PLD
Compilateur

Présentation du 06/04/2022

Hexanôme H4212:
BOUVIER Julien, CRISTOFORONI Stella, DELEGLISE Benoit, GUIGAL Allan, MARC Quentin, TOURNADE Aurélien, BOYER Maxime

Plan

1

Fonctionnalités et limites

Fonctionnalités implémentées

Notre compilateur est capable de comprendre un programme C composé :

- d'un seul fichier source (les directives pré-processing sont ignorées)
- d'un main et son retour
- d'initialisations de variables (pas forcément sur une seule ligne, les variables peuvent être initialisées n'importe où)
- d'affectations (tout type d'affectations)
- de conditions (if/else)
- de boucles
- de fonctions
- d'opérations arithmétiques

Analyse avancée

Notre compilateur comprends des fonctionnalités avancées telles que :

- Shadowing
- Scope
- Propagation de constante dans les expressions

Limites du compilateur

Notre compilateur est un projet universitaire. A ce titre, il ne réalise qu'une petite partie des fonctionnalités d'un vrai compilateur. Ainsi vous ne pouvez pas :

- compiler un programme contenant plusieurs sources
- utiliser les directives pré-processeur
- utiliser les pointeurs
- utiliser des structures
- utiliser des fonctions de plus de 6 arguments
- utiliser des types de variables autres que int et char

2

Fonctionnement du compilateur

Structure du compilateur

Note compilateur a été construit selon la structure suivante:

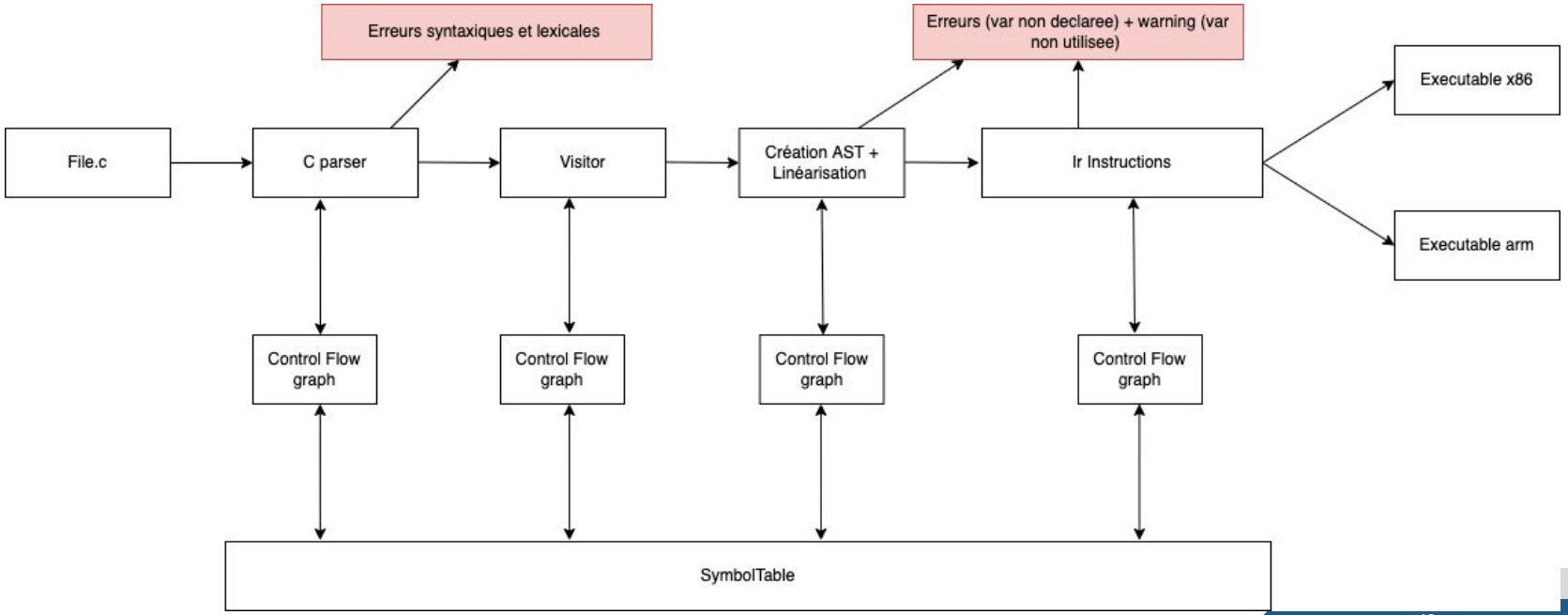


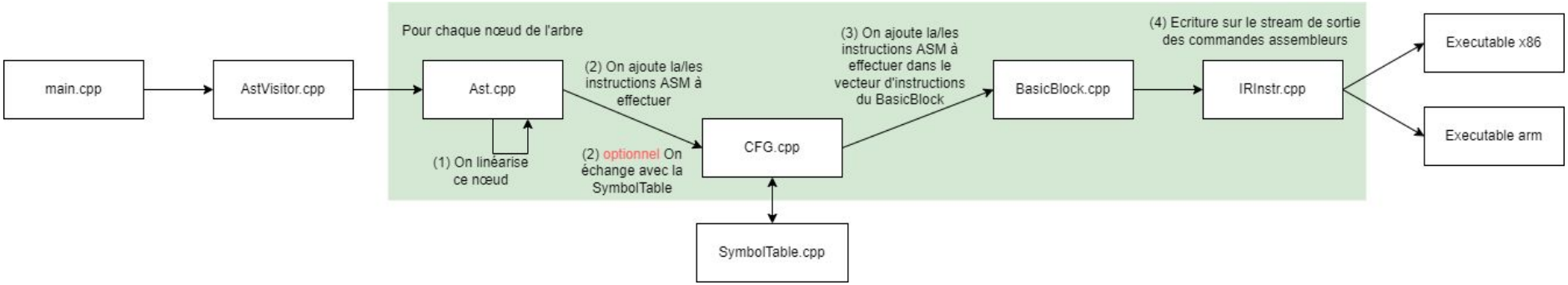
Table des symboles

Index	Name	Scope name	Scope table level	Scope current level	Scope context	Type	Additional	State	IsConst

```
***   Actual Symbol Table   ***
-------------------------------------------------------------------------------------------------
|   Index   ;    Name   ;   Scope name   ;   Scope table level   ;   Scope current level   ;   Scope context   ;   Ty
-------------------------------------------------------------------------------------------------
|  64  ;  !retvalue_main  ;  main  ;  TL=0  ;  CL=0  ;  Ctx=0   ;  0  ;  0  ;  1  ;  0  |
|  4  ;  !tmp0_main  ;  main  ;  TL=0  ;  CL=0  ;  Ctx=0 0   ;  0  ;  0  ;  1  ;  0  |
|  28  ;  !tmp1_main  ;  main  ;  TL=0  ;  CL=0  ;  Ctx=0 0   ;  0  ;  0  ;  1  ;  0  |
|  36  ;  !tmp2_main  ;  main  ;  TL=0  ;  CL=0  ;  Ctx=0 0   ;  4  ;  0  ;  1  ;  0  |
|  40  ;  !tmp3_main  ;  main  ;  TL=0  ;  CL=0  ;  Ctx=0 0   ;  0  ;  0  ;  1  ;  0  |
|  44  ;  !tmp4_main  ;  main  ;  TL=0  ;  CL=0  ;  Ctx=0 0   ;  0  ;  0  ;  1  ;  0  |
|  52  ;  !tmp5_main  ;  main  ;  TL=0  ;  CL=0  ;  Ctx=0 0   ;  4  ;  0  ;  1  ;  0  |
|  56  ;  !tmp6_main  ;  main  ;  TL=0  ;  CL=0  ;  Ctx=0 0   ;  0  ;  0  ;  1  ;  0  |
|  60  ;  !tmp7_main  ;  main  ;  TL=0  ;  CL=0  ;  Ctx=0 0   ;  0  ;  0  ;  1  ;  0  |
|  8  ;  a_main  ;  main  ;  TL=0  ;  CL=0  ;  Ctx=0 0   ;  0  ;  0  ;  1  ;  0  |
|  1  ;  getchar@PLT_GLOBAL  ;  GLOBAL  ;  TL=0  ;  CL=0  ;  Ctx=0   ;  2  ;  0  ;  2  ;  0  |
|  1  ;  main_GLOBAL  ;  GLOBAL  ;  TL=0  ;  CL=0  ;  Ctx=0   ;  0  ;  0  ;  2  ;  0  |
|  1  ;  putchar@PLT_GLOBAL  ;  GLOBAL  ;  TL=0  ;  CL=0  ;  Ctx=0   ;  3  ;  0  ;  2  ;  0  |
|  24  ;  tableau_main  ;  main  ;  TL=0  ;  CL=0  ;  Ctx=0 0   ;  0  ;  16  ;  0  ;  0  |
-------------------------------------------------------------------------------------------------
```

```c
#include <stdio.h>
int main() {
    int a = 5;
    int tableau[2] = {2,4};
    return 0;
}
```

Création du code Assembleur



3

Démonstration

4

Gestion de projet

Organisation

L'équipe s'est organisée de la manière suivante:

- **Début de séance**: "daily stand up" (briefing général sur ce qui a été fait et sur les tâches à réaliser pour la séance)
- **Mesure de l'avancement**: Coordination de l'avancement sur discord.
- **Tâches à prioriser**: regroupement de tâches / fonctionnalités en tickets, estimation du poids du ticket en équipe.
- **Supervision du travail**: Relecture du code des tickets terminés sauf si réalisé en Pair programming.

Répartition des tâches

Phase 1:

- Création des tests: 1 personne
- Création SymbolTable: 2 personnes
- Création backend: 2 personnes
- Parsing & Erreurs: 2 personnes

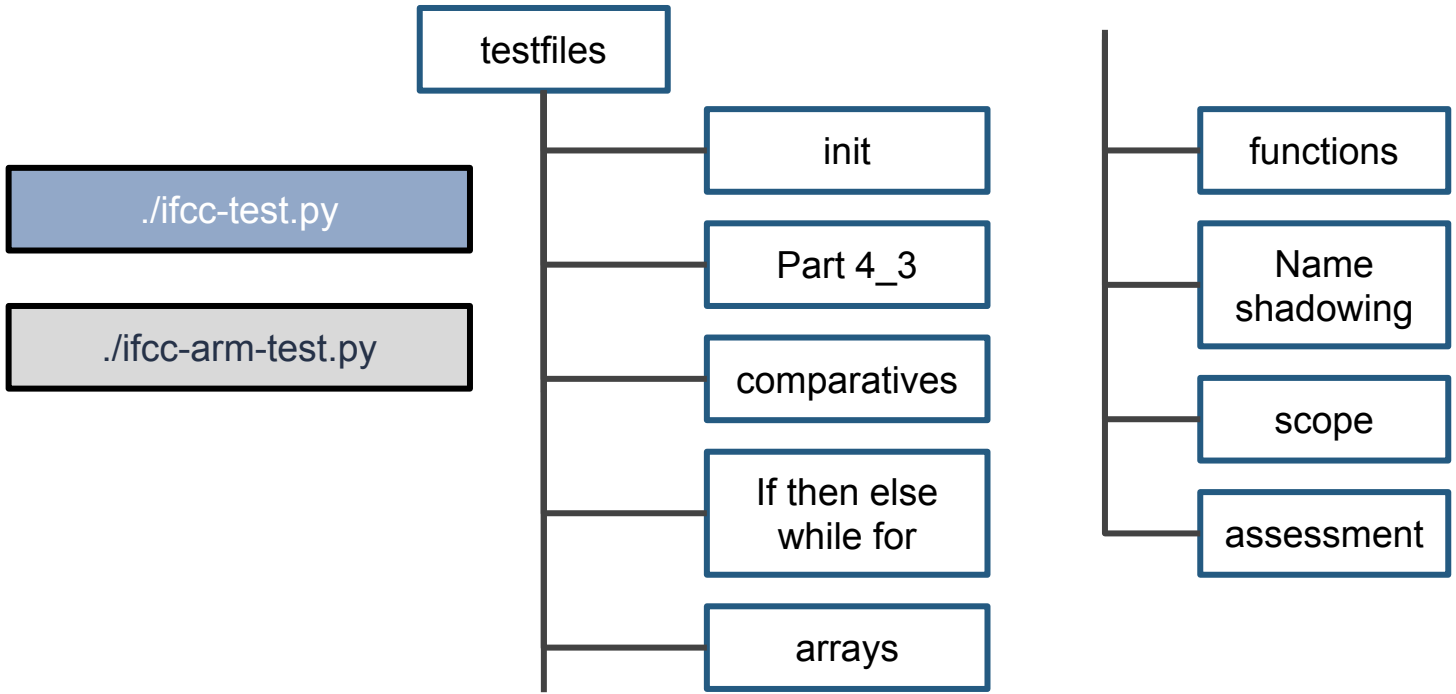
Phase 2:

- Création des tests: 1 personne
- Implementation arm: 1 personne
- Gestion des erreurs: 2 personnes
- Implémentation des conditions / boucles / fonctions: 1 personne
- Implémentation des Lvalue / gestion des tableaux: 2 personnes

Les tests - TDD

- Maintien difficile des fonctionnalités correctes avec arm x86 à l'ajout d'une fonctionnalités
- Gestion des installations en fonction des distributions
- Structuration des tâches
- Monitoring simple et rapide des fonctionnalités

Les tests - architecture



4

Conclusion

Conclusion

Apprentissages:

- Test driven development
- Gestion d'un projet aussi technique à 7
- Initier la structure avec autant de personnes
- Spécialisation sur des domaines techniques
- Gestion des exceptions de compilation
- Debug d'un projet aussi technique

Pistes d’amélioration

Pistes d’amélioration:

- Implémentation des pointeurs
- Meilleure optimisation de la propagation de variables / constantes
- Implémentation d’autres types
- Utilisation de structures