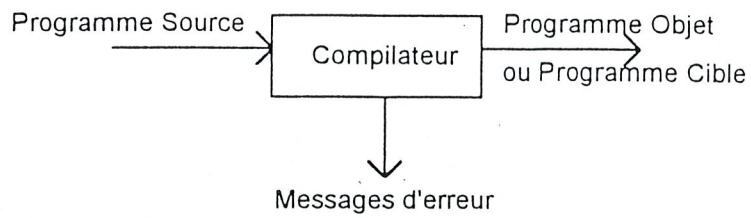


# ÉCRITURE D'UN COMPILEUR ET D'UN INTERPRETEUR



## 1. Le compilateur

Un compilateur est généralement structuré en trois composantes principales:

### ■ L'analyseur

L'analyseur permet de **comprendre** le programme source. Il est organisé en plusieurs parties:

- **l'analyseur lexical:** Il parcourt le programme source, caractère par caractère, élimine les caractères superflus, construit les unités lexicales à partir de ces caractères et les passe à l'analyseur syntaxique;
- **l'analyseur syntaxique:** Il utilise les unités lexicales fournies par l'analyseur lexical, il vérifie que le programme source est syntaxiquement correct, c'est-à-dire que la séquence des unités lexicales dans le programme source respecte les règles de la grammaire du langage;
- **l'analyseur sémantique:** Il vérifie que le programme est sémantiquement correct; par exemple, il analyse la portée des identificateurs et effectue le contrôle des types;

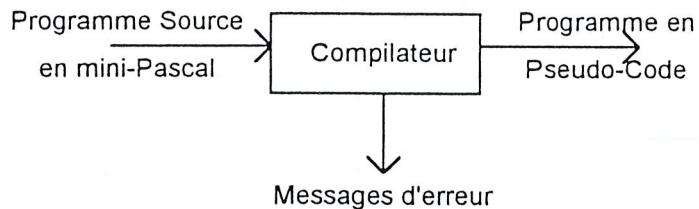
### ■ Le traducteur

La traduction proprement dite est réalisée par le générateur de code. Le code généré peut être du code machine, un code intermédiaire, etc.

### ■ Le gestionnaire d'erreurs

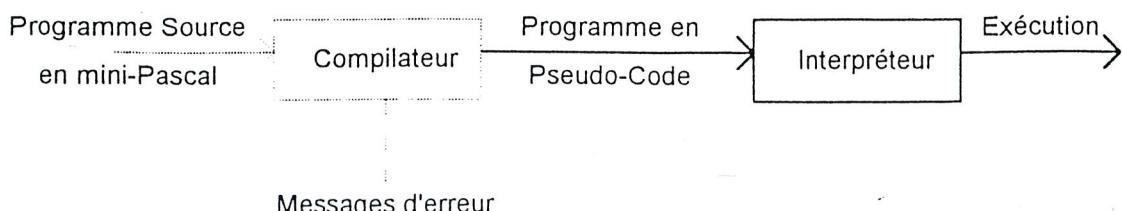
Il détecte les erreurs lexicales, syntaxiques ou sémantiques, pendant l'analyse, et envoie des messages d'erreurs appropriés.

Le compilateur que nous allons écrire comporte les caractéristiques suivantes:



- traduction de programmes écrits dans un langage source qui correspond à un sous-ensemble du langage Pascal. Nous appellerons mini-Pascal ce langage. Les mots-clés du mini-Pascal sont en français.
- compilation en une passe (c'est-à-dire que le fichier source ne sera parcouru qu'une seule fois);
- analyse syntaxique descendante récursive;
- traduction dirigée par la syntaxe;
- l'analyseur sémantique et le générateur de code sont intimement liés à l'algorithme de l'analyseur syntaxique (ce qui est dû à la méthode d'analyse syntaxique et à la méthode de traduction utilisées);
- arrêt de la compilation à la première erreur détectée;
- génération de pseudo-code.

## 2. L'interpréteur



La réalisation d'un interpréteur du pseudo-code généré par le compilateur, permettra finalement d'exécuter les programmes écrits en mini-Pascal.

# 1. ANALYSEUR LEXICAL.

L'objectif est d'écrire un analyseur lexical pour le langage mini-Pascal et de constituer un ensemble de jeux d'essais pour tester cet analyseur lexical.

Les **unités lexicales** du langage source sont:

⇒ **les nombres entiers**

Le langage ne connaît que le type de base entier. La valeur maximale d'un nombre entier est dépendante de la machine sur laquelle le langage est utilisé. Il ne peut dépasser 32767 pour une machine pour laquelle les entiers sont représentés sur 16 bits.

⇒ **les chaînes de caractères**

Une chaîne de caractères est délimitée par le caractère de début de chaîne ' (apostrophe) et le caractère de fin de chaîne ' (apostrophe). Exemple: 'Le résultat est: '

Lorsque la chaîne de caractères contient une apostrophe ', cette apostrophe doit être doublée de façon à être distinguée du délimiteur de début ou de fin de chaîne. Exemple: 'L"indice vaut'.

La longueur d'une chaîne de caractères est limitée à 50 caractères, par exemple.

⇒ **les identificateurs ou mots-clés réservés**

Les identificateurs sont formés d'une lettre suivie d'une suite de lettres ou de chiffres ou de caractères souligné '\_'. Exemples: a123 nombre\_1 Nom

Les mots-clés du mini-Pascal sont formés d'une suite de lettres. Exemples: DEBUT FIN VAR ECRIRE LIRE SI. Ce sont des mots réservés: ils ne peuvent pas être utilisés comme identificateurs.

La longueur maximale d'un identificateur ou mot-clé est fixée, par exemple: 20 caractères.

Le langage ne fait pas de distinction entre les lettres minuscules et les lettres majuscules dans un identificateur ou un mot-clé. Ainsi, index représente le même identificateur que Index ou INDEX dans un programme source.

⇒ **les symboles simples**

Ce sont les symboles formés d'un seul caractère suivants: , ; . : ( ) < > = + - \* /

⇒ **les symboles composés**

Ce sont les symboles formés de deux caractères suivants: <= >= <> :=

**Les séparateurs** sont les caractères qui séparent les mots du langage source. Un séparateur est un espace, une tabulation ou un caractère de fin de ligne.

**Les commentaires** sont délimités par un caractère de début de commentaire et un caractère de fin de commentaire, par exemple: { et }. Un commentaire peut s'étendre sur plusieurs lignes. Les commentaires peuvent être imbriqués. Exemples:

{ Ceci est un commentaire sur une seule ligne }

{ Ceci est { un commentaire imbriqué }

dans un commentaire sur deux lignes }

## ■ Types

- T\_UNILEX, le type énumérant toutes les unités lexicales du langage:

type T\_UNILEX = ( motcle, ident, ent, ch, virg, ptvirg, point, deuxpts, parouv, parfer, inf, sup, eg, plus, moins, mult, divi, infe, supe, diff, aff );

## ■ Constantes

- LONG\_MAX\_IDENT, la longueur maximale d'un identificateur, de valeur 20

- LONG\_MAX\_CHAINE, la longueur maximale d'une chaîne de caractères, de valeur 50

- NB\_MOTS\_RESERVES, le nombre de mots-clés réservés du langage source, de valeur 7

## ■ Variables globales

- SOURCE, le fichier texte contenant le programme source

- CARLU, le dernier caractère lu dans le fichier source

- NOMBRE, le dernier nombre lu

- CHAINE, le dernier identificateur, mot-clé, symbole simple, symbole composé ou chaîne lu

- NUM\_LIGNE, le numéro de la ligne du fichier source courante

- TABLE\_MOTS\_RESERVES, tableau de NB\_MOTS\_RESERVES chaînes de 9 caractères contenant l'ensemble des mots-clés réservés du langage, trié par ordre alphabétique.

## ■ La procédure ERREUR

La procédure ERREUR recense tous les messages d'erreur. A chaque message est associé un numéro d'erreur. La procédure admet un seul argument, le numéro de l'erreur, et affiche le message d'erreur correspondant avec le numéro de la ligne où l'erreur a été détectée (contenu par la variable globale NUM\_LIGNE).

La procédure ERREUR provoque l'arrêt du programme dès que le message d'erreur est affiché.

## ■ La procédure LIRE\_CAR

Le fichier source est lu caractère par caractère jusqu'à la fin du fichier. La procédure LIRE\_CAR lit un caractère dans le fichier source et affecte ce caractère à la variable globale CARLU. De plus, elle mémorise dans la variable globale NUM\_LIGNE, le numéro de la ligne courante du fichier source.

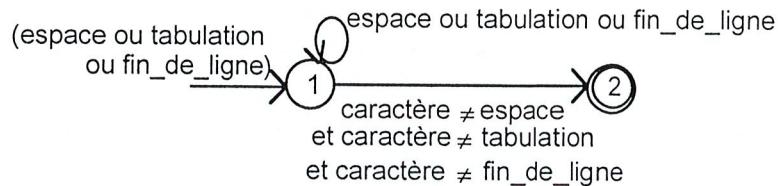
La procédure LIRE\_CAR gère les caractères spéciaux pouvant figurer dans le fichier source tels que:

- la fin de fichier:      Lorsque la fin de fichier est atteinte, la procédure LIRE\_CAR appelle la procédure d'erreur avec le paramètre effectif 1 (erreur n°1, message "fin de fichier atteinte");
- la fin de ligne:      Chaque fois que le caractère lu est le caractère de fin de ligne, il faut incrémenter la variable globale NUM\_LIGNE.

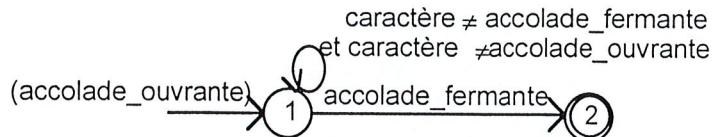
### ■ La procédure SAUTER\_SEPARATEURS

Lorsque cette procédure est appelée, CARLU contient un séparateur ou le caractère de début d'un commentaire. La procédure lit des caractères (en appelant la procédure LIRE\_CAR) dans le fichier source jusqu'à ce que le caractère lu CARLU ne soit pas à l'intérieur d'un commentaire et ne soit pas un séparateur. Elle lit ainsi les séparateurs ou commentaires consécutifs. En particulier, lorsque CARLU est le caractère de début de commentaire, elle doit lire tous les caractères jusqu'au caractère de fin de commentaire, en tenant compte du fait que des commentaires peuvent être imbriqués.

Pour les séparateurs:

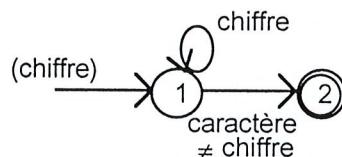


Pour des commentaires non imbriqués:



### ■ La fonction RECO\_ENTIER

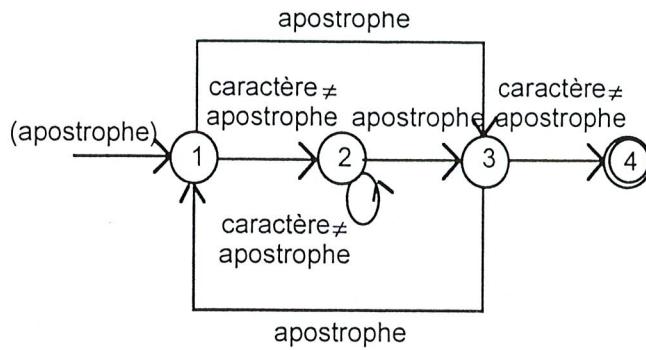
Cette fonction reconnaît les nombres entiers. Elle est appelée lorsque CARLU contient un chiffre. La fonction lit les chiffres consécutifs dans le fichier source jusqu'à ce que le caractère lu CARLU ne soit pas un chiffre et affecte le nombre entier correspondant à cette suite de chiffres à la variable globale NOMBRE. La fonction renvoie l'unité lexicale reconnue: ent.



De plus, cette fonction doit veiller à ce que le nombre entier obtenu ne dépasse pas la capacité de représentation des nombres entiers sur la machine cible. Dans notre cas, un nombre entier ne peut dépasser la valeur MAXINT (constante prédéfinie en PASCAL). En cas de dépassement de valeur, la fonction appelle la procédure d'erreur, de façon à signaler cette erreur à l'utilisateur.

### ■ La fonction RECO\_CHAINE

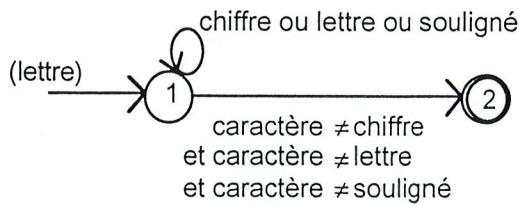
Cette fonction reconnaît les chaînes de caractères délimitées par des apostrophes. Elle est appelée lorsque CARLU contient une apostrophe. La fonction lit des caractères dans le fichier source jusqu'à ce que le caractère lu CARLU indique la fin de la chaîne (une apostrophe). La difficulté est que la chaîne peut elle-même contenir des apostrophes. La fonction affecte la chaîne de caractères construite à partir de la suite de caractères lus entre l'apostrophe de début de la chaîne et l'apostrophe de la fin de la chaîne, à la variable globale CHAINE. Elle renvoie l'unité lexicale reconnue: ch.



Lorsque la chaîne reconnue dépasse LONG\_MAX\_CHAINE caractères, la fonction génère une erreur.

### ■ La fonction RECO\_IDENT\_OU\_MOT\_RESERVE

Cette fonction reconnaît les identificateurs ou mots-clés réservés du programme source. Elle est appelée lorsque CARLU contient une lettre. La fonction lit les chiffres, lettres ou soulignés consécutifs dans le fichier source jusqu'à ce que le caractère lu CARLU ne soit ni un chiffre, ni une lettre, ni le caractère souligné. Elle affecte à la variable globale CHAINE, la chaîne de caractères formée par cette suite de lettres, chiffres et soulignés.



Lorsque l'identificateur ou le mot-clé reconnu dépasse LONG\_MAX\_IDENT caractères, la fonction ne génère pas d'erreur mais ignore les caractères superflus.

Comme le langage ne distingue pas les lettres minuscules et les lettres majuscules dans les identificateurs et les mots-clés, la fonction transforme toutes les lettres en lettres majuscules dans la variable CHAINE.

La fonction RECO\_IDENT\_OU\_MOT\_RESERVE renvoie l'unité lexicale reconnue: ident ou motclé. Par conséquent, une fois que la chaîne de caractères a été reconstituée (dans la variable globale CHAINE), la fonction doit encore déterminer s'il s'agit d'un identificateur ou d'un mot-clé réservé. Pour cela, elle utilise la fonction booléenne EST\_UN\_MOT\_RESERVE qui teste si la chaîne de caractères CHAINE est dans la table TABLE\_MOTS\_RESERVES. La fonction booléenne EST\_UN\_MOT\_RESERVE peut être déclarée comme une fonction locale à la fonction RECO\_IDENT. Elle n'a pas d'argument et renvoie TRUE si la chaîne CHAINE se trouve dans le tableau TABLE\_MOTS\_RESERVES, FALSE sinon. La recherche de CHAINE dans le tableau TABLE\_MOTS\_RESERVES devra utiliser le fait que le tableau est trié dans l'ordre alphabétique: elle pourra être séquentielle ou dichotomique.

#### algorithme de recherche séquentielle

```

i := 1;
while ( TABLE_MOTS_RESERVES[i] < CHAINE )
      and ( i < NB_MOTS_RESERVES ) do
        i := i+1;
EST_UN_MOT_RESERVE := CHAINE =
        TABLE_MOTS_RESERVES[i];

```

#### algorithme de recherche dichotomique

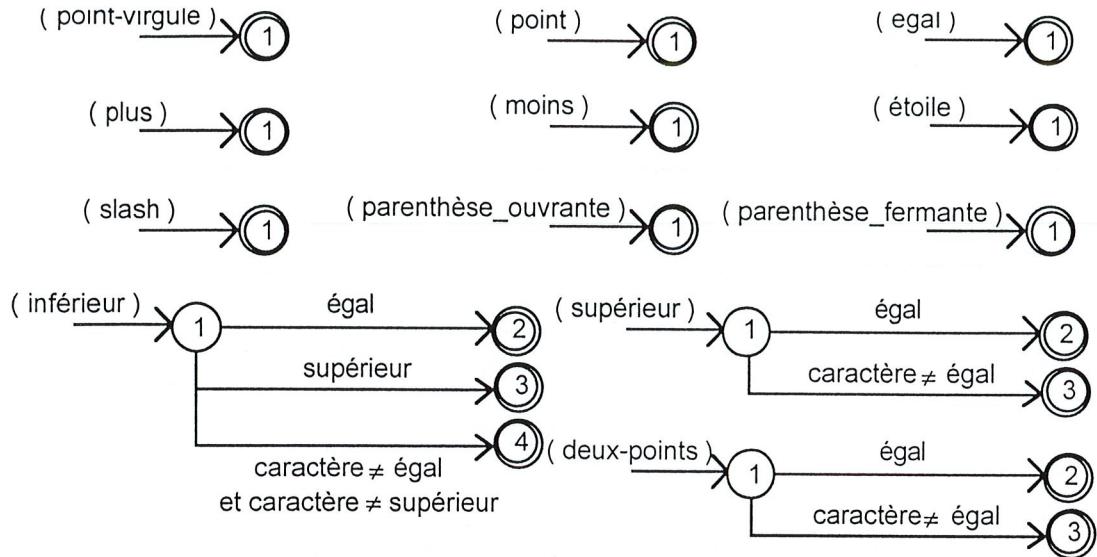
```

i := 1;
j := NB_MOTS_RESERVES;
repeat
  k := (i+j) div 2;
  if CHAINE > TABLE_MOTS_RESERVES[k]
    then i := k+1
    else j:= k-1
  until ( CHAINE = TABLE_MOTS_RESERVES[k] ) or (i > j);
EST_UN_MOT_RESERVE := CHAINE =
  TABLE_MOTS_RESERVES[k];

```

### ■ La fonction RECO\_SYMB

Cette fonction reconnaît les symboles simples et les symboles composés. Elle est appelée lorsque CARLU contient un symbole simple ou le premier caractère d'un symbole composé.



La fonction renvoie l'unité lexicale correspondant au symbole reconnu: ptvrg, point, parouv, parfer, inf, sup, eg, plus, moins, mult, divi, infe, supe, diff ou aff.

Remarque importante: L'analyseur lexical doit toujours être un caractère en avance sur la dernière unité lexicale lue. Par conséquent, à la fin de l'exécution de la fonction RECO\_SYMB, CARLU doit contenir le caractère dans le fichier source qui suit le symbole reconnu.

### ■ La fonction ANALEX

Cette fonction est le noyau même de l'analyseur lexical. Elle doit reconnaître l'unité lexicale suivante dans le fichier source et la passer à l'analyseur syntaxique.

Elle est appelée lorsque CARLU contient:

- soit le premier caractère du mot correspondant à l'unité lexicale à reconnaître,
- soit un séparateur ou le caractère de début de commentaire.

La fonction saute d'abord les séparateurs et les commentaires éventuels, elle essaie ensuite de reconnaître une unité lexicale.

Les unités lexicales que nous avons identifiées commencent toutes par un caractère différent. Il est facile de déduire de tous les automates déterministes associés aux unités lexicales, un automate déterministe pour la reconnaissance des différentes unités lexicales. La fonction ANALEX simule le comportement de cet automate. Elle utilise les fonctions précédentes RECO\_ENT, RECO\_CHAINE, RECO\_IDENT\_OU\_MOT\_RESERVE ET RECO\_SYMB.

### ■ La procédure INITIALISER

La procédure INITIALISER effectue les initialisations de début du programme. Son rôle, lorsque le programme est réduit à l'analyse lexicale, consiste à:

- initialiser la variable globale NUM\_LIGNE,
- ouvrir le fichier SOURCE en lecture,
- initialiser le tableau des mots-clés réservés TABLE\_MOTS\_RESERVES. Les mots-clés utilisés dans un premier temps sont: PROGRAMME, DEBUT, FIN, CONST, VAR, ÉCRIRE, LIRE. Ils peuvent être placés dans l'ordre alphabétique à l'aide d'une procédure locale INSERE\_TABLE\_MOTS\_RESERVES, qui admet un seul argument, la chaîne de caractères à insérer.

### ■ La procédure TERMINER

La procédure TERMINER effectue les instructions de fin du programme. Son rôle, lorsque le programme est réduit à l'analyse lexicale, consiste à fermer le fichier SOURCE.

### ■ Le programme principal

Le programme principal de l'analyseur lexical doit:

- appeler la procédure INITIALISER,
- reconnaître toutes les unités lexicales du fichier SOURCE et les afficher,
- appeler la procédure TERMINER.

L'analyseur lexical est toujours un caractère en avance sur l'unité lexicale reconnue. Cette façon de procéder est la plus simple. Il est donc essentiel d'appeler la fonction LIRE\_CAR avant le tout premier appel de la procédure ANALEX. Les unités lexicales reconnues par l'analyseur lexical seront utilisées par l'analyseur syntaxique. L'affichage à l'écran de chaque unité lexicale permet, dans un premier temps, de tester et valider l'analyseur lexical.