Chap. 2 : Analyseur syntaxique

Ecriture de l'analyseur syntaxique

Principe

A chaque règle:

$$R \rightarrow \alpha$$

On associe une procédure de la forme:

```
PROCEDURE R()
debut
        T(α);
fin
```

Traduction de la partie droite d'une règle

α	**Traitement associé a α**
a $\in V_t$ a_TOKEN	si (SYMB_COUR**.TOKEN**==**a_TOKEN**) alors SYMB_SUIV() sinon ERREUR(C_ERR) ;
A $\in V_n$	A() (*appel de la procédure associée à la règle A*)
ε	(*instruction vide*)
$\beta_1\beta_2$	T(β_1) T(β_2)
$\beta_1 \mid \beta_2$	cas SYMB_COUR**.TOKEN** parmi **D(β1| β2, β1)** : T(β_1) **D(β1| β2, β2)** : T(β_2) sinon ERREUR(C_ERR) fin cas
β*	tant que (SYMB_COUR**.TOKEN dans β′**) faire T(β) fin tant que

Ecriture des procédures syntaxiques

- Une procédure TEST_ACCEPT() teste si le prochain token est bien celui passé en paramètre à la procédure ; on s'arrête sur une erreur sinon (procédure ERREUR).
- **procedure TESTE_ACCEPT (T : TOKENS, C : CODES_ERR)**

```
debut
 si   (SYMB_COUR.TOKEN = T) alors
        SYMB_SUIV()
 sinon
        ERREUR(C)
 fin si
fin
```

procédure syntaxique de la règle PROGRAM

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

```
PROCEDURE   PROGRAM()
debut
        TEST_ACCEPT(PROGRAM_TOKEN, PROGRAM_ERR);
        TEST_ACCEPT(ID_TOKEN, ID_ERR);
        TEST_ACCEPT(PVIR_TOKEN, PVIR_ERR);
        BLOCK();
        TEST_ACCEPT(PNT_TOKEN, PNT_ERR);
fin
```

procédure syntaxique de la règle BLOCK

BLOCK ::= CONSTS VARS INSTS

```
PROCEDURE   BLOCK()
debut
        CONSTS();
        VARS();
        INSTS();
fin
```

Travail à faire : Analyseur syntaxique

- Programmer toutes les procédures pour toutes les règles syntaxiques.
- Tester votre analyseur syntaxique.