

Langage Z

(Niveau 2)

DESCRIPTION

Généralités

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

> Le langage Z admet les modules récurifs.

> Les objets globaux sont déclarés dans le module principal.

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

> Le langage permet

- tout type de paramètres : scalaires, structures, liste, file, vecteurs, pile, arbres et même les type complexes.

- l'allocation dynamique de tableaux et de structures

- l'affectation globale de tout type

> Quatre types standard sont autorisés : **ENTIER**, **BOOLEEN**, **CAR**, **CHaine**.

> Certaines fonctions usuelles sont prédéfinies : **MOD**, **MAX** et **MIN**.

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

> On définit ainsi des machines abstraites sur les objets composés (structures), les vecteurs de dimension quelconque, les piles, les files d'attente, les arbres de recherche binaire, les arbres de recherche m-aire, les listes mono-directionnelles, les listes bidirectionnelles.

> Les vecteurs et les structures peuvent être statiques ou dynamiques.

> Le langage permet les types composés du genre **PILE DE FILES DE LISTES DE** dont la dernière citée est de type scalaires ou structure.

> Le langage est doté des opérations de haut niveau permettant de construire des listes, des arbres, des files d'attente, etc. à partir d'un ensemble de valeurs (expressions) ou de structures.

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

Exemple d'un Z-algorithme

SOIENT

L1, L2 **DES LISTES**;

Rech, Tous : **FONCTION(BOOLEEN)**;

DEBUT

CREER_LISTE(L1, [2, 5, 9, 8, 3, 6]);

CREER_LISTE(L2, [12, 5, 19, 8, 3, 6, 2, 9]);

ECRIRE(Tous(L1, L2))

FIN

FONCTION Rech (L, Val) : **BOOLEEN**

SOIENT

L **UNE LISTE**; Val **UN ENTIER**;

DEBUT

SI L = NIL : Rech := **FAUX**

SINON

SI VALEUR(L) = Val

Rech := **VERAI**

SINON

Rech := Rech(**SUIVANT**(L), Val)

FSI

FSI

FIN

FONCTION Tous (L1, L2) : **BOOLEEN**

SOIENT

L1, L2 **DES LISTES**;

DEBUT

SI L1 = NIL

Tous := **VERAI**

SINON

SI NON Rech(L2, VALEUR(L1))

Tous := **FAUX**

SINON

Tous := Tous(**SUIVANT**(L1), L2)

FSI

FSI

FIN

Objets

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

Les objets peuvent être des machines abstraites : **FILE**(Files d'attente) **PILE**, **STRUCTURE**, **VECTEUR**, **LISTE**, **LISTEBI**(Listes bidirectionnelles), **ARB**(Arbres de recherche binaire), **ARM** (Arbres de recherche m-aire), **FICHER**.

Les objets peuvent être complexes, c'est-à-dire une combinaison de machines abstraites.

Exemples

A, B, C DES BOOLEENS ;
B UNE CHAINE ;
L1, L2 DES LISTES ;
A UNE STRUCTURE(CHAINE, ENTIER) ;
V1 UN VECTEUR(10, 60) DE (CAR, ENTIER) ;
Y UNE LISTE DE PILES DE VECTEUR(10)
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) \diamond 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,)

Appel : **APPEL** Idf [(E1, E2, ...)]
 Conditionnelle : **SI** E [:]
 { Instructions }
 [**SINON**
 { Instructions }]

Répétitive : **FSI**
(Forme 1) **TQE** [:]
FTQ { Instructions }

Répétitive : **POUR** V := E1, E2, E3
(Forme 2) { Instructions }
FPOUR

E3 désigne le pas.

Opérations liées aux machines abstraites

**Listes : ALLOUER, LIBERER, VALEUR,
SUIVANT, AFF_ADR, AFF_VAL**

Listes bidirectionnelles : ALLOUER, LIBERER, VALEUR, SUIVANT, AFF_VAL, PRECEDENT, AFF_ADRD, AFF_ADRG

Piles : CREERPILE, EMPILER, DEPIILER, PILEVIDE

Files : CREERFILE, ENFILER, DEFILER, FILEVIDE.

Arbres de recherche binaire : CREERNOEUD, FG, FD, PERE, LIBERERNOEUD, AFF_FG, AFF_FD, AFF_PERE, INFO, AFF_INFO

Arbres de recherche m-aire : CREERNOEUD, FILS, LIBERERNOEUD, AFF_FILS, INFOR, AFF_INFO, DEGRE, AFF_DEGRE, PERE, AFF PERE

Vecteurs : ELEMENT, AFF_ELEMENT.
ALLOC_TAB, LIBER_TAB (Si tableau
dynamique)

Structures : STRUCT, AFF_STRUCT
 ALLOC_STRUCT, LIBER_STRUCT (si structure
 dynamique)

Fichiers

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

Opérations de haut niveau

**CREER_ARB,CREER_LISTE,
CREER_LISTEBI,CREER_ARM,
CREER_PILE,CREER_FILE,
INIT_VECTEUR(ou INIT_TABLEAU)
INIT_STRUCT**

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.