

## 1- Alphabet du langage :

Alphabet= {a, ..., z, A, ..., Z, 0, ..., 9, , , ; , = , < , > , ! , + , - , \* , / , | , & , { , } , [ , ] , ( , ) , # }

2- Mots clés :

Mot clé= {main ,entier, car, si, alors, sinon, tantque, faire, écrire, lire, retour}

### 3- Grammaire et Intégration de l'option 1 :

L'option 1 consiste à intégrer dans la grammaire le type caractère.

Après l'intégration de cette structure, nous avons obtenu la grammaire suivante :

1 <programme> ::= <liste de déclarations> <liste de fonctions>

$$2 \text{ <liste de déclarations> } ::= \varepsilon \mid \text{<déclaration>}, \text{<liste de déclarations>} ;$$

3 <déclaration> ::= entier <identificateur> | Car <identificateur>  
| entier <identificateur> [<cste>]  
| Car <identificateur> [<cste>]

4 <liste de fonctions> ::=  $\varepsilon$  | <déclaration fonction> <liste d'instructions fonctions> <liste de fonctions>

5 <déclaration fonction> ::= <identificateur> (<liste de paramètres>)

$$6 \text{ <liste de paramètres> } ::= \varepsilon \mid \text{<paramètre>}, \text{<liste de paramètres>}$$

7 <paramètre> =:: entier <identificateur> | Car <identificateur>

8 <liste d'instructions fonctions> ::= ε | <liste de déclarations> { <liste d'instructions> } <liste d'instructions fonctions>

9  $\langle \text{liste d'instructions} \rangle ::= \varepsilon \mid \langle \text{instruction} \rangle ; \langle \text{liste d'instructions} \rangle$

10 <instruction> ::= <identificateur>=<expression>  
                           | <identificateur>[<expression simple>] =<expression>  
                           | retour <expression>  
                           | si <expression> alors { <instruction> } sinon { <instruction> }  
                           | si <expression> alors { <instruction> }  
                           | tantque <expression> faire { <instruction> }  
                           | ecrire( <liste d'expressions> )  
                           | <identificateur> = lire()

11 <expression> ::= <expression simple> <comparaison> <expression simple>  
                           | <expression simple>  
                           | <identificateur>(<liste d'expressions>)

12 <liste d'expressions> ::= ε | <expression> , <liste d'expressions>

13 <expression simple> ::= <expression simple>+<terme>  
                                   | <expression simple>-<terme>  
                                   | <expression simple>|<terme>  
                                   | <terme>  
                                   | -<terme>

14 <terme> ::= <terme>\*<facteur>  
                   | <terme>/<facteur>  
                   | <terme>&&<facteur>  
                   | !<facteur>  
                   | <facteur>

15 <facteur> ::= <identificateur>  
                   | <cste>  
                   | (<expression simple>)  
                   | <identificateur>[<expression simple>]  
                   | <lettre>

16 <comparaison> ::= <|>| == | <= | >= | !=

17 <identificateur> ::= <lettre> <mot>

18 <mot> = :: ε | <lettre><mot> | <chiffre><mot>

19 <cste> = :: <chiffre> | <chiffre><cste>

20 <chiffre> = :: 0|1|...|8|9

21 <lettre> = :: A|B|...|Z|a|b|...|z

## 4- Grammaire LL(1)

Dans cette partie, on va d'abord éliminer la récursivité { gauche de la grammaire, puis la factoriser et après vérifier qu'elle est une grammaire LL(1).

- Eliminer la récursivité à gauche de la grammaire :

Dans notre grammaire, la récursivité est éliminée sur la majorité des règles sauf :

```
13 <expression simple> ::= <expression simple>+<terme>
                        | <expression simple>-<terme>
                        | <expression simple>|<terme>
                        | <terme>
                        | -<terme>

14 <terme> ::= <terme>*<facteur>
              | <terme>/<facteur>
              | <terme>&&<facteur>
              | !<facteur>
              | <facteur>
```

Alors en appliquant les règles d'élimination de la récursivité à gauche on trouve :

```
13 <expression simple> ::= <terme><expression simple '>' | -<terme><expression simple '>'

13' <expression simple '>' ::= +<terme><expression simple '>'
                             | -<terme><expression simple '>'
                             | ||<terme><expression simple '>'
                             | ε

14 <terme> ::= !<facteur><terme '>' | <facteur><terme '>'

14' <terme '>' ::= *<facteur><terme '>'
                 | /<facteur><terme '>'
                 | &&<facteur><terme '>'
                 | ε
```

- Factorisation de la grammaire :

Dans notre grammaire, on a des problèmes de factorisation sur les règles suivantes :

```

3 <déclaration> ::= entier <identificateur> | Car <identificateur>
                  | entier <identificateur> [<cste>]
                  | Car <identificateur> [<cste>]

10 <instruction> ::= <identificateur>=<expression>
                  | <identificateur>[<expression simple>] =<expression>
                  | retour <expression>
                  | si <expression> alors { <instruction> } sinon { <instruction> }
                  | si <expression> alors { <instruction> }
                  | tantque <expression> faire { <instruction> }
                  | ecrire( <liste d'expressions> )
                  | <identificateur> = lire()

11 <expression> ::= <expression simple> <comparaison> <expression simple>
                  | <expression simple>
                  | <identificateur>(<liste d'expressions>)

15 <facteur> ::= <identificateur>
                | <cste>
                | (<expression simple>)
                | <identificateur>[<expression simple>]
                | <lettre>

19 <cste> ::= <chiffre> | <chiffre><cste>

```

Alors en appliquant les règles de factorisation à gauche on trouve :

```

3 <déclaration> ::= entier <déclaration '>' | Car <déclaration '>'

3'<déclaration '> ::= <identificateur><déclaration ">

3''<déclaration "> ::= ε | [<cste>]

10 <instruction> ::= <identificateur><instruction '>
                  | retour <expression>
                  | si <expression> alors { <instruction> } <instruction ">

```

| tantque <expression> faire { <instruction> }  
| ecrire( <liste d'expressions> )

10' <instruction ' > ::=                =lire()  
   | [<expression simple>] =<expression>  
   | =<expression>

10'' <instruction " > ::=  sinon { <instruction> }   |  ε

11 <expression> ::=        <expression simple> <expression ' >  
                                 | <identificateur>(<liste d'expressions>)

11' <expression ' > ::= <comparaison> <expression simple> | ε

15 <facteur> ::=            <identificateur><facteur ' >  
                                 | <cste>  
                                 | (<expression simple>)  
                                 | <lettre>

15' <facteur ' > ::= ε | [<expression simple>]

19 <cste> ::= <chiffre><cste ' >

19' <cste ' > ::= ε | <cste>

- Grammaire Après Changement :

1 <programme> ::= <liste de déclarations> <liste de fonctions>

2 <liste de déclarations> ::= ε | <déclaration> , <liste de déclarations> ;

3 <déclaration> ::= entier <déclaration ' > | Car <déclaration ' >

3' <déclaration ' > ::= <identificateur><déclaration " >

3'' <déclaration " > ::= ε | [<cste>]

4 <liste de fonctions> ::= ε | <déclaration fonction> <liste d'instructions fonctions> <liste de fonctions>

5 <déclaration fonction> ::= <identificateur> (<liste de paramètres>)

6 <liste de paramètres> ::= ε | <paramètre> , <liste de paramètres>

7 <paramètre> ::= entier <identificateur> | Car <identificateur>

8 <liste d'instructions fonctions> ::=  $\epsilon$  | <liste de déclarations> { <liste d'instructions> } <liste d'instructions fonctions>

9 <liste d'instructions> ::=  $\epsilon$  | <instruction> ; <liste d'instructions>

10 <instruction> ::=  
    <identificateur><instruction ' >  
    | retour <expression>  
    | si <expression> alors { <instruction> } <instruction " >  
    | tantque <expression> faire { <instruction> }  
    | ecrire( <liste d'expressions> )

10' <instruction ' > ::= lire()  
    | [<expression simple>] =<expression>  
    | =<expression>

10'' <instruction " > ::= sinon { <instruction> } |  $\epsilon$

11 <expression> ::=  
    <expression simple> <expression ' >  
    | <identificateur>(<liste d'expressions>)

11' <expression ' > ::= <comparaison> <expression simple> |  $\epsilon$

12 <liste d'expressions> ::=  $\epsilon$  | <expression> , <liste d'expressions>

13 <expression simple> ::= <terme><expression simple ' > | -<terme><expression simple ' >

13' <expression simple ' > ::=  
    +<terme><expression simple ' >  
    | -<terme><expression simple ' >  
    | ||<terme><expression simple ' >  
    |  $\epsilon$

14 <terme> ::= !<facteur><terme ' > | <facteur><terme ' >

14' <terme ' > ::=  
    \*<facteur><terme ' >  
    | /<facteur><terme ' >  
    | &&<facteur><terme ' >  
    |  $\epsilon$

15 <facteur> ::=  
    <identificateur><facteur ' >  
    | <cste>  
    | (<expression simple>)  
    | <lettre>

15' <facteur ' > ::=  $\epsilon$  | [<expression simple>]

16 <comparaison> ::= <|> | == | <=> | >= | !=

17 <identificateur> ::= <lettre> <mot>

18  $\langle \text{mot} \rangle = :: \varepsilon \mid \langle \text{lettre} \rangle \langle \text{mot} \rangle \mid \langle \text{chiffre} \rangle \langle \text{mot} \rangle$

19  $\langle \text{cste} \rangle = :: \langle \text{chiffre} \rangle \langle \text{cste} \rangle ' >$

19'  $\langle \text{cste} \rangle ' > = :: \varepsilon \mid \langle \text{cste} \rangle$

20  $\langle \text{chiffre} \rangle = :: 0 \mid 1 \mid \dots \mid 8 \mid 9$

21  $\langle \text{lettre} \rangle = :: A \mid B \mid \dots \mid Z \mid a \mid b \mid \dots \mid z$

- **Vérification de la grammaire:**

On teste si la grammaire est LL(1) ou non par la propriété suivante :

Propriété :

Une grammaire est LL(1) si pour tout non-terminal X apparaissant dans le membre gauche de deux productions :

$X \rightarrow a$  ,  $X \rightarrow b$

Alors :

- 1-  $\text{Premier}(a) \cap \text{Premier}(b)$  égale à l'ensemble vide.
- 2- Une des conditions suivante est vraie :
  - Ni a ni b n'est annulable et aucune ne se dérive en epsilon.
  - Uniquement, a ou bien b est annulable et  $\text{Premier}(X) \cap \text{Suivant}(X)$  égale à l'ensemble vide.

Premièrement, on calculera les Premiers et Les Suivants de tous les productions de la grammaire

Table d'analyse :

Non terminal	Premier	Suivant
--------------	---------	---------

<b>&lt;programme&gt;</b>	Entier, car , $\varepsilon$	\$
<b>&lt;liste de déclarations &gt;</b>	Entier, car , $\varepsilon$	Premier(liste de fonctions)→ lettre, \$
<b>&lt;déclaration &gt;</b>	entier, car	,
<b>&lt;déclaration '&gt;</b>	Lettre, $\varepsilon$	,
<b>&lt;déclaration "&gt;</b>	[, $\varepsilon$	,
<b>&lt;liste de fonctions &gt;</b>	lettre , $\varepsilon$	\$
<b>&lt;déclaration fonction&gt;</b>	lettre	Premier(liste d'instructions fonctions)→ entier, car , )
<b>&lt;liste de paramètres&gt;</b>	Entier, car , $\varepsilon$	)
<b>&lt;paramètre &gt;</b>	Entier, car	,
<b>&lt;liste d'instructions fonctions &gt;</b>	entier, car , $\varepsilon$	Premier(liste de fonctions)→lettre, \$
<b>&lt;liste d'instructions&gt;</b>	Retour, si, tantque, ecrire, $\varepsilon$	}



<b>&lt;instruction&gt;</b>	Retour, si, tantque, ecrire	, ;
<b>&lt;instruction '&gt;</b>	=, [	, ;
<b>&lt;instruction '&gt;</b>	sinon, ε	, ;
<b>&lt;expression&gt;</b>	Lettre, chiffre, (, !, -	, ,alors, faire, }, ;
<b>&lt;expression '&gt;</b>	<, >, ==, <=, >=, !=, , ε	, ,alors, faire, }, ;
<b>&lt;liste d'expressions&gt;</b>	<, >, ==, <=, >=, !=, , ε	)
<b>&lt;expression simple&gt;</b>	Lettre, chiffre, (, !, -	), ], Premier(expression') → <, >, ==, <=, >=, !=, , ,alors, faire, }, ;
<b>&lt;expression simple '&gt;</b>	+, -,   , , ε	), ], Premier(expression') → <, >, ==, <=, >=, !=, , ,alors, faire, }, ;
<b>&lt;terme&gt;</b>	!, Lettre, chiffre, (	Premier(expression simple') → +, -,   , , ), ], <, >, ==, <=, >=, !=, , ,alors, faire, }, ;
<b>&lt;terme '&gt;</b>	*, /, &&, ε	Premier(expression simple') → +, -,   , , ), ], <, >, ==, <=, >=, !=, , ,alors, faire, }, ;
<b>&lt;facteur&gt;</b>	Lettre, Chiffre, (	Premier(terme') → *, /, &&, +, -,   , , ), ], <, >, ==, <=, >=, !=, , ,alors, faire, }, ;
<b>&lt;facteur '&gt;</b>	[, ε	Premier(terme') → *, /, &&, +, -,   , , ), ], <, >, ==, <=, >=, !=, , ,alors, faire, }, ;

<b>&lt;comparaison&gt;</b>	<, >, ==, <=, >=, !=	Premier(expression simple) → Lettre, chiffre, (, !, -
<b>&lt;cste&gt;</b>	chiffre	], Premier(terme') → +, -,   , , ), ], <, >, ==, <=, >=, !=, , , alors, faire, }, ;
<b>&lt;cste '&gt;</b>	chiffre , ε	], Premier(terme') → +, -,    , , ), ], <, >, ==, <=, >=, != , , , alors, faire, }, ;
<b>&lt;identificateur&gt;</b>	lettre	, , (, Premier(instruction') → =, [, Premier(facteur') → *, /, && , +, -,    , , ), ], <, >, ==, <=, >=, != , , , alors, faire, }, ;
<b>&lt;mot&gt;</b>	Lettre, chiffre , ε	, , (, Premier(instruction') → =, [, Premier(facteur') → *, /, && , +, -,    , , ), ], <, >, ==, <=, >=, != , , , alors, faire, }, ;

### Vérification des productions :

- 1 <programme> ::= <liste de déclarations> <liste de fonctions> → La production est LL(1)
- 2 <liste de déclarations> ::= ε | <déclaration> , <liste de déclarations> ; → La production est LL(1) ( $\text{Pr}(\text{déclaration}) \wedge \text{Pr}(\varepsilon) = \Phi$  et  $\text{Pr} \wedge \text{Sv} = \Phi$ )
- 3 <déclaration> ::= entier <déclaration '> | Car <déclaration '> → La production est LL(1)
- 3' <déclaration '> ::= <identificateur> <déclaration "> → La production est LL(1)
- 3'' <déclaration "> ::= ε | [<cste>] → La production est LL(1) ( $\text{Pr} \wedge \text{Sv} = \Phi$ )
- 4 <liste de fonctions> ::= ε | <déclaration fonction> <liste d'instructions fonctions> <liste de fonctions> → La production est LL(1) ( $\text{Pr}(\text{déclaration fonction}) \wedge \text{Pr}(\varepsilon) = \Phi$  et  $\text{Pr} \wedge \text{Sv} = \Phi$ )
- 5 <déclaration fonction> ::= <identificateur> (<liste de paramètres>) → La production est LL(1)
- 6 <liste de paramètres> ::= ε | <paramètre> , <liste de paramètres> → La production est LL(1) ( $\text{Pr}(\text{paramètre}) \wedge \text{Pr}(\varepsilon) = \Phi$  et  $\text{Pr} \wedge \text{Sv} = \Phi$ )
- 7 <paramètre> ::= entier <identificateur> | Car <identificateur> → La production est LL(1)

8 <liste d'instructions fonctions> ::=  $\epsilon$  | <liste de déclarations> { <liste d'instructions> } <liste d'instructions fonctions>

9 <liste d'instructions> ::=  $\epsilon$  | <instruction> ; <liste d'instructions> → **La production est LL(1) ( $\text{Pr}(\text{instruction}) \wedge \text{Pr}(\epsilon) = \Phi$  et  $\text{Pr} \wedge \text{Sv} = \Phi$ )**

10 <instruction> ::=  
    <identificateur><instructions ' >  
    | retour <expression>  
    | si <expression> alors { <instruction> } <instruction " >  
    | tantque <expression> faire { <instruction> }  
    | ecrire( <liste d'expressions> )

→ **La production est LL(1) ( $\wedge \text{Pr} = \Phi$ )**

10' <instruction ' > ::= lire()  
    | [<expression simple>] =<expression>  
    | =<expression>

→ **La production est LL(1)**

10'' <instruction " > ::= sinon { <instruction> } |  $\epsilon$  → **La production est LL(1) ( $\text{Pr} \wedge \text{Sv} = \Phi$ )**

11 <expression> ::=  
    <expression simple> <expression ' >  
    | <identificateur>(<liste d'expressions>)

11' <expression ' > ::= <comparaison> <expression simple> |  $\epsilon$  → **La production est LL(1)**

12 <liste d'expressions> ::=  $\epsilon$  | <expression> , <liste d'expressions> → **La production est LL(1) ( $\text{Pr}(\text{expression}) \wedge \text{Pr}(\epsilon) = \Phi$  et  $\text{Pr} \wedge \text{Sv} = \Phi$ )**

13 <expression simple> ::= <terme><expression simple ' > | -<terme><expression simple ' > → **La production est LL(1) ( $\text{Pr}(\text{terme}) \wedge \text{Pr}(-) = \Phi$ )**

13' <expression simple ' > ::=  
    +<terme><expression simple ' >  
    | -<terme><expression simple ' >  
    | ||<terme><expression simple ' >  
    |  $\epsilon$

→ **La production est LL(1) ( $\text{Pr} \wedge \text{Sv} = \Phi$ )**

14 <terme> ::= !<facteur><terme ' > | <facteur><terme ' > → **La production est LL(1) ( $\text{Pr}(\text{facteur}) \wedge \text{Pr}(!) = \Phi$ )**

14' <terme ' > ::=  
    \*<facteur><terme ' >  
    | /<facteur><terme ' >  
    | &&<facteur><terme ' >  
    |  $\epsilon$

→ **La production est LL(1) ( $\text{Pr} \wedge \text{Sv} = \Phi$ )**

15 <facteur> =::  
    <identificateur><facteur ' >  
    | <cste>  
    | (<expression simple>)  
    | <lettre>

15' <facteur ' > =::  $\varepsilon$  | [<expression simple>] ➔ La production est LL(1) ( $Pr \wedge Sv = \Phi$ )

16 <comparaison> =:: <|> | == | <= | >= | != ➔ La production est LL(1)

17 <identificateur> =:: <lettre> <mot> ➔ La production est LL(1)

18 <mot> =::  $\varepsilon$  | <lettre><mot> | <chiffre><mot> ➔ La production est LL(1) ( $Pr \wedge Sv = \Phi$ )

19 <cste> =:: <chiffre><cste ' > ➔ La production est LL(1)

19' <cste ' > =::  $\varepsilon$  | <cste> ➔ La production est LL(1) ( $Pr \wedge Sv = \Phi$ )