple pour l'interpolation est par plus proche voisin. Pour un pixel à la position (i,j) dans la nouvelle matrice, on lui attribue la valeur du pixel le plus proche dans l'image originale, soit celui à la position (i/facteur, j/facteur).



Cette classe contient les attributs suivants :

- `matrice_ (M*)`

Les méthodes suivantes doivent être implémentées :

- Le constructeur par défaut
- `trouverLePlusProcheVoisin(const unsigned int& rapport, size_t posY, size_t posX) const`
  - Retourne les coordonnées du point le plus proche. Par exemple : `trouverLePlusProcheVoisin(3, 1, 5)` retourne la valeur `{0, 1}` et `trouverLePlusProcheVoisin(2, 1, 5)` retourne la valeur `{0, 2}`.
- `redimensionnerImage(const unsigned int& rapport)`
  - Utilise la méthode `clone` de la classe Matrice pour faire une copie de la matrice.
  - Met à jour la largeur et la hauteur de la matrice à redimensionner.
  - Utilise la méthode `trouverLePlusProcheVoisin` pour remplir tous les éléments de la nouvelle matrice.

`Classe PivoterMatrice<M>`

Cette classe est une classe générique qui implémente l'algorithme de rotation d'une matrice. Lisez attentivement le paragraphe suivant pour bien comprendre cet algorithme. ***Pour simplifier le TP, on suppose que toutes les matrices qu'on passe à cet algorithme sont des matrices carrées d'une taille impaire.***

Pour pouvoir faire une rotation d'une matrice d'un angle θ, on utilise la matrice de rotation pour appliquer une transformation sur tous les points de cette matrice. Pour trouver les coordonnées d'un point donné après la transformation, on fait la multiplication matricielle suivante :

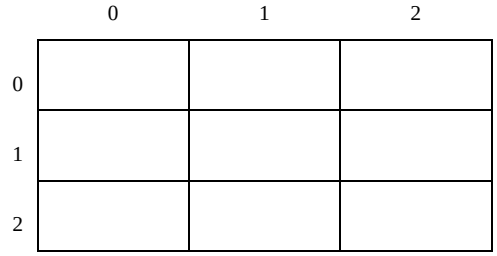
$$\begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x.cos\theta - y.sin\theta \\ x.sin\theta + y.cos\theta \end{pmatrix}$$

Pour ce TP, on va faire des rotations de 90 degrés dans les deux directions. Donc, si on remplace dans notre équation l'angle par 90 degrés et -90 degrés, on obtient les deux résultats suivants :

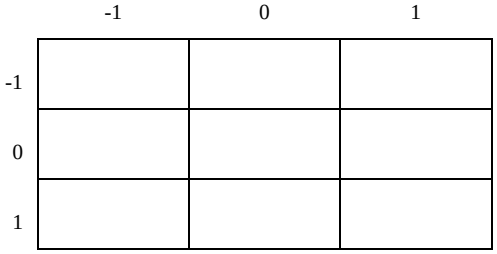
$$\begin{pmatrix} cos90 & -sin90 \\ sin90 & cos90 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -y \\ x \end{pmatrix}$$

$$\begin{pmatrix} \cos(-90) & -\sin(-90) \\ \sin(-90) & \cos(-90) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} y \\ -x \end{pmatrix}$$

On veut utiliser les résultats qu'on vient de trouver pour implémenter notre algorithme ! Cependant, il faut faire un changement d'axes pour avoir comme centre de rotation le centre de la matrice. Par exemple :



On veut transformer le système de coordonnées ci-haut pour avoir le système suivant :



Cette classe contient les attributs suivants :

- `matrice_ (M*)`

Les méthodes suivantes doivent être implémentées :

- Le constructeur par défaut
- `changerCoordonneesCentreMatrice(Coordonnees coords)`
  - Retourne les nouvelles coordonnées d'un point passé en paramètre tel qu'expliqué dans la section ci-dessus. Par exemple : l'appel à la méthode `changerCoordonneesCentreMatrice({1, 2})` dans une matrice de taille 5 doit retourner `{-1, 0}` et dans une matrice de taille 9 les coordonnées `{-3, -2}`.
- `recupererCoordonnees(Coordonnees coords)`
  - Récupére les anciennes coordonnées d'un point passé en paramètre. Par exemple : l'appel à la méthode `recupererCoordonnees({1, 2})` dans une matrice de taille 5 doit retourner `{3, 4}` et dans une matrice de taille 9 les coordonnées `{5, 6}`.
- `pivoterMatrice(Direction direction)`
  - Utilise la méthode `clone` de la classe Matrice pour faire une copie la matrice.
  - Utilise la méthode `changerCoordonneesCentreMatrice` pour trouver les coordonnées de chaque point par rapport au centre de la matrice.
  - Utilise une des équations expliquées ci-dessus pour trouver les nouvelles coordonnées par rapport au centre de la nouvelle matrice.
  - Utilise la méthode `recupererCoordonnees` pour revenir au système précédent.

main.cpp

Le fichier main.cpp vous est fourni et ne devrait pas avoir à être modifié pour la remise. Une série de tests sont fournis dans ce fichier pour vous aider à vérifier le comportement de votre programme. Vous pouvez désactiver des blocs de tests en changeant les #ifdef true pour des #ifdef false afin de compiler le programme avant d'avoir tout terminé. Si un test échoue, allez voir le but du test dans la fonction main.

Fichiers de lecture

Trois fichiers de lecture vous sont fournis. `Matrice_nombres.txt`, `matrice_pixel.txt` et `matrice_text.txt`. Ces fichiers sont utilisés pour tester votre programme et vos méthodes de la classe Matrice. Assurez-vous de mettre les fichiers à un endroit où votre programme peut les lire. Chaque ligne dans les fichiers est équivalente à une colonne dans la matrice et les lignes de la matrice sont séparés par le caractère « L ».

Sortie attendue

```
################################## Affichage de la matrice des entiers ##################################

99 | 15 | 20 | 33 | 40 | 57 | 61 |
62 | 70 | 84 | 90 | 99 | 16 | 27 |
21 | 34 | 49 | 52 | 64 | 76 | 81 |
89 | 94 | 99 | 13 | 27 | 38 | 18 |
47 | 53 | 64 | 75 | 86 | 97 | 11 |
99 | 14 | 23 | 32 | 41 | 50 | 69 |
12 | 24 | 36 | 48 | 50 | 62 | 74 |

############# Affichage de la matrice des entiers apres le pivotement vers la gauche #############

61 | 27 | 81 | 18 | 11 | 69 | 74 |
57 | 16 | 76 | 38 | 97 | 50 | 62 |
40 | 99 | 64 | 27 | 86 | 41 | 50 |
33 | 90 | 52 | 13 | 75 | 32 | 48 |
20 | 84 | 49 | 99 | 64 | 23 | 36 |
15 | 70 | 34 | 94 | 53 | 14 | 24 |
99 | 62 | 21 | 89 | 47 | 99 | 12 |

################ Affichage de la matrice des entiers apres le redimmensionnement ################

61 | 61 | 61 | 27 | 27 | 27 | 81 | 81 | 81 | 18 | 18 | 18 | 11 | 11 | 11 | 69 | 69 | 69 | 74 | 74 | 74 |
61 | 61 | 61 | 27 | 27 | 27 | 81 | 81 | 81 | 18 | 18 | 18 | 11 | 11 | 11 | 69 | 69 | 69 | 74 | 74 | 74 |
61 | 61 | 61 | 27 | 27 | 27 | 81 | 81 | 81 | 18 | 18 | 18 | 11 | 11 | 11 | 69 | 69 | 69 | 74 | 74 | 74 |
57 | 57 | 57 | 16 | 16 | 16 | 76 | 76 | 76 | 38 | 38 | 38 | 97 | 97 | 97 | 50 | 50 | 50 | 62 | 62 | 62 |
57 | 57 | 57 | 16 | 16 | 16 | 76 | 76 | 76 | 38 | 38 | 38 | 97 | 97 | 97 | 50 | 50 | 50 | 62 | 62 | 62 |
57 | 57 | 57 | 16 | 16 | 16 | 76 | 76 | 76 | 38 | 38 | 38 | 97 | 97 | 97 | 50 | 50 | 50 | 62 | 62 | 62 |
40 | 40 | 40 | 99 | 99 | 99 | 64 | 64 | 64 | 27 | 27 | 27 | 86 | 86 | 86 | 41 | 41 | 41 | 50 | 50 | 50 |
40 | 40 | 40 | 99 | 99 | 99 | 64 | 64 | 64 | 27 | 27 | 27 | 86 | 86 | 86 | 41 | 41 | 41 | 50 | 50 | 50 |
40 | 40 | 40 | 99 | 99 | 99 | 64 | 64 | 64 | 27 | 27 | 27 | 86 | 86 | 86 | 41 | 41 | 41 | 50 | 50 | 50 |
33 | 33 | 33 | 90 | 90 | 90 | 52 | 52 | 52 | 13 | 13 | 13 | 75 | 75 | 75 | 32 | 32 | 32 | 48 | 48 | 48 |
33 | 33 | 33 | 90 | 90 | 90 | 52 | 52 | 52 | 13 | 13 | 13 | 75 | 75 | 75 | 32 | 32 | 32 | 48 | 48 | 48 |
33 | 33 | 33 | 90 | 90 | 90 | 52 | 52 | 52 | 13 | 13 | 13 | 75 | 75 | 75 | 32 | 32 | 32 | 48 | 48 | 48 |
20 | 20 | 20 | 84 | 84 | 84 | 49 | 49 | 49 | 99 | 99 | 99 | 64 | 64 | 64 | 23 | 23 | 23 | 36 | 36 | 36 |
20 | 20 | 20 | 84 | 84 | 84 | 49 | 49 | 49 | 99 | 99 | 99 | 64 | 64 | 64 | 23 | 23 | 23 | 36 | 36 | 36 |
20 | 20 | 20 | 84 | 84 | 84 | 49 | 49 | 49 | 99 | 99 | 99 | 64 | 64 | 64 | 23 | 23 | 23 | 36 | 36 | 36 |
15 | 15 | 15 | 70 | 70 | 70 | 34 | 34 | 34 | 94 | 94 | 94 | 53 | 53 | 53 | 14 | 14 | 14 | 24 | 24 | 24 |
15 | 15 | 15 | 70 | 70 | 70 | 34 | 34 | 34 | 94 | 94 | 94 | 53 | 53 | 53 | 14 | 14 | 14 | 24 | 24 | 24 |
15 | 15 | 15 | 70 | 70 | 70 | 34 | 34 | 34 | 94 | 94 | 94 | 53 | 53 | 53 | 14 | 14 | 14 | 24 | 24 | 24 |
99 | 99 | 99 | 62 | 62 | 62 | 21 | 21 | 21 | 89 | 89 | 89 | 47 | 47 | 47 | 99 | 99 | 99 | 12 | 12 | 12 |
99 | 99 | 99 | 62 | 62 | 62 | 21 | 21 | 21 | 89 | 89 | 89 | 47 | 47 | 47 | 99 | 99 | 99 | 12 | 12 | 12 |
99 | 99 | 99 | 62 | 62 | 62 | 21 | 21 | 21 | 89 | 89 | 89 | 47 | 47 | 47 | 99 | 99 | 99 | 12 | 12 | 12 |
```

```
################################## Affichage de la matrice des pixels ##################################

#66 FF 7F | #00 C8 00 | #7B 7B 7B | #93 C7 59 | #00 4B F8 |
#FF FF FF | #FD FE FC | #D2 FF DC | #7D 80 83 | #00 1C 31 |
#9E B9 14 | #17 EA 18 | #15 0F 13 | #00 35 E7 | #AF FF 0F |
#F1 FD D6 | #7C 7B 64 | #7D 80 83 | #00 1C 31 | #00 38 39 |
#78 82 8C | #96 A0 AA | #B4 BE C8 | #D2 DC E6 | #F0 FA FF |

############# Affichage de la matrice des pixels apres le pivotement vers la droite #############

#78 82 8C | #F1 FD D6 | #9E B9 14 | #FF FF FF | #66 FF 7F |
#96 A0 AA | #7C 7B 64 | #17 EA 18 | #FD FE FC | #00 C8 00 |
#B4 BE C8 | #7D 80 83 | #15 0F 13 | #D2 FF DC | #7B 7B 7B |
#D2 DC E6 | #00 1C 31 | #00 35 E7 | #7D 80 83 | #93 C7 59 |
#F0 FA FF | #00 38 39 | #AF FF 0F | #00 1C 31 | #00 4B F8 |

################ Affichage de la matrice des pixels apres le redimmensionnement ################

#78 82 8C | #78 82 8C | #F1 FD D6 | #F1 FD D6 | #9E B9 14 | #9E B9 14 | #FF FF FF | #FF FF FF | #66 FF 7F | #66 FF 7F |
#78 82 8C | #78 82 8C | #F1 FD D6 | #F1 FD D6 | #9E B9 14 | #9E B9 14 | #FF FF FF | #FF FF FF | #66 FF 7F | #66 FF 7F |
#96 A0 AA | #96 A0 AA | #7C 7B 64 | #7C 7B 64 | #17 EA 18 | #17 EA 18 | #FD FE FC | #FD FE FC | #00 C8 00 | #00 C8 00 |
#96 A0 AA | #96 A0 AA | #7C 7B 64 | #7C 7B 64 | #17 EA 18 | #17 EA 18 | #FD FE FC | #FD FE FC | #00 C8 00 | #00 C8 00 |
#B4 BE C8 | #B4 BE C8 | #7D 80 83 | #7D 80 83 | #15 0F 13 | #15 0F 13 | #D2 FF DC | #D2 FF DC | #7B 7B 7B | #7B 7B 7B |
#B4 BE C8 | #B4 BE C8 | #7D 80 83 | #7D 80 83 | #15 0F 13 | #15 0F 13 | #D2 FF DC | #D2 FF DC | #7B 7B 7B | #7B 7B 7B |
#D2 DC E6 | #D2 DC E6 | #00 1C 31 | #00 1C 31 | #00 35 E7 | #00 35 E7 | #7D 80 83 | #7D 80 83 | #93 C7 59 | #93 C7 59 |
#D2 DC E6 | #D2 DC E6 | #00 1C 31 | #00 1C 31 | #00 35 E7 | #00 35 E7 | #7D 80 83 | #7D 80 83 | #93 C7 59 | #93 C7 59 |
#F0 FA FF | #F0 FA FF | #00 38 39 | #00 38 39 | #AF FF 0F | #AF FF 0F | #00 1C 31 | #00 1C 31 | #00 4B F8 | #00 4B F8 |
#F0 FA FF | #F0 FA FF | #00 38 39 | #00 38 39 | #AF FF 0F | #AF FF 0F | #00 1C 31 | #00 1C 31 | #00 4B F8 | #00 4B F8 |

################## Affichage de la matrice des couleurs en chaine de caracteres ##################

   Abricot |     Acajou |    Albôtre |   Amarante |      Ambre |       Noir |
 AmÜthyste |    Ardoise |     Argent |     Argile |    Asperge |  Aubergine |
    Auburn |     Aurore |       Azur |     Azurin |     Basanú |      Beige |
 Beigeasse |    Caramel |    Carotte |     Cassis |    Cúladon |     Cerise |
    Cúrulú |      Chair |    Chamois |  Champagne |  Chôtaigne |    Chôtain |
  Chaudron |   Chocolat |     Chrome |     Citron | Citrouille |  Coerulůum |
```

Compilation

Sous Windows : Faites une solution Visual Studio.
Sous Linux et MacOS: Un Makefile vous est fourni. Mettez tous les fichiers .h sous TP4/include, mettez tous les fichiers .cpp sous TP3/src et mettez le Makefile directement dans TP4. make -C path/to/TP4 all compile le programme. make -C path/to/TP4 run roule le programme.

Fuites de mémoire

Sous Windows :
Le fichier `debogageMemoire.h` est présent pour vous aider à vérifier s'il y a des fuites de mémoire dans votre code. Exécutez la solution en débogage pour qu'il fonctionne. Si une fuite de mémoire est détectée, un message sera affiché dans la fenêtre de sortie (Output) de Visual Studio.

Compilez avec -W4.
Sous Linux :
Utiliser Valgrind : Lancer la commande ci-dessous :
`valgrind --tool=memcheck --leak-check=yes ./PATH_VERS_PROGRAMME`
Pour installer valgrind : `sudo apt install valgrind`, `sudo pacman -S valgrind`, `etc.`
Si aucune fuite n'est détectée, vous devriez voir la ligne :
`All heap blocks were freed -- no leaks are possible.`

Spécifications générales

- Utilisez la liste d'initialisation pour l'implémentation de vos constructeurs.
- Ajouter le mot-clé *const* chaque fois que cela est pertinent.
- Ajouter des références dans les paramètres lorsque cela est pertinent
- Documentez le code source.
- Ne remettez que les fichiers .h et .cpp lors de votre remise
- Les entêtes de fichiers sont obligatoires
- Les fonctions doivent être documentées. Suivez les exemples des fonctions déjà documentées.
- Les fonctions doivent être implémentées dans le même ordre que la définition de la classe.
- **Le guide de codage sur Moodle doit être suivi.**