

Mini-Projet INFO3205 — Paradigmes de programmation

Université de Moncton
Département d'informatique

À rendre le 10 décembre avant 23h59

Contenu

1	Attribution des langages	2
2	Importance et structure du rapport	3
3	Implémentation	3
4	Consignes importantes	4
5	Problèmes disponibles (en choisir 1)	5
5.1	Problème 1 : Nombres complexes	5
5.2	Problème 2 : Nombres rationnels	6

1 Attribution des langages

Table 1: Attribution des langages par étudiant

ID Étudiant	Langages attribués	
1	A00215640	C++ / Java
2	A00217236	C++ / Rust
3	A00218917	Rust / Go
4	A00220976	Rust / Go
5	A00221780	C++ / Java
6	A00222852	Java / Rust
7	A00223594	C++ / Go
8	A00223643	Julia / Rust
9	A00224202	C++ / Julia
10	A00225427	Julia / Rust
11	A00226112	Rust / Go
12	A00227139	C++ / Julia
13	A00227556	C++ / Java
14	A00228070	Julia / Lua
15	A00228391	Julia / Rust
16	A00228758	Java / Julia
17	A00228930	C++ / Rust
18	A00230745	Rust / Lua
19	A00231380	C++ / Lua
20	A00232689	C++ / Rust
21	A00235273	C++ / Go
22	A00245884	C++ / Lua
23	A00246055	Java / Lua
24	A00246822	C++ / Julia
25	A00248597	Java / Rust
26	A00248912	Java / Julia
27	A00249476	Java / Go
28	A00251586	Java / Go
29	A00252327	C++ / Go
30	A00252529	Java / Go
31	A00253551	Java / Rust
32	A00254738	C++ / Lua

Objectif. Ce projet vise à vous aider à comprendre en profondeur les paradigmes de programmation en travaillant avec deux langages différents. L'objectif est de vous confronter à des choix de conception, à des différences syntaxiques et sémantiques, et à des approches variées pour résoudre des problèmes concrets. Vous devez respecter **strictement les langages qui vous ont été assignés** : aucune substitution n'est permise.

Interdiction d'utiliser des outils d'IA générative (ChatGPT, Copilot, etc.) pour tout le projet.

2 Importance et structure du rapport

Importance du rapport. Le rapport est un élément central du projet. Il doit être rédigé en \LaTeX selon la template fournie et respecter les bonnes pratiques de mise en page : titres hiérarchisés, tableaux clairs, figures si nécessaire, et surtout une **présentation simple et lisible**. La clarté et la cohérence sont essentielles : évitez les paragraphes trop longs, et utilisez des listes pour structurer vos idées.

Structure attendue du rapport.

1. **Introduction** : objectifs, contexte, présentation des deux langages.
2. **Analyse des langages** : paradigmes, typage, gestion mémoire, exceptions, outils.
3. **Comparaison** : tableau synthétique + discussion argumentée.
4. **Implémentations** : description des problèmes, choix de conception, extraits de code.
5. **Discussion** : difficultés rencontrées, différences marquantes.
6. **Conclusion** : synthèse et recommandations.

3 Implémentation

Ne pas insérer la totalité du code dans cette section du rapport. Seulement des extraits du code au besoin !

Vous devez implanter **un problème** dans les **deux langage**. Cela signifie que vous produirez **deux programmes au total**. Chaque problème doit être résolu en exploitant les paradigmes propres au langage choisi.

Approche recommandée. Essayez de **varier les paradigmes** entre les deux problèmes pour chaque langage. Par exemple, si vous utilisez une approche orientée objet pour le premier problème, tentez une approche plus fonctionnelle ou modulaire pour le second. Cela vous permettra de comparer non seulement les langages, mais aussi la manière dont leurs paradigmes influencent la conception.

4 Consignes importantes

Consignes Importantes — Règles générales

- Rapport maximum : **10 pages**. Tout dépassement pourra être pénalisé.
- Code source : seules les bibliothèques standards sont autorisées (par ex. `<vector>`, `<string>` en C++). **Pas de bibliothèques externes (sauf indication contraire)**.
- Code bien commenté, lisible et structuré. Inclure un README avec instructions de compilation et d'exécution (commandes précises).
- Fournir des tests simples permettant de vérifier la fonctionnalité (exemples d'entrée/sortie).
- Rapport en \LaTeX (qualité de la mise en page, sections, tableaux, bibliographie).
- Livrables attendus : (1) archive du code source, (2) rapport PDF (max 10 pages), (3) README (compilation/exécution), (4) scripts de test ou exemples.
- Pénalités appliquées pour : non-reproductibilité, absence de README, non-respect des I/O spécifiés, plagiat, dépassement de 10 pages, code non commenté.
- Vous devez choisir **un problème** et l'implémenter dans **deux langages différents**.
- Pour chaque langage, utilisez **deux paradigmes différents** :
 - Exemple : orienté objet pour le premier problème, impératif ou fonctionnel pour le second.
 - Justifiez brièvement votre choix (commentaire ou documentation).
- Lorsque le langage le permet, **surchargez les opérateurs**. Sinon, utilisez des méthodes explicites.
- Gérez les erreurs (division par zéro, etc.).

Remarque sur la surcharge des opérateurs

- Langages avec surcharge : C++, Julia, Rust, Lua (via métaméthodes).
- Langages sans surcharge : Java, Go (pour types personnalisés).
- Si la surcharge est impossible, utilisez des méthodes (`add`, `sub`, `mul`, `div`, `equals`).

Structure du dossier à soumettre.

```

PrpjetINF03205ZoubairMlikaCppJava/
├─ rapport/
│   └─ main.tex
├─ src/
│   ├── CppComplex/[les fichiers sources]
│   └── JavaComplex/[les fichiers sources]
└─ README.md
  
```

Ressources officielles (documentation)

- C++: <https://en.cppreference.com>
- Java: <https://docs.oracle.com/javase>
- Julia: <https://docs.julialang.org>
- Lua: <https://www.lua.org/manual/>
- Go: <https://go.dev/doc>
- Rust: <https://doc.rust-lang.org>

Grille d'évaluation (Barème : 140 pts)

Barème (140 points): Note finale $\min(100, \text{votrenote})$

A. Rapport (50 pts)

- Structure, clarté, qualité rédactionnelle : 10 pts
- Analyse des paradigmes et concepts (typage, mémoire, erreurs) : 15 pts
- Comparaison argumentée (tableaux, exemples) : 15 pts
- Qualité L^AT_EX (sections, tableaux, bibliographie) : 10 pts

B. Implémentations (50 pts)

- Fonctionnalité correcte : 20 pts
- Lisibilité, modularité, commentaires, tests : 15 pts
- Respect des idiomes / paradigmes du langage : 15 pts

C. Bonus (40 pts)

5 Problèmes disponibles (en choisir 1)

5.1 Problème 1 : Nombres complexes

On souhaite modéliser un nombre complexe :

$$z = a + bi,$$

où a est la partie réelle et b la partie imaginaire.

Structure Complex

Votre structure doit contenir :

- Deux attributs : `re` et `im`
- Un constructeur par défaut.
- Un constructeur avec paramètres.
- Un destructeur si allocation dynamique.

Fonctions / opérateurs à implémenter (utiliser la surcharge quand c'est possible)

1. Addition : $z_1 + z_2 = (a + c) + (b + d)i$.
2. Soustraction : $z_1 - z_2 = (a - c) + (b - d)i$.
3. Multiplication : $z_1 z_2 = (ac - bd) + (ad + bc)i$.
4. Division :

$$\frac{z_1}{z_2} = \frac{(ac + bd) + (bc - ad)i}{c^2 + d^2}.$$
5. Conjugué : $\bar{z} = a - bi$.
6. Égalité : $z_1 == z_2 \iff a_1 = a_2 \text{ et } b_1 = b_2$.

Exemples de jeux de tests

- $1 + i$, $2 - 3i$, $0 + 0i$, nombres purement réels, purement imaginaires.
- Très grands nombres (10^9), très petits (10^{-12}).
- Division par un complexe presque nul.

Bonus (40 pts) : choisissez un langage purement fonctionnel et implémentez `Complex`.

5.2 Problème 2 : Nombres rationnels

On souhaite modéliser un nombre rationnel :

$$\frac{p}{q}, \quad q \neq 0.$$

Structure Rational

Votre structure doit contenir :

- Deux attributs : `num` (numérateur) et `den` (dénominateur).
- Un constructeur par défaut.
- Un constructeur avec paramètres (avec normalisation).
- Un destructeur si allocation dynamique.

Normalisation

- Réduire la fraction avec le PGCD (implémenter ou utiliser une bibliothèque externe).
- Assurer un dénominateur positif.
- Gérer les signes : $\frac{-2}{-4} = \frac{1}{2}$.

Opérations à implémenter

1. Addition : $\frac{a}{b} + \frac{c}{d} = \frac{ad + bc}{bd}$.
2. Soustraction : $\frac{a}{b} - \frac{c}{d} = \frac{ad - bc}{bd}$.
3. Multiplication : $\frac{a}{b} \cdot \frac{c}{d} = \frac{ac}{bd}$.
4. Division : $\frac{a}{b} \div \frac{c}{d} = \frac{ad}{bc}$.
5. Égalité : $\frac{a}{b} == \frac{c}{d} \iff ad = bc$.
6. Forme mixte : $\frac{a}{b} = q + \frac{r}{b} \iff a = qb + r$.

Exemples de jeux de tests

- $\frac{1}{2} + \frac{1}{3} = \frac{5}{6}$
- $\frac{2}{3} \cdot \frac{3}{4} = \frac{1}{2}$
- $\frac{1}{2} \div \frac{1}{4} = 2$
- $\frac{6}{5} = 1 + \frac{1}{5}$
- Cas négatifs, grands entiers, réduction automatique.

Bonus (40 pts) : choisissez un langage purement fonctionnel et implémentez Rational.