**l'analyseur sémantique :**

Dcl →dcl

dcl → var Liste\_id : Type ; dcl | Ꜫ {add\_listof\_type(list,typeo)}//similaire a ajouter\_type() dans le cour

Type 🡪interger {type.typeo :=integer}

Type🡪char {type.typeo:=char}

I🡪 id := Exp\_simple{I.type:=si compatible(rechercher(id),Exp\_simple.type) alor vide sinon erreur\_de\_type}

Exp\_simple → Terme exp\_simple

Exp\_simple → opadd Terme exp\_simple | Ꜫ

Terme → Facteur terme

terme → opmul Facteur terme | Ꜫ

Facteur → Id |nb | (exp\_simple)

{Facteur.type=rechercher(id)}

{Facteur.type=integrer}

Dans le code updateLastExpressionType() fonction pour determiner le type de Exp\_simple() retourne -1 si l expression est mixte et retourne type de id ou integer pour une expression des entiers.

**Le generateur de code:**

I🡪id := Exp\_simple {Inst.t:=Valeurg id ||Exp\_simple.t||:=}

I🡪id {emettre(valeurg,id)}

:=

Exp\_simple

{emettre ( :=)}

**if**

I🡪si Exp alors I {sortie := sortie\_etiq() ;I.t := || allersifaux sortie ||I.t||Etiq sortie}

I→ Si Exp

{Sortie := sortie\_etiq();

Emettre(Allersifaux Sortie)}

alors

I

{Emettre(Etiq Sortie)}

**while**

I→ Tant que Exp faire I

{Test :=sortie\_etiq();

Sortie := sortie\_etiq();

I.t := Etiq Test || Exp.t

|| Allersifaux Sortie || I.t

|| Allerà Test

|| Etiq Sortie}

Inst → Tant que

{Test := sortie\_etiq(); Sortie := sortie\_etiq ();

Emettre(Etiq Test);}

Exp

{Emettre(Allersifaux Sortie);}

Faire

I

{Emettre(Allerà Test)

Emettre(Etiq Sortie)}

Exp\_simple 🡪Terme1 + exp\_simple1 {Exp.t := Exp1.t || Terme.t || +}

{ emettre(+)}

Exp\_simple 🡪Terme1 - exp\_simple1 {Exp.t := Exp1.t || Terme.t || -}

{ emettre(-)}

Exp\_simple 🡪Terme1 ||exp\_simple1 {Exp.t := Exp1.t || Terme.t || ||}

{ emettre(||)}

Terme 🡪 Facteur \* terme {Terme.t:= Terme1.t || Facteur.t ||\*} {emettre(\*)}

Terme 🡪 Facteur / terme {Terme.t:= Terme1.t || Facteur.t ||/}{emettre(/)}

Terme 🡪 Facteur && terme {Terme.t:= Terme1.t || Facteur.t ||&&} {emettre(&&)}

Facteur → nb {Facteur.t := empiler nb.val} {Emettre(Empiler nb.val)}

Facteur → (Exp\_simple) {Facteur.t := Exp\_simple.t}

Facteur → id {Facteur.t := Valeurd id.entrée} {Emettre(Valeurd id.entrée}

EXP🡪 Exp\_simple == Exp\_simple {Emettre(comparer-si-egal)}

EXP🡪 Exp\_simple <= Exp\_simple {Emettre(comparer-si-inferieur-ou-egal)}

EXP🡪 Exp\_simple >= Exp\_simple {Emettre(comparer-si-superieur-ou-egal)}

EXP🡪 Exp\_simple <> Exp\_simple {Emettre(comparer-si-diffrent)}

EXP🡪 Exp\_simple > Exp\_simple {Emettre(comparer-si-superieur)}

EXP🡪 Exp\_simple < Exp\_simple {Emettre(comparer-si-inferieur)}