# Chapitre 2 : Compilation

## 2.2 Structure d'un compilateur

Un compilateur est découpé en plusieurs phases. Chaque phase constitue une partie de traduction en elle-même.

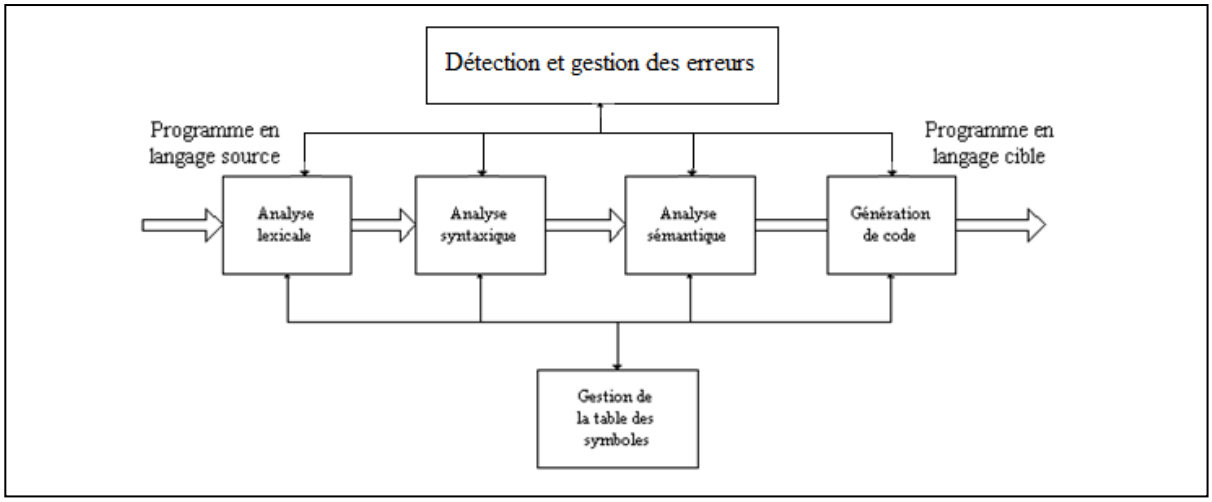


Figure 2.2 : Les différentes phases de compilation

### 2.2.1 Analyse lexicale

Consiste à récupérer les mots, que l'on appelle " **tokens** ", à partir d'une suite de caractères. Par exemple déterminer, à partir de l'énoncé suivant :

```
for i :=1 to vmax do a :=a+i;
```

On peut dégager la suite de tokens suivante :

```
for             : mot clé
i               : identificateur
:=              : affectation
1               : entier
to              : mot clé
vmax            : identificateur
do              : mot clé
a               : identificateur
:=              : affectation
a               : identificateur
+               : opérateur arithmétique
i               : identificateur
;               : séparateur
```

Et que l'on peut construire la table des symboles suivante :

| Numéro de symbole | Token | Type de token | Type de variable |
|---|---|---|---|
| 10 | for | mot clé | |
| 11 | to | mot clé | |
| 12 | do | mot clé | |
| 13 | ; | séparateur | |
| … | … | … | |
| 100 | := | affectation | |
| 101 | + | opérateur arithmétique | |
| … | … | … | |
| 1000 | i | identificateur | |
| 1001 | a | identificateur | |
| 1002 | vmax | identificateur | |
| … | … | … | |
| 5001 | 1 | entier | |
| … | … | … | |

Ensuite, l'énoncé précédent peut s'exprimer ainsi :

```
10, 1000, 100, 5001, 11, 1002, 12, 1001, 100, 1001, 101, 1000, 13
For   i    :=    1   to  vmax  do   a     +     a    +    i     ;
```

C'est-à-dire comme une suite de références à la table des symboles.

- Les blancs séparent les caractères formant ces unités lexicales seront éliminées au cours de l'analyse lexicale.
- Une fonction essentielle d'un compilateur est d'enregistrer les identificateurs utilisés dans le programme source et de collecter de l'information sur divers attributs de chaque identificateur. Ces attributs fournissent de l'information concernant, par exemple, l'emplacement mémoire assigné à un identificateur, son type, sa portée.
- Une table de symboles est une structure de données contenant un enregistrement pour chaque identificateur, muni de champs pour chaque identificateur. Cette structure de données permet de retrouver rapidement l'enregistrement correspondant à un certain identificateur et d'accéder rapidement aux données qu'il contient.
- Quand l'analyseur lexical détecte un identificateur, il le fait entrer dans la table des symboles, cependant ses attributs ne peuvent normalement pas être déterminés pendant l'analyse lexicale ; par exemple, le type ne sera déterminé que pendant l'analyse sémantique, il sera donc ajouté durant cette phase.

### 2.2.2 Analyse syntaxique

Lors de l'analyse syntaxique, on vérifie que l'ordre des tokens correspond à l'ordre définit pour le langage. On dit que l'on vérifie la syntaxe du langage à partir de la définition de sa grammaire. Cette phase produit une représentation sous forme d'arbre de la suite des tokens obtenus lors de la phase précédente. Par exemple, l'arbre suivant représente la structure de la phrase :

```
for i :=1 to vmax do a :=a+i
```

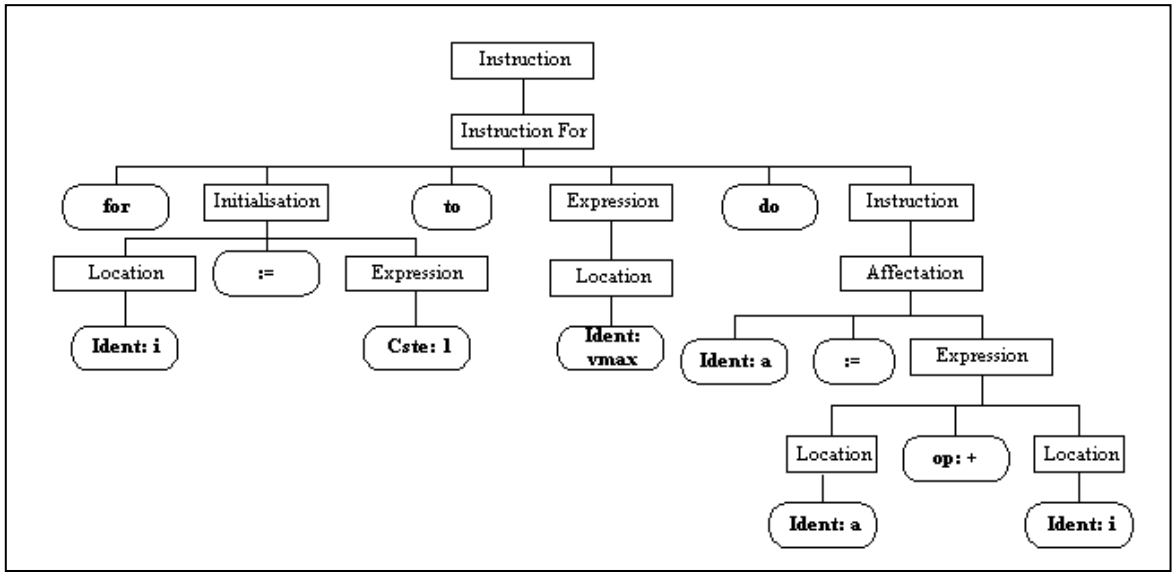


Figure 2.3 : L'arbre syntaxique obtenu après analyse syntaxique

### 2.2.3 Analyse sémantique ou vérification de type

Dans cette phase on vérifie que les variables ont un type correct. Par exemple, il faut vérifier que la variable 'i' possède bien le type 'entier', et que la variable 'a' est bien un nombre. Cette opération s'effectue en parcourant l'arbre syntaxique et en vérifiant à chaque niveau que les opérations sont correctes.

### 2.2.4 Génération de code

On génère le code pour du langage machine ou un langage d'assemblage. Voici un exemple de code généré pour une (pseudo) machine.

```
        var_a                   A0000           ; les étiquettes des
variables
        var_i                   A0001
        var_vmax                A0002

        ...                                     ; le code du programme
        mov var_i,1
loop:
        mov A0, (var_i)                         ; comparaison i >= vmax
        jge A0, (var_vmax), finFor              ; si vrai aller en finFor
        mov A0, (var_a)                         ; calcul de a+i
        add A0, A0, (var_i)
        mov var_a,A0                            ; a := a+i
        mov A0, (var_i)                         ; on incrémente i
        add A0, A0, 1
        mov var_i, 1
        jmp loop                                ; et on continue la boucle
finFor:
        ...
```

### 2.2.5 Détection et gestion des erreurs

Chaque phase peut rencontrer des erreurs. La phase d'analyse lexicale peut détecter des erreurs quand les caractères restant à lire ne peuvent former aucune unité lexicale du langage. Par exemple : l'analyseur lexical peut détecter une erreur de non validité du nom de variable qui commence par un caractère non valide (&x, ày, …).

Peu d'erreurs sont détectées au niveau lexical, car un analyseur lexical à une vison très localisée du programme source. Par exemple : si on rencontre la chaine fi dans un programme avec fi (i = = j) l'analyseur lexical ne peut pas dire qu'il s'agit du mot clé if mal écrit ou d'un identificateur non déclaré.

La phase d'analyse syntaxique détecte les erreurs dues au fait que le flot d'unités lexicales n'est pas conforme aux règles structurelles (syntaxe) du langage.

Dans la phase d'analyse sémantique, le compilateur détecte les constructions ayant une structure grammaticale correcte, mais telle que les opérations qu'elles mettent en œuvre n'ont pas de sens. Par exemple lorsqu'on essaye d'additionner deux identificateurs ; l'un est le nom d'un tableau et l'autre celui d'une procédure.