Les règles de production de la grammaire :

S → **programme Ident ; D debut Inst fin**
D → **C V**
C → **const Dec / λ**
Dec → **Ident = Num ; Dec / Ident = Num**
V → **var Dev / λ**
Dev → **Ident M : Type ; Dev / Ident M : Type**
Type → **ent / bool**
Inst → **Instr / λ**
Instr → **Ident := Exp I / si Ident alors Instr sinon Instr fsi / si Ident alors Instr fsi**
I → **; Instr / λ**
Exp → **Exp + Exp / Ident / Cste**
Ident → **Lettre SuitL**
SuitL → **Lettre SuitL / Chiffre SuitL**
Lettre → **a/b/…./z/A/B/…/Z**
Chiffre → **0/1/2/3/4/5/6/7/8/9**
Cste → **Chiffre SuitC**
SuitC → **SuitC Chiffre / λ**
M → **, Ident M/ λ**
Num → **Cste / vrai / faux**

1- implementation dans la parti Flex/Lex

C'est pour l'analyse lexical de la grammaire , entre autre en retrouve les différents terminaux de la grammaire et l'ajout d'une variable nombre de ligne qui cert a indiquer dans quelle ligne se trouve l'erreur

2- implementation dans la parti Yacc/Bison

- C'est pour l'analyse syntaxique de la grammaire , en retrouve les différents non terminaux associer a leur règle respective.

- Pour résoudre le problème de conflit/reduction, il suffit de declarer la priorité a l'opérateur dans la règle

Exp → Exp + Exp

pour cela il faut ajouter %left "+" dans le fichier yacc.

- La fonction **yyerror()** a été redéfini pour afficher dans quelle ligne se trouve l'erreur grace a la variable **numLigne** qui est incrémenter a chaque fois qu'on lit un saut de ligne.

3- implementation de la table des symboles

- Dans le fichier tableSymbole.h en retrouve :
  - une structure de donnée pour une liste chainé.
  - plusieurs fonctions pour la manipulation de la table des symboles.

- Pour la structure de la table des symbole , elle contient 3 champs (**code** , **Identifiant** , **l'adresse du suivant**).
- Le code sert a exprimé le type de l'identifiant (1 constant , 2 variable ent , 3 variable bool).

- La fonction **init()** cert a initialiser notre liste a zero.

- La fonction **rechercher()** permet de rechercher un identifiant si il existe deja dans la table pour éviter des redondances d'insertion inutile.

- La fonction **insertionFin()** permet de faire une insertion en fin de la table.

- La fonction **insertionDebut()** permet de faire une insertion en debut de la table.

- La fonction **isMotCle()** permet de verifier si notre identifiant n'est pas un mot clé de la grammaire (programme si alors .. ect)

- La fonction **insertion()** permet de tester notre identifiant grace aux fonction **recherche()** **isMotCle()** et **insertiondebut()** ou **insertionFin()** , c'est la fonction a utiliser dans le fichier lex ou yacc.

- La fonction **afficheTable()** permet d'afficher notre table de symbole une fois l'analyse terminer.