Ministère de l'Enseignement supérieur et de la Recherche scientifique

Université de La Manoube

Ecole Nationale des Sciences de l'Informatique


---

# Rapport
# Mini Projet de Compilation

---

**Réalisé par :**

OUACHANI Ahmed

Classe II2D

Année Universitaire 2017/2018

**Partie I**

**Analyse Lexicale**

| Unité Lexicale | Modèle |
|---|---|
| program | program |
| var | var |
| integer | integer |
| char | char |
| begin | begin |
| end | end |
| if | if |
| then | then |
| else | else |
| while | while |
| do | do |
| read | read |
| readln | readln |
| write | write |
| writeln | writeln |
| ; | ; |
| : | : |
| . | . |
| := | := |
| ; | ; |
| ) | ) |
| ( | ( |
| nb | Chiffre . Chiffre* |
| id | Lettre . (Lettre \| Chiffre)* |
| oprel | == \| <> \| < \| > \| <= \| >= |
| opadd | + \| - \| \|\| |

| | |
|---|---|
| opmul | / \| * \| % \| && |

**Partie II**

**Analyse Syntaxique**

| |
|---|
| P → **program id ;** Dcl Inst_composée **.** |
| Dcl → **var** Liste_id **:** Type **;** Dcl \| ε |
| Liste_id → **id** Liste_in |
| Liste_in → **, id** Liste_in \| ε |
| Type → **integer** \| **char** |
| Inst_composée → **begin** Inst **end** |
| Inst → Liste_inst \| ε |
| Liste_inst → I Liste_I |
| Liste_I → **;** I Liste_I \| ε |
| I → **id :=** Exp_simple \| **if** Exp **then** I **else** I \| **while** Exp **do** I \| **read** ( id ) \| **readln** ( id )\| **write** ( id ) \| **writeln** ( id ) |
| Exp → Exp_simple Exp_I |
| Exp_I → **oprel** Exp_simple \| ε |
| Exp_simple → Terme Exp_S |
| Exp_S → **opadd** Terme Exp_S \| ε |
| Terme → Facteur Term_I |
| Terme_I → **opmul** Facteur Terme_I \| ε |
| Facteur → **id** \| **nb** \| **(** Exp_simple **)** |

| Les Terminaux | Les non-terminaux |
|---|---|
| id nb oprel opadd opmul if then else while do; , : read write readln writeln var integer char program begin end . ( ) | P Dcl Inst_composée Liste_id Liste_in Type Inst Liste_inst Liste_I I Exp_simple Exp Exp_I Terme Exp_S Facteur Term_I |

| Non Terminal | Premiers | Suivants |
|---|---|---|
| P | program | $ |
| Dcl | var \| ε | begin |
| Liste_id | id | : |
| Liste_in | , \| ε | : |
| Type | integer \| char | ; |
| Inst_composée | begin | . |
| Inst | id \| if \| while \| read \| readln \| write \| writeln \| ε | end |
| Liste_inst | id \| if \| while \| read \| readln \| write \| writeln | end |
| Liste_I | ; \| ε | end |
| I | id \| if \| while \| read \| readln \| write \| writeln | ; \| end \| else |
| Exp | id \| nb \| ( | then \| do |
| Exp_I | oprel \| ε | then \| do |
| Exp_simple | id \| nb \| ( | ; \| end \| else \| oprel \| then \| do \| ) |
| Exp_S | opadd \| ε | ; \| end \| else \| oprel \| then \| do \| ) |
| Terme | id \| nb \| ( | opadd \| ; \| end \| else \| oprel \| then \| do \| ) |
| Terme_I | opmul \| ε | opadd \| ; \| end \| else \| oprel \| then \| do \| ) |
| Facteur | id \| nb \| ( | opmul \| opadd \| ; \| end \| else \| oprel \| then \| do \| ) |

## Partie III

### Analyse Sémantique et Génération de Code.

*Pour Les types des ids : 0 signifie non déclaré, 1 pour les char, 2 pour les integer et 3 pour le nom du programme.*

| Instruction | Règles sémantiques et Code intermédiaire |
|---|---|
| P → **program id ;** Dcl Inst_composée **.** | **id**.type = **3** *//Pour le nom du programme* |

| | |
|---|---|
| Dcl → var Liste_id : Type ; Dcl | Liste_id = **new temp**<br>Type = **new temp**<br>Liste_id.place = Type.place |
| Dcl → ε | // |
| Liste_id → id Liste_in | Si **id.**type = **0** alors<br>**id.**type.place = liste_id.place<br>sinon ***Erreur***. |
| Liste_in → **,** **id** Liste_in | Si **id.**type = **0** alors<br>**id.**type.place = liste_id.place<br>sinon ***Erreur***. |
| Liste_in → ε | // |
| Type → **integer** | Type.place = **2** |
| Type → **char** | Type.place = **1** |
| Inst_composée → **begin** Inst **end** | // |
| Inst → Liste_inst \| ε | // |
| Liste_inst → I Liste_I | // |
| Liste_I → **;** I Liste_I \| ε | // |
| I → **id :=** Exp_simple | Exp_simple = **new temp**<br>Si **id.**type !=0 alors<br>Exp_simple.type = **id.**type<br>Sinon **Erreur**.<br>***Gen(id.place := Exp_simple.place) ;*** |
| I → **if** Exp **then** I **else** I | ***Gen(Si Exp.place !=0)***<br>***Incomplète.*** |
| I → **while** Exp **do** I | ***Gen(Tantque Exp.place !=0 ) ;***<br>***Incomplète.*** |
| I → **read** ( id ) | ***Gen(read ( id ) );*** |
| I → **readln** ( id ) | ***Gen(readln ( id ) );*** |
| I → **write** ( id ) | ***Gen(write ( id ) );*** |

| | |
|---|---|
| I → **writeln** ( id ) | ***Gen(writeln ( id ) );*** |
| Exp → Exp_simple Exp_I | Exp_simple = **new temp**<br>Exp_I = **new temp**<br>Exp_simple.type = Exp.type<br>Exp_I.type = Exp.type<br>***Gen(Exp.place := Exp_simple.place) ;***<br>***Gen(Exp_I.place := Exp_simple.place) ;*** |
| Exp_I → **oprel** Exp_simple | Exp_simple = **new temp**<br>Exp_I.type = Exp_I.type<br>***Gen(Exp_I.place := Exp_I.place oprel Exp_simple.place);*** |
| Exp_I → ε | // |
| Exp_simple → Terme Exp_S | Terme = **new temp**<br>Exp_S = **new temp**<br>Terme.type = Exp_simple.type<br>Exp_S.type = Exp_simple.type<br>***Gen(Exp_simple.place := Terme.place) ;***<br>***Gen(Exp_S.place := Terme.place) ;*** |
| Exp_S → ε | // |
| Exp_S → **opadd** Terme Exp_S | Terme = **new temp**<br>Exp_S1 = **new temp**<br>Terme.type = Exp_S.type<br>Exp_S1.type = Exp_S.type<br>***Gen(Exp_S.place := Terme.place opadd Exp_S.place) ;***<br>***Gen(Exp_S1.place := Terme.place) ;*** |
| Terme → Facteur Term_I | Facteur = **new temp**<br>Term_I = **new temp**<br>Facteur.type = Terme.type<br>Terme_I.type = Terme.type<br>***Gen(Terme.place = Facteur.type) ;***<br>***Gen(Terme_I.type = Facteur.type) ;*** |