LANGAGE C-PASCAL

Grammaire Prédictive LL1

Réalisé par :

* Abchir Ouijdane
* Amanzou Anas
* Bahbah Ibrahim
* Lahnech Hamza

1. Introduction

Dans cette partie on rendra la grammaire du language C- Pascal LL( 1 ) en élevant toute éventuelle récursivité à gauche, non factorisation ou ambiguïté, dans le but de la rendre analysable par la méthode descendante récursive. Pour ce, on vérifiera ces conditions pour chaque production du grammaire, en évoquant toutes les justifications possibles .

Encadré par :

* Pr ELGHAZI Souhail

**//OUIJDANE ABCHIR**

* **Grammaire finale avec le tableau des premiers et des suivants**

1. <Programme> : : = <liste-declarations>< liste-fonctions>|< liste-fonctions>

//< liste-declarations> : : = < liste-declarations><declaration> | **epsilon**

//<liste-declarations> : : = <liste-declarations> ;<declaration> | <declaration>  propre

1. <liste-declarations> : : = <declaration> ;<liste-declarations’>
2. <liste-declarations’>: : = <declaration> ;<liste-declarations’>  | **epsilon**
3. <liste-declarations-sp> : : = <declaration-sp> ;<liste-declarations-sp’>
4. <liste-declarations-sp’> : : = <declaration-sp> ;<liste-declarations-sp’>  | **epsilon**

//<declaration> : : = <declaration>, < declarateur-cp> | < declarateur-cp>

1. <declaration> : : = <declarateur-cp><declaration’>
2. <declaration’> : : = ,< declarateur-cp><declaration’>**|epsilon**
3. <declaration-sp> : : = <declarateur-sp>< declaration-sp’>
4. <declaration-sp’> : : = ,< declarateur-sp><declaration-sp’>**|epsilon**
5. <declarateur-sp> : : = **entier**<variable>
6. <declarateur-cp> : : = **entier** <declarateur-cp’> | **struct** <variablestruct>**{**<liste-declarations-sp>}
7. <declarateur-cp’> : : = <variabletab>[<nombre>] | < variable>\*
8. < liste-fonctions> : : = <liste-fonctions’> **main()**<fonction’’>
9. < liste-fonctions’> : : = < fonction> <liste-fonctions’> | **epsilon**

// <fonction> : : = <identificateur> ( <liste-parms> ) < liste-declarations-sp><bloc> |<identificateur> ( <liste-parms> )<bloc> |**main()** < liste-declarations-sp><bloc> |**main()**<bloc>

1. <fonction> : : = <identificateur>(<liste-parms>)<fonction’>
2. <fonction’> : : = < liste-declarations-sp><bloc> | <bloc>
3. <fonction’’> : : = < liste-declarations-sp><blocmain> | <blocmain>
4. <liste-parms> : : = <parm><liste-parms’>
5. <liste-parms’> : : = ,<parm><liste-parms’> | **epsilon**
6. <parm> : : = **entier** <variable>
7. <bloc> : : = {<liste-instructions>}
8. <blocmain> : : = {<liste-instructions>}
9. <blocfaire> : : = {<bloc’>
10. <blocalors> : : = {<bloc’>

<bloc’> : : = <liste-instructions>} | }

1. <liste-instructions> : : = <instruction ><liste-instructions’>
2. <liste-instructions’>: : = <instruction ><liste-instructions’> | **epsilon**
3. <instruction> : : = < iteration>|<selection>| <saut> ; | <affectation> ;|<appel> ;
4. <iteration> : : = **tantque** <condition> **faire** <blocfaire>
5. <selection> : : = **si** <condition> **alors** <blocalors> <selection’>
6. <selection’> : : = **sinon** <blocfaire> | **epsilon**
7. <saut> : : = **retour** <expression>
8. <affectation> : : = <variable>=<affectation’> | <variablestruct>.<variable>=<affectation’>|<variabletab>[<nombre>]=< affectation’>\*

<affectation ’> : : = <expression> | **lire** ( )|<appel>\*

<affectation ’’> : : = <expression> | **lire** ( )|<appel>\*

1. <affectation ’> : : = <expression> | **lire** ( )|<appel>\*
2. <appel> : : = <identificateur> ( <liste-expressions>) | **ecrire** (<expression>)
3. <liste-expressions> : : = <expression><liste-expressions’>
4. <liste-expressions’>: : = ,< expression><liste-expressions’> | **epsilon**
5. <condition> : : = !(<expression>) | <expression><condition’>
6. <condition’> : : = <binary-rel><expression> | <binary-comp><expression>
7. <expression> : : = <terme><expression’>
8. <expression’> : : = <binary-op><expression> | **epsilon**
9. <terme>  : : = <variable> | <nombre> |<variablestruct>.<variable>|<variabletab>[<nombre>]\*
10. <variable>  : : = < lettre><variable’>
11. <variablestruct>  : : = < lettre><variable’>
12. <variabletab>  : : = < lettre><variable’>
13. <variable’>  : : = < variable’’><variable’> | **epsilon**
14. <variable’’>  : : = < lettre>| <nombre>
15. <nombre>  : : = <constante><nombre’>
16. <nombre’>  : : = <constante><nombre’> | **epsilon**
17. <identificateur>  : : = < lettre>< identificateur’>
18. < identificateur’>  : : = < variable’’>< identificateur’> | **epsilon**
19. <lettre>  : : = A|B|C|...|Z|a|b|c|...­|z
20. <constante>  : : = 0|1|…­|9
21. <binary-op> : : = +|-|\*|/
22. <binary-rel> : : = **&** | |
23. <binary-comp> : : = <|>|>=|<=|==|!=

// à verifer

1. <caractere> : : = <constante>|<lettre>|<binary-op>|<binary-rel>|<binary-comp><caractere>|**epslion**
2. <ligne> : : = <caractere><ligne’ >
3. <ligne’> : : = <caractere><ligne’>|**epsilon**
4. <commentaire> : : = #<ligne>

|  |  |  |
| --- | --- | --- |
| Symboles non terminaux | Premiers | Suivants |
| R1 | Entier,struct, A,B,C,...,Z,a,b,c,...,z,main | **$** |
| R2 | entier,struct | A,B,C,...,Z,a,b,c,...,z,main |
| R3 | entier,struct , **epsilon** | A,B,C,...,Z,a,b,c,...,z,main |
| R4 | entier | **{ ,}** |
| R5 | entier , **epsilon** | **{ ,}** |
| R6 | entier,struct | **,** , **;** |
| R7 | entier,struct, **epsilon** | **,** , **;** |
| R8 | entier | **,** , **;** |
| R9 | entier, **epsilon** | **,** , **;** |
| R10 | entier | **,** , **;** |
| R11 | entier,struct | **,** , **;** |
| R12 | A,B,C,...,Z,a,b,c,...,z | **,** , **;** |
| R13 | A,B,C,...,Z,a,b,c,...,z,main | **$** |
| R14 | A,B,C,...,Z,a,b,c,...,z, **epsilon** | main |
| R15 | A,B,C,...,Z,a,b,c,...,z | A,B,C,...,Z,a,b,c,...,z , main |
| R16 | Entier , { | A,B,C,...,Z,a,b,c,...,z, main |
| R17 | Entier | **$** |
| R18 | Entier | **)** |
| R19 | Entier , **epsilon** | **)** |
| R20 | Entier | **,** , ) |
| R21 | { | A,B,C,...,Z,a,b,c,...,z, main() |
| R22 | { | **$** |
| R23 | { | Tantque , si , ecrire , retour , } ,A,B,C,...,Z,a,b,c,...,z |
| R24 | { | Tantque , si , ecrire , retour , sinon , } , A,B,C,...,Z,a,b,c,...,z |
| R25 | Tantque , si , retour , A,B,C,...,Z,a,b,c,...,z | } , ; |
| R26 | Tantque , si , retour , A,B,C,...,Z,a,b,c,...,z,**epsilon** | } , ; |
| R27 | Tanque , si , retour , A,B,C,...,Z,a,b,c,...,z | Tantque , si , ecrire , retour , A,B,C,...,Z,a,b,c,...,z |
| R28 | Tantque | Tantque , si , ecrire , retour , } ,A,B,C,...,Z,a,b,c,...,z |
| R29 | Si | Tantque , si , ecrire , retour , sinon , } , A,B,C,...,Z,a,b,c,...,z |
| R30 | Sinon | Tantque , si , ecrire , retour , } ,A,B,C,...,Z,a,b,c,...,z |
| R31 | Retour | **;** |
| R32 | A,B,C,...,Z,a,b,c,...,z | **;** |
| R33 | A,B,C,...,Z,a,b,c,...,z , lire | **;** |
| R34 | A,B,C,...,Z,a,b,c,...,z , ecrire | **; , , , )** |
| R35 | A,B,C,...,Z,a,b,c,...,z | ) |
| R36 | **,** , **epsilon** | ) |
| R37 | ! , A,B,C,...,Z,a,b,c,...,z | Faire , alors |
| R38 | | , & , < , >, >= , <= , == , != | Faire , alors |
| R39 | A,B,C,...,Z,a,b,c,...,z | **,** , **; , )** |
| R40 | **= ,** +, - ,\* , / , **epsilon** | **,** , **; , )** |
| R41 | 0 , 1 , …­ , 9 , A,B,C,...,Z,a,b,c,...,z | **,** , **; , = ,** +, - ,\* , / , < , >, >= , <= , == , != , & , | , ) , faire , alors |
| R42 | A,B,C,...,Z,a,b,c,...,z | **,** , **; , = ,** +, - ,\* , / , < , >, >= , <= , == , != , & , | , ) , faire , alors |
| R43 | A,B,C,...,Z,a,b,c,...,z | . |
| R44 | A,B,C,...,Z,a,b,c,...,z | [ |
| R45 | A,B,C,...,Z,a,b,c,...,z**, epsilon** | **,** , **; , = ,** +, - ,\* , / , < , >, >= , <= , == , != , & , | , ) , faire , alors |
| R46 | A,B,C,...,Z,a,b,c,...,z, | **,** , **; , = ,** +, - ,\* , / , < , >, >= , <= , == , != , & , | , ) , faire , alors |
| R47 | 0 , 1 , …­ , 9 | **,** , **; , = ,** +, - ,\* , / , < , >, >= , <= , == , != , & , | , ) , ] , faire , alors |
| R48 | 0 , 1 , …­ , 9, **epsilon** | **,** , **; , = ,** +, - ,\* , / , < , >, >= , <= , == , != , & , | , ) , ] , faire , alors |
| R49 | A,B,C,...,Z,a,b,c,...,z | **(** |
| R50 | A,B,C,...,Z,a,b,c,...,z , **epsilon** | **(** |
| R51 | A,B,C,...,Z,a,b,c,...,z | **,** , **; , = ,** +, - ,\* , / , < , >, >= , <= , == , != , & , | , ) , ] , faire , alors , (,A,B,C,...,Z,a,b,c,...,z , 0 , 1 , …­ , 9 |
| R52 | 0 , 1 , …­ , 9 | 0 , 1 , …­ , 9 , A,B,C,...,Z,a,b,c,...,z , **,** , **; , = ,** +, - ,\* , / , < , >, >= , <= , == , != , & , | , ) , ] , faire , alors , ( |
| R53 | **= ,** +, - ,\* , / | 0 , 1 , …­ , 9 , A,B,C,...,Z,a,b,c,...,z |
| R54 | | , & | 0 , 1 , …­ , 9 , A,B,C,...,Z,a,b,c,...,z |
| R55 | < , >, >= , <= , == , != | 0 , 1 , …­ , 9 , A,B,C,...,Z,a,b,c,...,z |

* **Vérification des productions**