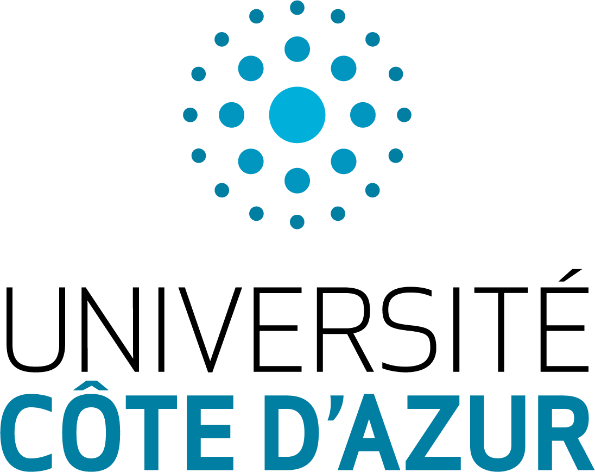
Rapport de Projet de compilation

Compilateur miniC

Langage miniC vers DOT

**2025**



**Réalisé par :**

**CARRIBA NOSRATI Daniel et DI PLACIDO Bence**

**SOMMAIRE**

Table des matières

[1. INTRODUCTION 3](#_Toc198754652)

[2. MÉTHODE DE TRAVAIL 4](#_Toc198754653)

[3. ARBORESCENCE DU PROJET 4](#_Toc198754654)

[4. DÉVELOPPEMENT 5](#_Toc198754655)

[4.1. ANALYSE LEXICALE 5](#_Toc198754656)

[4.2. ANALYSE SYNTAXIQUE: 5](#_Toc198754657)

[4.3. ANALYSE SÉMANTIQUE 6](#_Toc198754658)

[4.4. GÉNÉRATION DE CODE 7](#_Toc198754659)

[5. CE QUE LE PROGRAMME PERMET 8](#_Toc198754660)

[6. ERREURS DÉTECTÉS PAR L’ANALYSE SÉMANTIQUE 8](#_Toc198754661)

[6.1. ERREURS 8](#_Toc198754662)

[6.2. WARNINGS 8](#_Toc198754663)

[7. CONCLUSION 9](#_Toc198754664)

# 

# 1. INTRODUCTION

Dans le cadre de l’UE Compilation, nous avons dû réaliser un compilateur du langage C capable de traduire ce dernier en langage intermédiaire DOT, sous formes de graphes.

En réalisant ce projet, nous avons pu mettre en pratique les connaissances acquises durant le cours et les TD.

Nous avons essayé d’avancer dans le rythme du cours afin de pouvoir mettre en œuvre les connaissances acquises.

Dans un premier temps nous avons complété l’analyse lexicale (ANSI-C.l) qui génère les tokens. Par la suite, nous avons réalisé l’analyse syntaxique a l’aide des définitions donner dans le fichier miniC., durant l’évolution du projet, nous avons modifier les règles de la grammaire (voir [**analyse syntaxique**](#_4.2._ANALYSE_SYNTAXIQUE)). De plus, nous avons implémenter l’analyse sémantique pour respecter aux maximum les attentes du langage miniC et généré le code DOT conformément aux consignes.

Pour les différentes parties du projet, nous avons utilisé les technologies respectives :

* LEX , pour l’analyse lexicale
* YACC, pour l’analyse syntaxique
* C, pour la sémantique et la génération de code

Pour ce projet, nous avons dû réaliser un compilateur a deux parses. Lors de la première parse, le compilateur détecte les erreurs sémantiques et remplie une structure de donnés qui pour la seconde parse va être utiliser pour faire la traduction en DOT.

# 2. MÉTHODE DE TRAVAIL

TODO

# 3. ARBORESCENCE DU PROJET

TODO

# 4. DÉVELOPPEMENT

## 4.1. ANALYSE LEXICALE

TODO

## 4.2. ANALYSE SYNTAXIQUE :

TODO

## 4.3. ANALYSE SÉMANTIQUE

TODO

## 4.4. GÉNÉRATION DE CODE

# 5. CE QUE LE PROGRAMME PERMET

# 6. ERREURS DÉTECTÉS PAR L’ANALYSE SÉMANTIQUE

## 6.1. ERREURS

TODO

## 6.2. WARNINGS

TODO

# 7. CONCLUSION

TODO