

Z Language (Level 2)

DESCRIPTION

Overview

A Z algorithm is a set of modules. The first one is the main module and the others are either actions (**ACTION**) or functions (**FUNCTION**).

> The Z language accepts recursion.

> Static objects are declared in the main module.

> The communication between modules is made via parameters and static variables.

> The language allows:

- Any type of parameters : scalars, structures, linked lists , queues, stacks, arrays, trees and also complex types.

- The dynamic allocation of arrays and structures

- The global assignment of any type

> Four standard types (scalars) are allowed : **INTEGER, BOOLEAN, CHAR, STRING .**

> Some usual functions exist : **MOD, MAX, MIN, RANDNUMBER, RANDSTRING,...**

> The langage is the set of abstract algorithms written by using abstract machines.

> So, we consider abstract machines on structures, arrays of any dimension, queues, stacks, binary and m-ary search trees, linked lists and bidirectional linked lists.

> We also consider an abstract machine on the files allowing their use and the construction of simple structures of files as well as the most complex structures.

> The language allows compound types such as **STACK OF QUEUE OF LISTS OFOF** which the last one quoted is of scalar type or simple structure.

> The language has high-level operations to build lists, trees, queues, etc. from a set of values (expressions) or structures.

> the language offers two very useful functions to randomly generate strings (**ALEACHAINE**) and integers (**ALEANUMBER**).

> The language allows reading and writing scalars, n-dimensional arrays of scalars and simple or complex structures.

Structure of a Z-algorithm

LET

{ Local and statitc objects }

{ announcement of the modules }

BEGIN

{ Statements }

END

Module 1

....

Module n

Each module can be either a function or an action.

Definition of an action

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

{ Local objects and parameters }

BEGIN

{ Statements }

END

Calling an action is made by the operation **CALL** followed by the name of the action and its parameters. Parameters are not protected by the action.

Définition d'une fonction

FUNCTION Name (P1, P2, ..., Pn) : Type

{ Local objects and parameters }

BEGIN

{ Statements }

END

Type can be any.

Functions are used in expressions.

Parameters are not protected by the function.

Example of a Z-algorithm

LET

L1, L2 : **LISTS**;

Rech, Tous : **FUNCTION(BOOLEAN)**;

BEGIN

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

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

WRITE(Tous(L1, L2))

END

FUNCTION Rech (L, Val) : **BOOLEAN**

LET

L : **LIST**; Val : **INTEGER**;

BEGIN

IF L = **NULL** : Rech := **FALSE**

ELSE

IF **CELL_VALUE**(L) = Val :

Rech := **TRUE**

ELSE

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

ENDIF

ENDIF

END

FUNCTION Tous (L1, L2) : **BOOLEAN**

LET

L1, L2 : **LISTS**;

BEGIN

IF L1 = **NULL** : Tous := **TRUE**

ELSE

IF NOT Rech(L2, **CELL_VALUE**(L1))

Tous := **FALSE**

ELSE

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

ENDIF

ENDIF

END

