Compte-rendu pour TP N°1 : Compilation

Sujet : réalisation d'un analyseur lexicale et syntaxique pour une grammaire SLR **.**

Réalisé par :

- Jouini Wajih (IDS3)
- Chaari Mahmoud (IGL3)

Année universitaire : 2022 / 2023

Chapitre 1 : Choix du grammaire SLR

Pour ce projet nous avons recours à une grammaire SLR qui définie un simple langage de programmation. Cette grammaire nous permet de bien définir les différents types de variables (entier, caractère, booléen, réel, chaine, pointeur …) ,les fonctions logiques (ET,OU), les structures de données (Tableau) et des instructions itératives(Boucle pour ,boucle répéter) et conditionnelles (si alors , si alors sinon) .

Cette grammaire peut aussi supporter les fonctions et les classes.

Voici la grammaire :

S->PRG

PRG->package id class id {CONTENT}

CONTENT->DEC_LIST CONTENT

CONTENT->DEC_LIST

CONTENT->INS_LISTI CONTENT

CONTENT->INS_LIST

CONTENT->IF CONTENT

CONTENT->IF

IF->if (T) {CONTENT} else {CONTENT}

IF->if (T) {CONTENT}

CONTENT->BOUCLE CONTENT

CONTENT->BOUCLE

BOUCLE->while (T) {CONTENT}

BOUCLE->for (id ; T ; INS) { CONTENT }

CONTENT->FNCT CONTENT

CONTENT->FNCT

DEC_LIST->DEC ; DEC_LIST

DEC_LIST->DEC ;

DEC->TYPE id

DEC->tab id [TYPE]

TYPE->int

TYPE->float

TYPE->char

TYPE->String

INS_LIST->INS ; INS_LIST

INS_LIST->INS ;

INS->id := T

INS->print (T)

T->id

T->(T)

T->T or T

T->T o T

T->T et T

T->T OU T

T->NBR

T->FLT

FNCT->TYPE id (TYPE id) { CONTENT }

Chapitre 2 : Tableau des premiers et suivants

Item	Premiers	Suivants
S	{package}	{$}
PRG	{package}	{$}
CONTENT	{INS_LISTI,if,while,for,int,float,char,String,tab,id,print}	{}}
IF	{if}	{INS_LISTI,if,while,for,int,float,char,String,tab,id,print,}}
BOUCLE	{while,for}	{INS_LISTI,if,while,for,int,float,char,String,tab,id,print,}}
DEC_LIST	{tab,int,float,char,String}	{INS_LISTI,if,while,for,int,float,char,String,tab,id,print,}}
DEC	{tab,int,float,char,String}	{;}
TYPE	{int,float,char,String}	{id,]}
INS_LIST	{id,print}	{}}
INS	{id,print}	{),;}
T	{id,(,NBR,FLT}	{),;,or,o,et,OU}
FNCT	{int,float,char,String}	{INS_LISTI,if,while,for,int,float,char,String,tab,id,print,}}

Goto	Kernel
	{S -> .PRG}
goto(0, PRG)	{S -> PRG.}
goto(0, package)	{PRG -> package.id class id { CONTENT }}
goto(2, id)	{PRG -> package id.class id { CONTENT }}
goto(3, class)	{PRG -> package id class.id { CONTENT }}
goto(4, id)	{PRG -> package id class id.{ CONTENT }}
goto(5, {)	{PRG -> package id class id {.CONTENT }}
goto(6, CONTENT)	{PRG -> package id class id { CONTENT.}}
goto(6, DEC_LIST)	{CONTENT -> DEC_LIST.CONTENT; CONTENT -> DEC_LIST.}
goto(6, INS_LISTI)	{CONTENT -> INS_LISTI.CONTENT}
goto(6, INS_LIST)	{CONTENT -> INS_LIST.}
goto(6, IF)	{CONTENT -> IF.CONTENT; CONTENT -> IF.}
goto(6, BOUCLE)	{CONTENT -> BOUCLE.CONTENT; CONTENT -> BOUCLE.}
goto(6, FNCT)	{CONTENT -> FNCT.CONTENT; CONTENT -> FNCT.}
goto(6, DEC)	{DEC_LIST -> DEC.; DEC_LIST; DEC_LIST -> DEC.;}
goto(6, INS)	{INS_LIST -> INS.; INS_LIST; INS_LIST -> INS.;}
goto(6, if)	{IF -> if.(T) { CONTENT } else { CONTENT }; IF -> if.(T) { CONTENT }}
goto(6, while)	{BOUCLE -> while.(T) { CONTENT }}
goto(6, for)	{BOUCLE -> for.(id ; T ; INS) { CONTENT }}
goto(6, TYPE)	{FNCT -> TYPE.id (TYPE id) { CONTENT }; DEC -> TYPE.id}
goto(6, tab)	{DEC -> tab.id [TYPE]}
goto(6, id)	{INS -> id.:= T}
goto(6, print)	{INS -> print.(T)}
goto(6, int)	{TYPE -> int.}
goto(6, float)	{TYPE -> float.}
goto(6, char)	{TYPE -> char.}
goto(6, String)	{TYPE -> String.}
goto(7, })	{PRG -> package id class id { CONTENT }.}
goto(8, CONTENT)	{CONTENT -> DEC_LIST CONTENT.}
goto(8, DEC_LIST)	{CONTENT -> DEC_LIST.CONTENT; CONTENT -> DEC_LIST.}
goto(8, INS_LISTI)	{CONTENT -> INS_LISTI.CONTENT}
goto(8, INS_LIST)	{CONTENT -> INS_LIST.}
goto(8, IF)	{CONTENT -> IF.CONTENT; CONTENT -> IF.}
goto(8, BOUCLE)	{CONTENT -> BOUCLE.CONTENT; CONTENT -> BOUCLE.}
goto(8, FNCT)	{CONTENT -> FNCT.CONTENT; CONTENT -> FNCT.}

Le table SLR est tellement grand (95 lignes) qu'on ne peut pas l'insérer dans ce rapport.

Le table SLR et le calcul des premiers et suivants est fait à l'aide du site web ParseSLR

Chapitre 3 : Exemple d'exécution

Ce programme est développé en langage Java. Voici un simple exemple d'exécution

```
1   package comp class test {
2       int i ;
3       tab t [ int ] ;
4       string ch ;
5   print ( x ) ; }
```

```
wajih/mahmoud compilateur version 0.0

donner le nom de fichier qui contient le code à compilé : code.txt

------------------------Analyse lexical------------------------------
les unités lexicales:
<keyword,package>
<id,id>
<keyword,class>
<id,id>
<par,{>
<keyword,int>
<id,id>
<pv,;>
<keyword,tab>
<id,id>
<par,[>
<keyword,int>
<par,]>
<pv,;>
<keyword,string>
<id,id>
<pv,;>
<keyword,print>
<par,(>
<id,id>
<par,)>
<pv,;>
<par,}>
------------------------Fin de l'analyse lexical---------------------
```

```
-----------------------Analyse syntaxique--------------------------

_____________________________________________________________________
pile:   [0]
_____________________________________________________________________
Entrée: packageidclassid{intid;tabid[int];stringid;print(id);}$
_____________________________________________________________________
Action:
_____________________________________________________________________




_____________________________________________________________________
pile:   [0, package, 2]
_____________________________________________________________________
Entrée: idclassid{intid;tabid[int];stringid;print(id);}$
_____________________________________________________________________
Action: shift
_____________________________________________________________________




_____________________________________________________________________
pile:   [0, package, 2, id, 3]
_____________________________________________________________________
Entrée: classid{intid;tabid[int];stringid;print(id);}$
_____________________________________________________________________
Action: shift
_____________________________________________________________________
```

```
_____________________________________________________________________
pile:   [0, package, 2, id, 3, class, 4, id, 5, {, 6, CONTENT, 7]
_____________________________________________________________________
Entrée: }$
_____________________________________________________________________
Action: reduce:CONTENT->DEC_LIST CONTENT
_____________________________________________________________________




_____________________________________________________________________
pile:   [0, package, 2, id, 3, class, 4, id, 5, {, 6, CONTENT, 7, }, 27]
_____________________________________________________________________
Entrée: $
_____________________________________________________________________
Action: shift
_____________________________________________________________________




_____________________________________________________________________
pile:   [0, PRG, 1]
_____________________________________________________________________
Entrée: $
_____________________________________________________________________
Action: reduce:PRG->package id class id { CONTENT }
_____________________________________________________________________


------------------analyze SLR successfully---------------------
```