INFO-F105 – Langages de programmation I
Projet - Simulation d'une Intelligence Artificielle Générative
Phase III

Youcef Bouharaoua
youcef.bouharaoua@ulb.be

Année académique 2023-2024

1 Introduction

À travers cette phase du projet, vous apprendrez à définir et à implémenter des classes, ce qui vous aidera à comprendre comment ces éléments peuvent être structurés dans des programmes plus sophistiqués à l'avenir.

Durant cette phase, vous aurez également l'opportunité d'optimiser le code que vous avez développé lors des phases précédentes. C'est l'occasion d'améliorer l'efficacité et la clarté de vos programmes en révisant vos choix de conception initiaux. Chaque modification ou optimisation que vous apportez doit être justifiée de manière détaillée dans votre rapport . Ce processus de réflexion critique sur vos propres codes vous aidera à mieux comprendre les décisions de programmation et à développer un raisonnement plus approfondi sur la manière dont ces décisions affectent la performance globale de votre application.

2 La Phase III

Amélioration des parties précédentes

Dans cette phase du projet, nous vous demandons de revoir et d'améliorer les parties que vous avez déjà soumises, avec un focus particulier sur l'intégration et l'optimisation de l'utilisation des classes. Cette révision représente une occasion précieuse pour non seulement optimiser le code et améliorer la structure des données, mais également pour restructurer vos solutions en utilisant des classes afin de mieux organiser les fonctionnalités et les responsabilités au sein de votre programme. Vous êtes encouragés à identifier les sections de votre code où l'adoption d'une approche approche basée sur les classes et l'encapsulation pourrait apporter des bénéfices significatifs en termes de clarté, de maintenabilité et d'efficacité. Chaque modification apportée, notamment l'introduction ou la modification de classes, doit être clairement justifiée dans votre rapport final.

Exemples d'améliorations possibles

- **Utilisation des Classes:** Examinez les phases précédentes et évaluez où l'introduction de classes pourrait améliorer la clarté, la maintenabilité ou l'efficacité du code. Envisagez de structurer les données et les comportements associés à l'aide de classes là où vous constatez des regroupements logiques d'attributs et de méthodes. Cette démarche devrait être guidée par une analyse de vos besoins spécifiques en termes de fonctionnalités et d'organisation du code.
- **Optimisation du code:** Recherchez les occasions de réduire la complexité algorithmique ou d'améliorer la performance du code. Par exemple, si vous avez implémenté une recherche dans une liste de manière séquentielle, envisagez d'utiliser un ensemble ou une table de hachage pour accélérer les opérations de recherche.
- **Refactoring du code:** Simplifiez et clarifiez votre code en refaisant des parties qui sont confuses ou mal structurées. Par exemple, si vous avez des fonctions très longues ou avec de multiples boucles imbriquées, essayez de les diviser en fonctions plus petites avec des responsabilités claires.
- **Amélioration de la gestion des erreurs:** Assurez-vous que votre programme gère efficacement les erreurs, comme les entrées utilisateur incorrectes ou les fichiers qui ne peuvent pas être ouverts. Ajouter des vérifications robustes peut prévenir des défaillances et améliorer la stabilité de votre application.
- **Utilisation de fonctionnalités de langage avancées:** Si vous avez appris de nouvelles fonctionnalités de C++ ou Python après avoir soumis les premières parties du projet, appliquez ces connaissances pour rendre votre code plus efficace ou plus lisible.

Justification des choix

Il est crucial que chaque modification apportée à votre projet soit accompagnée d'une justification détaillée. Expliquez pourquoi vous avez choisi d'apporter ces changements et comment ils améliorent la fonctionnalité, la performance, ou la lisibilité de votre projet. Ces justifications sont essentielles pour évaluer votre capacité à analyser et à améliorer votre propre travail de manière critique.

Note: Ces améliorations doivent être intégrées de manière à conserver une cohérence globale du projet. Chaque choix technique doit être défendu par des arguments solides, basés sur des principes de programmation, de performance ou de meilleures pratiques.

3 Rapport d'analyse

Dans le cadre de cette troisième phase du projet, vous êtes invités à rédiger un rapport qui évaluera plusieurs aspects de votre implantation. Ce rapport devra couvrir les points suivants :

- Les choix que vous avez dû faire tout au long du projet (lors des trois phases)
- Une description des structures (ou des classes) que vous avez implémentées.
- Une discussion de votre optimisation.

Votre rapport devra non seulement refléter une compréhension technique des aspects abordés mais aussi présenter une réflexion critique sur vos choix de conception et d'implantation. Ce document représente une opportunité de démontrer votre capacité à analyser et à justifier les décisions prises durant le développement du projet.

Le rapport ne doit pas dépasser six pages et doit être soumis au format PDF en même temps que votre code source, dans le fichier zip nommé *phase3.zip*, avant la date limite du 26 mai 2024 à 23h59.

4 Évaluation

Lors de la correction, la pondération suivante sera utilisée :

- Rapport **/6**.
  - Forme **/1**.
    - Un rapport technique universitaire doit avoir un titre et une introduction.
    - La structure du rapport doit être claire. Il doit contenir plusieurs sections ayant des sous titres et des paragraphes.
    - Attention à l'orthographe.
    - Si le rapport contient des tables, figures ou listing de codes, elles doivent être numérotées et avoir une légende.
    - Respectez les consignes pour la longueur du rapport qui n'inclut pas d'éventuelles figures, page de garde, ni table de matière.
  - Contenu **/5**
    - Discussion de l'optimisation et réponses aux questions.
- Makefile **/2**.
- Fonctionnement correct du programme **/4**.
- Qualité du code **/8**.
  - Bonnes pratiques : noms de variables et fonctions clairs et cohérents, camelCase pour fonction et variables et PascalCase pour les structures et les classes.
  - ode bien structuré : utilisation pertinente de fonctions/méthodes et pas de duplication de code.
  - Bonne gestion de la mémoire. L'allocation à l'aide de new/new[] nécessite par la suite une désallocation à l'aide de delete/delete[]. Il ne doit pas avoir de fuite de mémoire (memory leak).
  - Efficacité du code en temps et en mémoire.

5 Délivrables

Il vous est demandé de soumettre un fichier zip qui contient au minimum :

- Vos fichiers .cpp et .hpp.
- Makefile.
- Un rapport sous format pdf.

Si vous avez utilisé des fichiers supplémentaires, vous devez également les inclure. Comme indiqué dans les consignes générales du projet, votre code doit compiler sans erreur avec .g++ sur les machines du NO.

Pour rappel la date limite de remise est fixée pour **le 26 mai 2024 à 23h59.** Une pénalité de -1/20 sera appliquée pour chaque jour de retard.