

## Énoncé TP3

### *Analyse sémantique*

#### Directives :

- Vous devez remettre votre travail pour le dimanche 16 novembre à 23h59. Les retards sont acceptés jusqu'au lundi suivant à 07h59 avec une pénalité de 5% sur la note finale du travail par heure de retard. La note de 0% vous sera automatiquement attribuée si vous n'avez pas remis votre travail avant 08h00.
- Il s'agit d'un travail individuel. On vous encourage à collaborer sur les forums du cours, tant que vous remettez votre propre travail. Les sanctions habituelles concernant le plagiat s'appliquent.
- Vous devez compléter le code *C#* qui vous est fourni. Toutes les modifications jugées nécessaires ou pertinentes sont permises.
- Aucune librairie externe n'est autorisée. Vous êtes limités à la librairie standard *C#*.
- La pondération pour ce travail est de 15% de la note finale du cours. Les critères de correction sont disponibles en annexe A.
- L'utilisation de l'IA générative est autorisée, quoique fortement déconseillée. Les modalités de citation exigées sont présentées en annexe B.

#### Contexte :

Pour cette troisième phase du projet de session, vous devez compléter l'implémentation d'un analyseur sémantique pour le langage fictif élaboré lors de la phase précédente.

L'objectif de l'analyseur sémantique est d'annoter un arbre d'analyse en exécutant des actions sémantiques. Les actions peuvent être utilisées pour effectuer des vérifications statiques ou encore produire du code intermédiaire. Ce dernier sera directement repris pour la prochaine phase du projet.

Une librairie de classe est fournie pour accélérer le développement de votre analyseur sémantique. Vous devez simplement ajouter le code nécessaire dans la méthode `Program.Main` du projet `Core` pour compléter le travail.

Aucune documentation du code de départ n'est fournie autre que les quelques commentaires inclus dans les méthodes. Les algorithmes, procédures et transformations à votre disposition reprennent directement les concepts explorés en classe.

## Tâches :

### 0. Mise en place de l'environnement de développement.

Un nouveau projet **SemanticAnalysis** est disponible afin d'augmenter la solution partagée pour les phases précédentes. Vous devez soit inclure ce projet dans votre environnement existant, soit copier votre projet **Core** existant dans un nouvel environnement en prenant soin d'ajouter les nouvelles dépendances.

Le code qui vous est fourni a été conçu avec .NET 9.0 et la version *C#* 13.0.

### 1. Réparations.

L'analyseur syntaxique doit être fonctionnel pour progresser dans le processus de compilation de votre langage. Corrigez tous les soucis encore présents dans votre dernière remise.

Vous pouvez récupérer jusqu'à la moitié des points perdus à la remise précédente pour une seule tâche générale parmi les suivantes :

- Recherche
- Grammaire
- Transformation
- Fichiers

Indiquez clairement dans votre rapport la tâche choisie pour la récupération des points et tous les ajustements nécessaires pour améliorer votre travail.

### 2. Recherche.

Avant de créer votre propre langage, il est essentiel de consulter quelques ressources pertinentes pour en apprendre davantage sur le domaine.

Menez quelques recherches de base pour tenter de récupérer des informations supplémentaires à propos des règles sémantiques de quelques langages réels pertinents pour votre contexte.

Présentez vos trouvailles dans votre rapport. Motivez vos choix pour votre langage. Commentez votre expérience en résumant rapidement les pistes empruntées et les difficultés rencontrées.

### 3. Création des règles sémantiques.

Les règles sémantiques définissent le coeur d'un langage de programmation. Définissez des actions sémantiques adaptées à votre langage dans votre programme. N'hésitez pas à revenir sur certaines décisions créatives dans les règles lexicales et syntaxiques du langage.

Vos règles sémantiques doivent minimalement couvrir les opérations arithmétiques et les structures de contrôle de votre langage. Ajoutez également les vérifications statiques pertinentes à travers les actions.

Expliquez brièvement dans votre rapport le rôle de chaque règle ainsi que les modifications apportées aux règles lexicales ou syntaxiques. Identifiez la classe de votre grammaire attribuée.

### 4. Transformation des règles sémantiques.

Vous devez ensuite reprendre les étapes de traitement de l'analyseur sémantique tel que vu en classe.

Le code fourni contient les algorithmes et procédures nécessaires pour appliquer les étapes dans le bon ordre.

### 5. Fichiers source.

Créez quelques fichiers de code dans votre langage pour tester votre implémentation. Réutilisez votre dernière remise pour obtenir la séquence de dérivations à traiter par l'analyseur sémantique et ainsi calculer la valeur des attributs.

Présentez vos résultats dans votre rapport et comparez les attributs calculés avec vos attentes. Au moins un fichier devrait contenir une erreur sémantique.

### 6. *Bonus facultatif* : Appels de fonctions.

Ajoutez tout le code nécessaire pour produire des instructions de code à trois adresses qui représentent des appels de fonctions via des règles sémantiques. Votre implémentation devrait également inclure les vérifications statiques pertinentes et le backpatching lorsque applicable.

Complétez quelques exemples dans un fichier source et comparez les résultats obtenus avec ceux attendus. Discutez brièvement de vos choix et de vos observations dans votre rapport.

**Rapport :**

Votre texte doit être rédigé en français. Vous pouvez évidemment utiliser les termes techniques originaux dans la langue de votre choix. Il n'est pas nécessaire de traduire les figures ou autre matériel référencé.

Vous pouvez structurer votre document de la manière qui vous convient. SVP ne pas inclure de page titre, table des matières ni toute autre formalité dans votre rapport.

**Remise :**

Vous devez remettre dans la boîte de dépôt un fichier **\*.zip** qui contient l'entièreté de votre travail, y compris les fichiers de code que vous n'avez pas modifiés et tout le matériel pertinent pour vos tests personnels. Assurez-vous de remettre seulement le code sans les répertoires **bin**, **obj**, **.vs**, **.idea**, etc.

Vous devez également ajouter à votre remise votre rapport au format PDF dans lequel vous indiquerez la version **.NET** et **C#** utilisée.

**Bon travail !**

## Annexe A : Critères de correction.

Votre note finale pour ce travail est la somme des notes de chaque tâche, pour un minimum de 0/15 et un maximum de 16/15.

| Tâche          | Min | Max | Critère   | Pénalité |
|----------------|-----|-----|---|----------|
| Recherche      | 0   | 4   | Recherche d'information insuffisante  | -4       |
| Actions        | -2  | 4   | Les règles ne fonctionnent pas  | -6       |
|                |     |     | Pertinence des règles   | -3       |
|                |     |     | Ne couvre pas toutes les règles de la grammaire   | -1       |
| Transformation | -2  | 2   | Transformations absentes ou incomplètes   | -4       |
|                |     |     | Conversion en SDT   | -1       |
|                |     |     | Calcul des attributs  | -1       |
| Fichiers       | -1  | 2   | Fichiers de code source absents   | -2       |
|                |     |     | Les fichiers ne permettent pas de tester toutes les actions   | -1       |
|                |     |     | Aucune erreur sémantique  | -1       |
| Rapport        | -2  | 3   | Le rapport est absent ou incomplet  | -5       |
|                |     |     | Qualité et présentation générale  | -5       |
|                |     |     | Résumé de la recherche  | -3       |
|                |     |     | Explications des règles   | -3       |
|                |     |     | Identification de la classe de grammaire attribuée  | -2       |
|                |     |     | Discussion sur les résultats obtenus  | -2       |
| Bonus          | 0   | 1   | Bonus facultatif non complété   | -1       |
|                |     |     | Ne répond pas aux besoins suggérés  | -1       |
|                |     |     | Intégration maladroite ou forcée  | -1       |
|                |     |     | La solution n'est pas suffisamment créative   | -1       |
| Général        | -16 | 0   | Le code remis ne compile pas  | -16      |
|                |     |     | Modifications non autorisées  | -16      |
|                |     |     | Erreurs pendant l'exécution   | -16      |
|                |     |     | Qualité du code   | -16      |
|                |     |     | Répertoires résiduels ( <code>bin</code> , <code>obj</code> , <code>.vs</code> , <code>.idea</code> ) inclus dans la remise | -2       |
|                |     |     |   |          |

## **Annexe B : Modalités de citation pour IA générative.**

*Une utilisation de l'IA générative qui n'a pas été autorisée dans les consignes d'une évaluation ou qui n'est pas mentionnée ou citée correctement constitue une infraction relative aux études selon le Règlement disciplinaire à l'intention des étudiants et étudiantes de l'Université Laval. Le non-respect de cette exigence constitue une infraction relative aux études et pourrait mener à l'application des sanctions prévues au Règlement disciplinaire.*

Il n'est pas demandé de documenter une utilisation générale pour votre compréhension personnelle. Si vous choisissez d'utiliser l'IA générative pour vous assister directement dans la réalisation de ce travail, vous devez ajouter une section dans votre rapport en conséquence. Cette section doit mentionner le nom et la version des modèles consultés.

Il est également requis de fournir l'intégralité (dans la mesure du possible) de toutes les conversations sous forme de captures d'écrans. Vos questions et les réponses du modèle doivent être entièrement visibles.

Identifiez clairement les sections du code de votre remise qui sont générées automatiquement, même si vous avez apporté des modifications à la sortie suggérée par le modèle.

Finalement, ajoutez dans votre rapport un paragraphe décrivant votre processus d'utilisation de l'IA générative ainsi que les mesures adoptées pour valider la véracité des propos des modèles.