# ECRITURE D'UN MINI COMPILATEUR

## ANALYSE SEMANTIQUES

```
PROGRAM ::= program ID; BLOCK.
```

BLOCK ::= CONSTS VARS INSTS

CONSTS ::= const ID = NUM; { ID = NUM; }  $\epsilon$ 

VARS ::=  $var ID \{, ID \}; | \epsilon$ 

INSTS ::= begin INST { ; INST } end

•Les identificateurs et les constantes sont les objets sémantiques du programme

•Ils seront utilisés lors du calcul de l'analyse sémantique du programme



Il faut les mémoriser avec leurs propriétés



**TABLE DES SYMBOLES** 

**IDENTIFICATEURS** 

СН	CLS
test	ID_TOKEN
toto	ID_TOKEN
titi	ID_TOKEN
X	ID_TOKEN
у	ID_TOKEN

IL FAUT INSERER LES IDENTIFICATEURS
DANS LA TABLE DES SYMBOLES





Y a des règles sémantiques, des contrôles sémantiques

# LE CONTRÔLE SEMANTIQUE DES DECLARATIONS ET DE TYPE DES IDENTIFICATEURS

```
PROGRAM ::= program ID; BLOCK.

BLOCK ::= CONSTS VARS INSTS

CONSTS ::= const ID = NUM; { ID = NUM; } | ε

VARS ::= var ID { , ID }; | ε

INSTS ::= begin INST {; INST } end
```

#### Règles sémantiques:

- 1. Règle 1: Toutes les déclarations dans CONSTS et VARS
- 2. Règle 2: PAS DE DOUBLE DECLARATIONS
- 3. Règle 3: Apres BEGIN, tous les symboles doivent être déjà déclarés
- 4. Règle 4: Une constante ne peut changer de valeur dans le programme
- 5. Règle 5: Le ID du programme ne peut être utilisé dans le programme

#### **Exemple 1:**

```
program test;
const tata=12;
var x;
begin
titi:=tata;
end.
```

ERR: identificateur titi non déclaré

#### **Exemple 2:**

```
program test;
const tata =12;
var x,tata;
begin
x:=tata;
end.
```

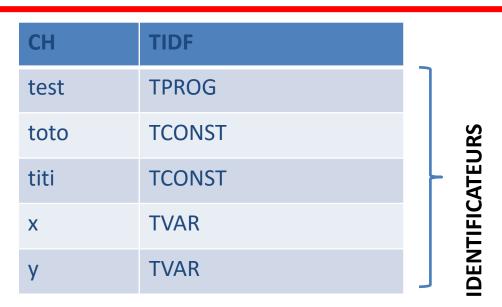
**ERR:** double déclaration

```
Exemple 3:
                                                           Exemple 5:
  program test;
                                                            program test;
  const tata=12;
                                                           const tata=12;
  var x;
                                                           var x;
  begin
                                                           begin
                                                             read(tata);
    tata :=15;
  end.
                                                           end.
ERR: constante ne peut changer de valeur
  Exemple 4:
  program test;
  const tata=12;
  var x;
                                       ERR: constante ne peut changer de valeur
  begin
    x :=test ;
  end.
```

ERR: nom de programme non autorisé

R. Oulad Haj Thami

```
Exemple:
program test;
const toto=12; titi=23;
var x, y;
begin
.....
end.
```



```
Typedef enum {TPROG, TCONST, TVAR} TSYM;

Typedef struct {
    char NOM[20];
    TSYM TIDF;
} T_TAB_IDF;
T_TAB_IDF TAB_IDFS[NbrIDFS];
```

# TRAITEMENT SEMANTIQUE DES NOMBRES

### TRAVAIL A FAIRE

- 1. Implémentation de la table des symboles
- 2. Implémentation du contrôle des déclarations
- 3. Implémentation du contrôle des constantes et du nom du programme
- 4. Traitement des nombres
- 5. Tester et valider

## SEANCE LIBRE

# A VOS MACHINES et BON COURAGE