Langage Z

(Niveau 1)

D E S C R I P T I O N

Généralités

> Un Z-algorithme est un ensemble de modules parallèles dont le premier est principal et les autres sont soient des actions composées (**ACTION**) soient des fonctions de type quelconque (**FONCTION**).

> Les objets globaux sont déclarés dans le module principal.

> La communication entre les modules se fait via les paramètres et les variables globales.

> Les paramètres peuvent être de n'importe quel type.

> Quatre types standard sont autorisés : **ENTIER**, **BOOLEAN**, **CAR**, **CHAINE** .

> Le langage est l'ensemble des algorithmes abstraits, écrits à base de modèles (machines abstraites).

> On définit ainsi des machines de Turing : Machine-caractères et Machine-nombres permettant l'initiation à l'algorithmique.

> On définit également des machines abstraites sur les structures, les vecteurs et les listes permettant l'initiation aux structures élémentaires de données.

> Le langage peut être étendu avec d'autres machines abstraites.

> Le langage est doté des opérations de haut niveau permettant d'initialiser des vecteurs ou de construire des listes à partir d'un ensemble de valeurs (scalaires ou structures) et d'initialiser les machines de Turing

> Le format des instructions est libre.

Structure d'un Z-algorithme

SOIENT
    { Objets locaux et globaux }
    { Annonce des modules }
DEBUT
    { Instructions }
FIN

Module 1
Module 2
....
Module n

Chaque module peut être soit une fonction soit une action.

Définition d'une action

ACTION Nom (P1, P2, ..., Pn)

    { Objets locaux et paramètres }
DEBUT

    { Instructions }
FIN

Définition d'une fonction

FONCTION Nom (P1, P2, ...,Pn) : Type
    { Objets locaux et paramètres }
DEBUT
    { Instructions }
FIN

Type peut être quelconque.

Exemple d'un Z-algorithme

```
SOIENT
    Mot : CHAINE;
    C : CAR;

DEBUT

    CREER_MCAR(M, ['  Jhh  Jsthd Lkql ifd ']);
    LIRECAR(M, C);
    TANTQUE C <> '.'

        TQ (C=' ') ET (C <> '.')
            LIRECAR(M, C)
        FTQ   ;

        Mot := '';

        TQ (C <> ' ') ET (C <> '.')
            Mot := Mot + C ;
            LIRECAR(M, C)
        FTQ;

        SI Mot <> ''  ECRIRE(Mot) FSI

    FINTANTQUE
FIN
```

Objets

Les objets peuvent être des scalaires :**ENTIER**, **BOOLEEN, CAR**, **CHAINE**

Exemples :

A, B, C **DES BOOLEENS** ;
I, J : **ENTIER ;**
Ch **UNE CHAINE** ;
C **UN CAR** ;

Les objets peuvent être des machines abstraites : **STRUCTURE** (Structures), **VECTEUR**(Vecteurs) et **LISTE**(Listes linéaires chaînées)

Exemples

I, J, K **DES ENTIERS** ;
L1, L2 **DES LISTES DE CHAINES** ;
A **UNE STRUCTURE (CHAINE, ENTIER**) ;
V1 **UN VECTEUR**(10, 60) ;
V2 **UNE STRUCTURE (ENTIER, BOOLEEN) ;**
F **UN FICHIER DE (ENTIER, VECTEUR**(10))
ENTETE ENTIER BUFFER BUF1, BUF2 ;

Expressions

Comme dans les langages de programmation.
Exemples :
(B+C) / F , **NON** Trouv, (X # 5) **ET NON** Trouv, F(x) <> 5

Instructions

V désigne une variable, E une expression et Idf un nom de module.

[ ] désigne une partie facultative, { } un ensemble.

Affectation	:	V := E
Lecture	:	**LIRE**(V1, V2, ...)
Ecriture	:	**ECRIRE**(E1, E2, ...)
Conditionnelle	:	**SI** E [:] { Instructions } [**SINON** { Instructions }] **FSI**
Appel	:	**APPEL** Idf [(E1, E2, ...)]
Répétitive (Forme 1)	:	**TQ** E [:] { Instructions } **FTQ**
Répétitive (Forme 2)	:	**POUR** V := E1, E2,E3 { Instructions } **FPOUR**

E3 désigne le pas.

Opérations liées aux machines abstraites

Machine-caractères

LIRECAR, NBRCAR

Machine-nombres

LIRENOMBRE, NBRNOMBRE

Listes linéaires chaînées

ALLOUER , LIBERER , VALEUR, SUIVANT, AFF_ADR, AFF_VAL

Vecteurs

ELEMENT, AFF_ELEMENT

Structures

STRUCT, **AFF_STRUCT**

Fichiers

OUVRIR, FERMER, ENTETE, AFF_ENTETE, LIRESEQ, LIREDIR, ECRIRESEQ, ECRIREDIR, RAJOUTER, ALLOC-BLOC, FINFICH

Opérations de haut niveau

CREER_MCAR,
CREER_MNOMBRE,
CREER_LISTE,
INIT_STRUCT,
INIT_VECTEUR(ou **INIT_TABLEAU)**

Exemple:

CREER-LISTE (L, [12, 23, 67, I, I+J])
Crée la liste linéaire chaînée L avec les valeurs entre crochets dans l'ordre indiqué.