Khawarizm

D.E ZEGOUR

École Supérieure d'Informatique

ESI

Khawarizm

C O N T E N T S

Khawarizm

OVERVIEW

KHAWARIZM is an environment for learning and deepening the main data and file structures.

KHAWARIZM offers the possibility to write algorithms in an algorithmic language (Z-language), to indent them, to run or simulate them and to convert them automatically to the PASCAL and C programming languages.

Khawarizm

O V E R V I E W

Use of Z language

Writing algorithms on abstract machines simulating the main data structures.

Aims:

- Experimenting on the main data structures, regardless of their implementation, by developing algorithms on arrays, records, linked lists, doubly linked lists, queues, stacks, binary search trees, m-ary search trees.

- Creating and managing complex data structures including linked list of queues, linked list of stacks, tree of linked lists, linked list of stacks of arrays, etc.

Khawarizm

U S E

Familiarization with an arbitrary algorithmic language
Learn the algorithmic language used.
Use the help.

Editing the algorithm
Write an algorithm or correct an existing algorithm.

Syntax check
Repeat as long as there are errors
. Run the *Indent* module.
. Correct the errors

At this point, your algorithm is well written and has been indented for you. (You can change the presentation modes of your algorithm (see "Options" in the menu)

Khawarizm

U S E

Running

Start the execution of your algorithm

The windows then show
- the data read by your algorithm (Data button)
- the writings emitted by your algorithm (Results Button)

Your algorithm gives either the expected results or not. In this last case, launch the simulation to try to determine the logic errors.

Khawarizm

U S E

Simulation

Run the simulation of your algorithm.
This is an execution with a trace.

The windows show
- the data read by your algorithm (Data button)
- the writings issued by your algorithm (Results Button)
- all the changes made on the objects used (Simulation Button)

You thus have the complete trace of your algorithm that you can print and analyze to detect errors.

If you want to see more closely the different steps of your algorithm, ask for a trace.

Khawarizm

U S E

Trace

Request the simulation with trace again.

You can then follow step by step the evolution of your algorithm, exit the current loop or even the current module.

In order to avoid having a complete trace which can be long, it is possible to limit the length of the loops used in your algorithm.

You can change the simulation modes (see "Options" in the menu).

Khawarizm

U S E

Translation to a programming language

Once your algorithm is "running", it is possible to translate it automatically into PASCAL or C. Just click on the "To Pascal" or "To C" button.

Two windows organized as "Tiles" are then shown. One contains your algorithm and the other the result of the translation.

You can consult the help concerning the transition to PASCAL or C.

In this help, you will find
- The Z to PASCAL and Z to C equivalents.
- All implementations of Z machines.

Khawarizm's task stops here.

Khawarizm

U S E

PASCAL or C programming

Use the PASCAL or C compiler to finalize your program. In particular, you can add all the procedures of data entry and restitution of the results.

Khawarizm

Z LANGUAGE

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.

Khawarizm

Z LANGUAGE

Overview

> 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, ...

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

Khawarizm

Z LANGUAGE

Overview

> So, we consider abstract machines on structures, arrays of any dimension, queues, stacks, binary and M-ary search trees, linked lists and doubly 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 OF ....OF which the last one quoted is of scalar type or simple structure.

Khawarizm

Z LANGUAGE

<u>Overview</u>

> 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 (RANDSTRING) and integers (RANDNUMBER).

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

Khawarizm

Z LANGUAGE

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.

Khawarizm

Z LANGUAGE

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.

Definition of a function

```
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.

Khawarizm

Z LANGUAGE

Example of a Z algorithm

```
{ Is a linked list included in another ? }
LET
L1 , L2 : LISTS ;
Search , All : FUNCTION ( BOOLEAN ) ;
BEGIN
CREATE_LIST ( L1 , [ 2 , 5 , 9 , 8 , 3 , 6 ] ) ;
CREATE_LIST ( L2 , [ 12 , 5 , 19 , 8 , 3 , 6 , 2 , 9 ] ) ;
WRITE ( All ( L1 , L2 ) )
END
```

```
{ Search for a value in a linked list }
FUNCTION Search ( L , Val ) : BOOLEAN
LET
L : LIST ;
Val : INTEGER ;
BEGIN
IF L = NULL
Search := FALSE
ELSE
IF CELL_VALUE ( L ) = Val
Search := TRUE
ELSE
Search := Search ( NEXT ( L ) , Val )
ENDIF
ENDIF
END
```

```
{ Is L1 included in L2? }
FUNCTION All ( L1 , L2 ) : BOOLEAN
LET
L1 , L2 : LISTS ;
BEGIN
IF L1 = NULL
All := TRUE
ELSE
IF NOT Search ( L2 , CELL_VALUE ( L1 ) )
All := FALSE
ELSE
All := All ( NEXT ( L1 ) , L2 )
ENDIF
ENDIF
END
```

Khawarizm

Z LANGUAGE

Objects

Objects can be scalars : INTEGER, BOOLEAN, CHAR, STRING.

Objects can be abstract machines :
QUEUE, STACK,
STRUCTURE, ARRAY,
LIST, BILIST(Doubly linked lists),
BST(Binary search trees), MST(M-ary search trees),
FILE.

Objects can be complexes, i.e, a combination of abstract machines.

Khawarizm

Z LANGUAGE

Objects (Examples)

Scalars :

A, B, C : BOOLEANS ; Ch : STRING ;

Abstract machines :

A : BST;
L1, L2 : LISTS ;
A : STRUCTURE(STRING, INTEGER) ;
F : FILE OF (INTEGER, ARRAY(10)) HEADER INTEGER BUFFER BUF1, BUF2

Complex structures :

V1 :ARRAY(10, 60) OF (CHAR, INTEGER) ;
Y : LISTE OF STACKS OF ARRAYS10)

Khawarizm

Z LANGUAGE

<u>Z Expressions</u>

As in the programming languages.

Arithmetical expressions : + , - , / , *
Logical expressions : AND, OR, NOT
Expressions on strings : +
Relational expressions : <, <=, >, >=, =, <> (or #)

Logical constants : TRUE, FALSE
Pointer constant : NULL

Examples

B+C / F
NOT Found
(X # 5) AND NOT Found
F(X) <> 5
P = Null

Khawarizm

Z LANGUAGE

Statements

V denotes a variable, E an expression and Idf an identifier of an action or a function
[ ] denotes an optional part, { } a set.

```
Assignment:       V := E
Reading:          READ(V1, V2, .....)
Writing:          WRITE(E1, E2, .....)
Calling:          CALL Idf [ ( E1, E2, ...) ]
Conditionnal :    IF E [ : ]  { Statements}  [ ELSE { Statements} ] ENDIF
While Loop:       WH E [ : ]
                     { Statements}
                  EWH
For Loop :        FOR V := E1, E2,E3  // E3 denotes the step
                     { Statements}
                  ENDFOR
```

Khawarizm

Z LANGUAGE

Abstract Machines

Linked lists : ALLOCATE_CELL, FREE CELL_VALUE, NEXT, ASS_ADR, ASS_VAL

Doubly linked lists : ALLOCATE_CELL, FREE, CELL_VALUE, NEXT, ASS_VAL, PREVIOUS, ASS_L_ADR, ASS_R_ADR

Stacks : CREATESTACK, PUSH, POP, EMPTY_STACK

Queues : CREATEQUEUE, ENQUEUE, DEQUEUE, EMPTY_QUEUE

Binary search trees : ALLOCATE_NODE, LC, RC, PARENT, FREE_NODE, ASS_LC, ASS_RC, ASS_PARENT, NODE_VALUE, ASS_NODE_VAL

Khawarizm

Z LANGUAGE

Abstract Machines

M-ary search trees : ALLOCATE_NODE, CHILD, FREE_NODE, ASS_CHILD, NODE_VALUE_MST, ASS_NODE_VAL_MST, DEGREE, ASS_DEGREE, PARENT, ASS_PARENT

Arrays : ELEMENT, ASS_ELEMENT.
ALLOC_ARRAY, FREE_ARRAY (If dynamic array)

Structures : Struct, ASS_struct
ALLOC_STRUCT, FREE_STRUCT(If dynamic structure)

Files : OPEN, CLOSE, HEADER, ASS_HEADER,HEADSEQ,READDIR,WRITESEQ, WRITEDIR, ADD, ALLOC-BLOCK, ENDFILE

Khawarizm

Z LANGUAGE

<u>High level operations</u>

CREATE_LIST (L, [Exp1, Exp2, ...])
CREATE_BILIST (LB, [Exp1, Exp2, ...])
CREATE_BST (A, [Exp1, Exp2, ...])
CREATE_MST (M, [Exp1, Exp2, ...])
CREATE_QUEUE (F, [Exp1, Exp2, ...])
CREATE_STACK (P, [Exp1, Exp2, ...])
INIT_STRUCT(S, [Exp1, Exp2, ...])
INIT_ARRAY(T, [Exp1, Exp2, ...])

Examples
CREATE-LIST (L, [12, 23, 67, I, I+J])
creates the linked list L with values with the values in square brackets in the order shown.

Khawarizm

DOCUMENTATION

Download :

Khawarizm II^+ AFE (Win 64) : http://zegour.esi.dz/Ftp/Khawarism2_afe.zip

Khawarizm

D O C U M E N T A T I O N

Documentation Integrated into Khawarizm

Introduction (TXT)

Presentation (TXT)

Use (TXT)

Exposition about Khawarizm (HTML)

Z language description (HTML)