FUNCTION read\_size () : INTEGER

VAR

n : INTEGER;

BEGIN

REPEAT

Write ("Give a valid size positive and less than 100");

Read (n);

UNTIL (n >0 and n<100);

RETURN n ;

END

PROCEDURE read\_vectors(n:INTEGER, VAR tabF:ARRAY\_OF FLOAT, VAR tabS:tabF:ARRAY\_OF FLOAT)

VAR

i,j: INTEGER;

BEGIN

FOR i FROM 0 TO n-1 DO

Read(tabF[i]);

END\_FOR

FOR j FROM 0 TO n-1 DO

Read(tabS[j]);

END\_FOR

END

PROCEDURE dot\_product (n: INTEGER, vectorOne:ARRAY\_OF FLOAT,vectorTwo:ARRAY\_OF FLOAT, VAR ps: FLOAT)

VAR

i, j: INTEGER;

BEGIN

FOR i FROM 0 TO n-1 DO

ps:= ps + vectorOne[i]\*vectorTwo[i];

END

PROCEDURE result (ps: FLOAT)

BEGIN

IF (ps = 0) THEN

Write ("The two vectors are orthogonals");

ELSE

Write ("the two vectors are not orthogonals");

END\_IF

END

ALGORITHM orthogonal\_vectors

VAR

vectorOne : ARRAY\_OF FLOAT[100];

vectorTwo : ARRAY\_OF FLOAT[100];

i,n: INTEGER;

ps: FLOAT;

BEGIN

//Read size of elements

n:= read\_size();

//Read Elements

//sub algorithm reading elements of array

read\_vectors(n, vectorOne, vectorTwo);

//add successive elements

//sub algorithm dot\_prod (resolution of our problem)

dot\_product(n, vectorOne, vectorTwo, ps);

//display result

//sub algorithm displating result

result(ps);

END