INFO-F105 – Langages de programmation 1

Projet 1 – Feedback

Mars 2025

Ce document est un retour commun pour tout le monde. Il est donc possible que certains points ne s'appliquent pas à votre travail. Lisez-le soigneusement et vérifiez s'il y a des choses à améliorer dans votre projet pour la phase 2.

1 Consignes

TL;DR Respectez les consignes.

- Implémentez les fonctions demandées dans les consignes (surtout pour la phase 3).
- Le fichier d'instructions doit être exécuté à la volée. Vous ne pouvez pas enregistrer les instructions dans un `std::vector` ou autre. Imaginez ce qu'il se passerait si votre fichier d'instructions faisait 10 Go.
- L'instruction `IFNZ` ne doit pas arrêter le programme, simplement ignorer l'instruction suivante si le registre est à 0.
- Compilez avec les flags demandés : `-Wall -Wextra -Wpedantic -std=c++23`
- Assurez-vous que votre code compile sans warning et passe tous les tests.
- Nommez votre fichier `.zip` avec votre matricule. Si votre matricule est `000123456`, nommez votre fichier '`123456.zip`'.

2 Structure de fichiers

TL;DR Structurez votre projet.

- Placez un `Makefile` à la racine de votre projet pour simplifier la compilation.
- Ne remettez que les fichiers sources (`.cpp`, `.hpp`), votre `Makefile` et votre rapport. Les autres fichiers (fichiers objets, exécutables, configuration de l'IDE, tests, …) ne sont pas nécessaires et prennent de l'espace inutilement.
- Nommez toujours votre fichier principal '`main.cpp`'.
- Il est inutile de créer un fichier `.hpp` pour votre fichier principal.
- Nommez vos fichiers sources (`.cpp` et `.hpp`) avec des minuscules. Si votre fichier ne contient qu'une classe, vous *pouvez* l'appeler du nom de la classe et laisser la majuscule.

3 Général

TL;DR Veillez tant que possible à la qualité de votre code.

- Nommez vos variables adéquatement.
- N'utilisez pas de variable globale, à moins qu'il n'y ait pas d'alternative propre.
- N'utilisez pas `break` et `continue`, à moins qu'il n'y ait pas d'alternative propre. Si un `break` s'avère nécessaire à la propreté du code, arrangez-vous pour le mettre en évidence dans votre code : soit en le mettant en tout début de boucle, soit en indiquant clairement les conditions d'arrêt de la boucle dans un commentaire.
- Mettez des `{ }` pour tous vos blocs, même s'ils ne contiennent qu'une seule ligne.
- Structurez votre code de façon à ce qu'on voie clairement où se trouve le code pour chaque instruction.
- N'utilisez pas les fonctionnalités du langage que vous ne maitrisez pas.
- Uniquement en respectant le point précédent : pensez au mot-clé `inline` (cf. synthèse 2).
- Pensez à la performance de votre code : #instructions, mémoire utilisée, etc.
- Ne mélangez pas anglais et français : soit l'un, soit l'autre.

4 Typage

TL;DR Utilisez les types adéquats.

- N'utilisez pas `int` pour un entier non signé dans une boucle `for`.
- Un `uint16_t` ne dépassera jamais `65535`.
- Ne comptez pas sur le fait qu'un `int` peut contenir un `uint16_t`. Utilisez plutôt un type de taille fixe lorsque vous manipulez vos nombres au niveau binaire.

5 Modularité

TL;DR Veillez tant que possible à la modularité de votre code.

- Évitez les très longues fonctions.
- Séparez les morceaux de codes logiquement liés en différentes fonctions.
- Évitez les *valeurs magiques* dans votre code (ex : `65535`). Utilisez plutôt des constantes (ou mieux `constexpr`) au début de votre fichier.

6 Formattage

TL;DR Veillez tant que possible à la lisibilité de votre code.

- Évitez les *one-liners*[1] lorsque cela nuit à la lisibilité du code.
- Évitez les espaces inutiles, par exemple avant une ‘;’ à la fin d’une ligne de code.
- Essayez de garder les espaces utiles à la lisibilité. Par exemple :

```
for (unsigned i = 0; i < SIZE; i += 2) {
   std::cout << i << std::endl;
}
```

- De manière générale, gardez votre code cohérent : indentation, espaces, ‘`std::`’, etc.
- Séparez les morceaux de code logiquement distincts par une ligne vide.
- Nommez vos constantes en majuscules.

7 Rapport

TL;DR Simplifiez la vie du correcteur qui décidera de votre note.

- Remettez votre rapport au format PDF.
- Restez concis·e dans vos explications.
- Montrez que vous comprenez votre implémentation :
  - Vos astuces pour rendre votre code à la fois plus simple et plus performant.
  - Ce qu’il se passe au niveau de la mémoire lors de l’exécution.
  - Comment s’assurer que les cas limites sont couverts.

8 IA

TL;DR Faites votre projet intelligemment.

Certains projets ont de manière évidente été écrits par une IA.

Il n’est pas permis de recopier telle quelle une réponse donnée par l’IA. Il est autorisé de demander conseil à une IA, mais vous devez impérativement écrire le code vous-même.

Rappelez-vous que la fraude est un motif d’exclusion et que cela peut mener à un refus d’inscription dans tout établissement d’enseignement supérieur de la Communauté française pour une durée de trois ans.

De plus, si vous n’écrivez pas le code vous-même ou ne suivez pas le processus normal de rédaction du projet, vous ne serez pas capable de répondre aux questions d’examen dédiées au projet et n’aurez donc pas de point pour le projet.

[1] Des lignes de codes qui font plusieurs choses à la fois.