Ecole Supérieure Des Sciences
et De La Technologie De Hammam Sousse

Chapitre 1 : Introduction à la Compilation

Sonia KEFI

2 éme LFI

Année universitaire : 2022-2023

Plan du cours

- Chapitre 1 : Introduction
- Chapitre 2 : Analyse Lexicale
- Chapitre 3 : Analyse Syntaxique
- Chapitre 4 : Analyse Sémantique

Prerequis

- Théorie des langages formels: expressions régulières, automates, grammaires, ambiguïté
- Langages de programmation
- Algorithmique

Chapitre 1 :

Introduction

Introduction (1)

- Si on vous demande d'écrire un programme capable d'évaluer (de calculer) des expressions arithmétiques donnée par un utilisateur.
- Par exemple : (24+45.5)/87*1.2E3

- Comment vous allez procéder ?

Introduction (2)

- Ecrire un sous-programme permettant de :
  - reconnaître un entier
  - reconnaître un réel
  - reconnaître une addition
  - ….

INTRODUCTION (3)

- Il faut suivre des méthodes et des techniques afin de résoudre ce type de problème :

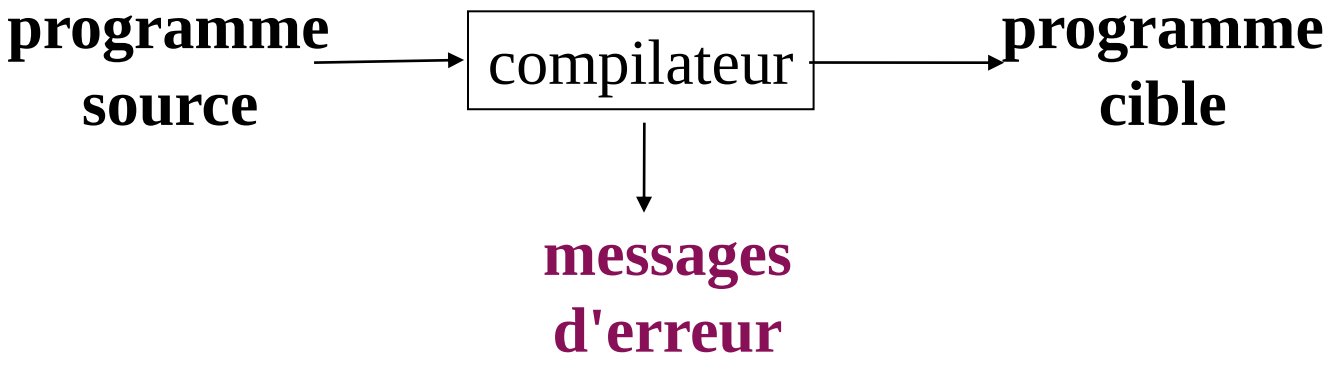
➔ C'est le rôle du **COMPILATEUR**

Qu'est ce qu'un compilateur

- Au début de l'informatique, on programmait directement sur les ordinateurs en langage machine.
- C'était très fastidieux pour le programmeur.
- Pour cela il a fallu construire des programmes qui traduisent des énoncés exprimés dans le langage de **haut niveau** utilisé par les programmeurs, ce qu'on appelle le **langage source**, en instructions pour la **machine cible**.
- Les programmes effectuant ce genre d'opération s'appellent des **compilateurs**

Définition d'un Compilateur

- C'est un programme qui traduit un programme écrit dans un langage source vers un langage cible en indiquant les erreurs éventuelles que pourrait contenir le programme source.
- **Exemples :**
  - D'un langage de haut niveau « C, C++, … » vers un langage d'assembleur.
  - Du langage d'assembleur vers du code binaire.



Suite

- **Premier compilateur :** compilateur Fortran de J. Backus (1957)
- **Langage source :** langage de haut niveau (C, C++, Java, Python…)
- **Langage cible :** langage de bas niveau (assembleur, langage machine)

SUITE

Récapitulation

- Langage machine (le langage binaire) : c'est le seule langage qui est compréhensible par la machine mais c'est difficile à utiliser et à le manipuler par l'homme.

- Langage de bas niveau (le langage d'assemblage) : c'est un langage alphanumérique. Il est plus lisible, mais il reste toujours difficile à utiliser.

- Langage de haut niveau : il est proche du langage humain. On utilise un compilateur ou un interpréteur pour traduire un programme en langage machine.

Interpréteur versus Compilateur

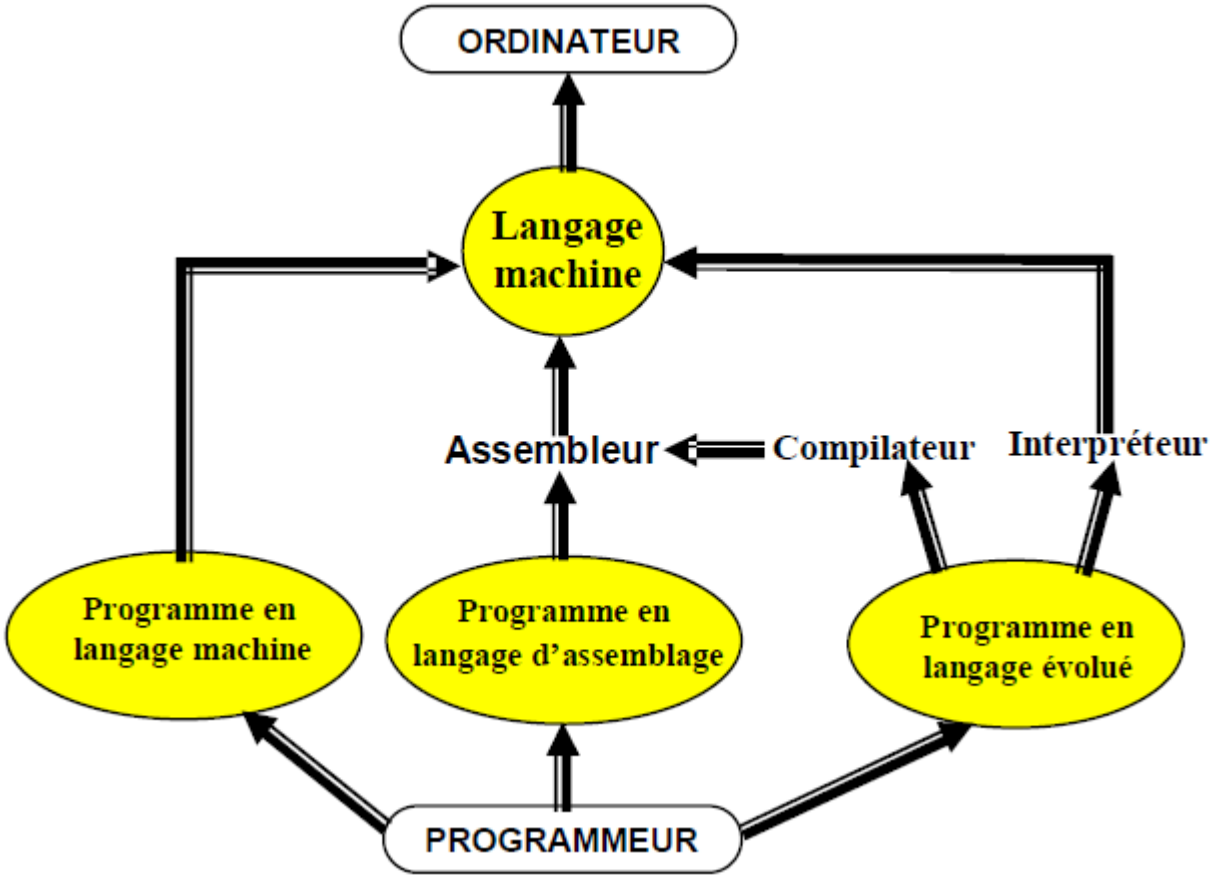
- **L'interpréteur** est un logiciel qui interprète le programme source consiste à traduire chaque ligne du programme source en quelques instructions du langage machine, qui sont ensuite directement exécutées au fur et à mesure. Aucun programme objet n'est généré. L'interpréteur doit être utilisé chaque fois que l'on veut faire fonctionner le programme. Les langages Lisp et Prolog sont des exemples de langages interprétés.

- La **compilation** consiste à traduire la totalité du code source en une seule fois. Le **compilateur** lit toutes les lignes du programme source et produit une nouvelle suite de codes appelé programme objet (ou binaire). Celui-ci peut être exécuté indépendamment du compilateur et être conservé dans un fichier (« fichier exécutable»). Les langages C et C++ sont des exemples de langages compilés.

SUITE

- **Semi-compilation** : Certains langages tentent de combiner les deux techniques afin de retirer le meilleur. C'est le cas des langages **Python** et **Java**. De tels langages commencent par compiler le code source pour produire un code intermédiaire, similaire à un langage machine (mais pour une machine virtuelle), que l'on appelle **bytecode**, lequel sera ensuite transmis à un interpréteur pour l'exécution finale. Pour l'ordinateur, le **bytecode** est très facile à interpréter en langage machine. Cette interprétation sera donc beaucoup plus rapide que celle d'un code source.

SUITE : RÉCAPITULATION



Différents niveaux de langages de programmation

Suite

- Le processus de compilation peut aussi être décomposé : comme le bytecode java qui est un programme *semi-compilé*.

- L'éditeur de liens, reliant le programme compilé avec les librairies précompilées qu'il utilise peut éventuellement faire des optimisations. Le programme peut même être amélioré à l'exécution avant d'être exécuté grâce `à des informations supplémentaires présentes à l'exécution.

Qu'attend-on d'un compilateur ?

- La détection des erreurs : Un compilateur doit pouvoir détecter les erreurs statiques qui ne nécessitent pas l'exécution du programme et être capable de reporter cette erreur à l'utilisateur.
- L'efficacité : Un compilateur doit être rapide.
- La fiabilité : Un compilateur doit être correct.
- La modalité : Lorsque l'on construit un gros développement, il est important de pouvoir profiter de la compilation séparée.

Structure Globale du compilateur

- Il est formé en deux parties :
  - Analyse/reconnaissance
  - Synthèse/transformation

SUITE

- **Partie Avant ou frontale (*front end*) : Analyse**

La phase d'analyse correspond à reconnaitre qu'une entrée est un programme correct du langage source. Cette analyse doit être la plus rapide possible, et elle est formée en trois parties (analyse lexicale, analyse syntaxique, analyse sémantique)

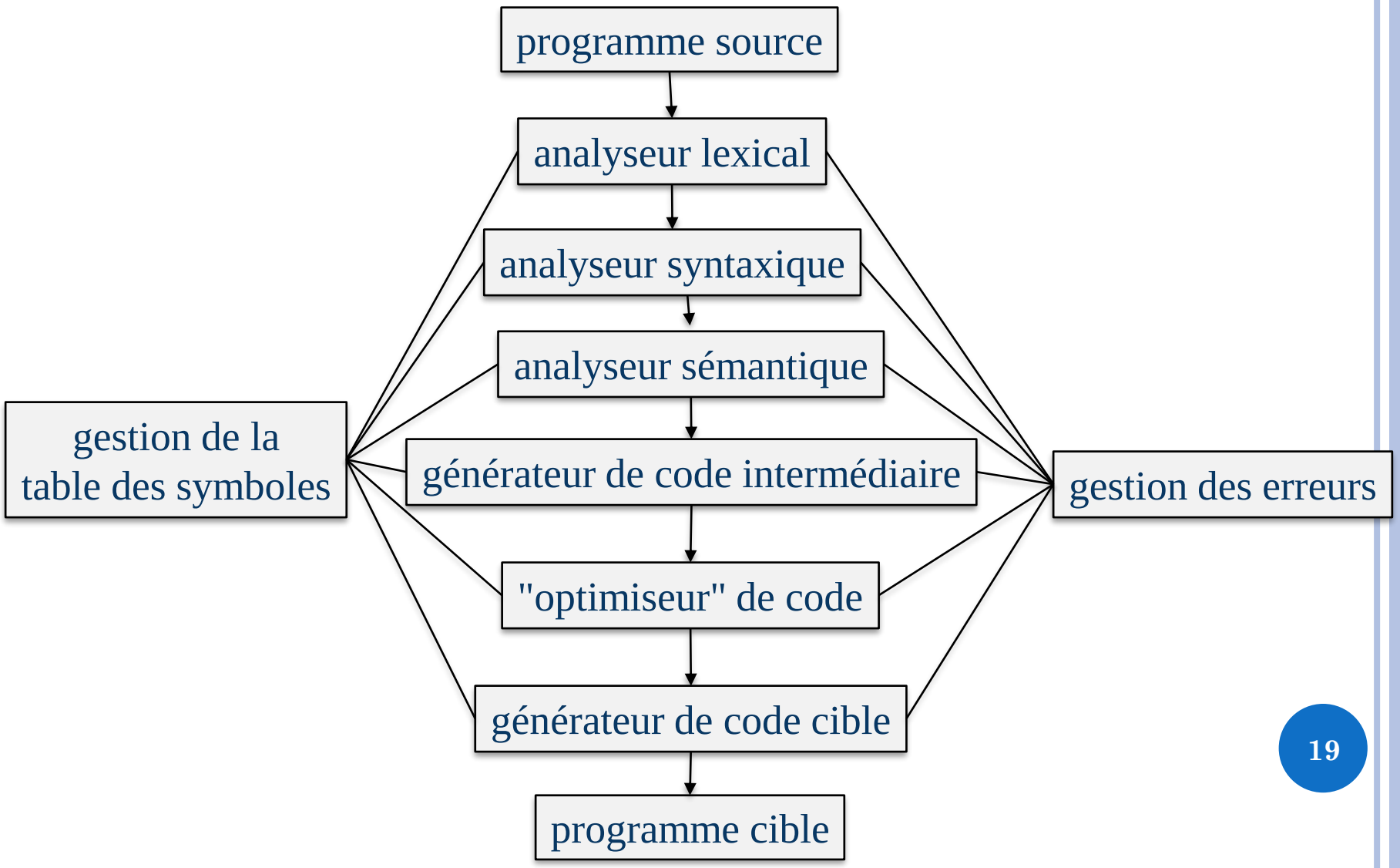
- **Partie arrière (*back end*) : Synthèse**

La synthèse correspond à la phase de reconstruction de l'expression du langage cible à partir de l'arbre syntaxique.

- **Phase parallèle : Gestion de la table des symboles**

On construit pendant toute l'analyse une table des symboles qui associe à chaque identificateur déclaré dans le programme des divers attributs calculées au moment de l'analyse. Ces attributs fournissent des informations concernant, par exemple l'emplacement mémoire assigné à un identificateur, son type, sa portée (c à d la région du programme dans laquelle il est valide)

Les phases de la compilation



Fin