Module : Compilation

Niveau : 3ème Année LMD SI

Enseignant : MAHDJOUBI Roussafi

TP : réalisation d'un mini analyseur lexical

Le but de ce TP est de faire apprendre à l'étudiant la notion d'analyse lexicale du processus de compilation en réalisant un programme en langage C qui permet d'obtenir les différentes entités lexicales composant un texte quelconque. L'écriture du programme se fait d'une façon graduelle en termes de complexité par l'introduction au fur et à mesure de nouvelles notions requises.

1ère étape : Il s'agit d'écrire un programme qui, étant donné une chaîne de caractères quelconque, permet de la décomposer en une suite d'entités lexicales en considérant comme seul séparateur, le caractère blanc. Les entités seront affichées les unes au-dessus des autres. On utilisera pour cela une variable qu'on appellera « Token » pour contenir à chaque passage, l'entité reconnue.

Ex : soit la chaîne **si i+j>=2 alors x :=1**;

On obtient :

si

i+j>=2

alors

x :=1;

2ème étape : Une fois le programme réalisé, on étend l'ensemble des séparateurs en ajoutant d'autres symboles tels que les opérateurs. On commence par introduire les symboles **(+,*,=, >,<, :,;**) puis on laisse libre cours à l'étudiant d'ajouter d'autres symboles à son choix. On remarque ici, que ces symboles sont reconnus à part malgré l'absence du caractère blanc : ils sont considérés comme séparateurs et entités en même temps.

Pour notre exemple, on obtient :

si

i

+

J

>

=

2

alors

x

:

=

1

;

3ème étape : On introduit les opérateurs composés de plus d'un symbole (>=,<=,<> , :=). Notre programme devrait être capable de reconnaître les opérateurs à deux symboles de ceux composés d'un seul. On obtiendrait alors dans notre exemple :

si

i

+

J

>=

2

alors

x

:=

1

;

4ème étape : Une fois terminé, on transforme le programme obtenu en une procédure ou fonction qu'on appellera « Next_Token » qui sera appelée dans un programme principal. Cette procédure devrait fournir une seule entité à chaque appel donc, on doit l'intégrer dans une boucle qui permet d'obtenir toutes les entités de la chaîne introduite.

```
Fonction Next_Token() : Chaine de caractères ;
Debut
        …….
Fin ;
……………….
Debut
        ……………….
         Tant que non fin_de_chaine faire
         Debut
            Next_Token ;
            Afficher (Token) ;
         Fin ;
         ……..
Fin ;
```

5ème étape : Ici, on introduit la notion de nombre pour distinguer les nombres des identificateurs. On doit considérer toute suite de chiffres en tant que nombre tant qu'elle n'a pas une lettre à son début. Ex :

A253 est un identificateur car il commence par une lettre

253 est un nombre.

253abc21 est considéré comme deux entités : un nombre(253) suivi d'un identificateur (abc21)

6ème étape : Pour préparer cette procédure à travailler avec l'analyseur syntaxique, on ajoutera alors, la possibilité de reconnaître la nature de chaque entité et de l'afficher à son côté : identificateur(Ident) , Operateur(Op), mot réservé (KeyWord), Nombre(Nbr), Ponctuation(Pnct),…

Si	**KeyWord**
I	**Ident**
+	**Op**
J	**Ident**
>=	**Op**
2	**Nbr**
Alors	**KeyWord**
x	**Ident**
:=	**Op**
1	**Nbr**
;	**Pnct**