Kitabın kodları

Fortran 95 Example program

! Input: An integer, List\_Len, where List\_Len is less

! than 100, followed by List\_Len-Integer values

! Output: The number of input values that are greater

! than the average of all input values

Implicit none

Integer Dimension(99) :: Int\_List

Integer :: List\_Len, Counter, Sum, Average, Result

Result= 0

Sum = 0

Read \*, List\_Len

If ((List\_Len > 0) .AND. (List\_Len < 100)) Then

! Read input data into an array and compute its sum

Do Counter = 1, List\_Len

Read \*, Int\_List(Counter)

Sum = Sum + Int\_List(Counter)

End Do

! Compute the average

Average = Sum / List\_Len

! Count the values that are greater than the average

Do Counter = 1, List\_Len

If (Int\_List(Counter) > Average) Then

Result = Result + 1

End If

End Do

! Print the result

Print \*, 'Number of values > Average is:', Result

Else

Print \*, 'Error - list length value is not legal'

End If

End Program Example

ALGOL 60

**comment** ALGOL 60 Example Program

Input: An integer, listlen, where listlen is less than

100, followed by listlen-integer values

Output: The number of input values that are greater than

the average of all the input values ;

**begin**

**integer array** intlist [1:99];

**integer** listlen, counter, sum, average, result;

sum := 0;

result := 0;

readint (listlen);

**if** (listlen > 0) ∧(listlen < 100) **then**

**begin**

**comment** Read input into an array and compute the average;

**for** counter := 1 **step** 1 **until** listlen **do**

**begin**

readint (intlist[counter]);

sum := sum + intlist[counter]

**end;**

**comment** Compute the average;

average := sum / listlen;

**comment** Count the input values that are > average;

**for** counter := 1 **step** 1 **until** listlen **do**

**if** intlist[counter] > average

**then** result := result + 1;

**comment** Print result;

printstring("The number of values > average is:");

printint (result)

**end**

**else**

printstring ("Error—input list length is not legal";

**end**

**BASIC**

REM BASIC Example Program

REM Input: An integer, listlen, where listlen is less

REM than 100, followed by listlen-integer values

REM Output: The number of input values that are greater

REM than the average of all input values

DIM intlist(99)

result = 0

sum = 0

INPUT listlen

IF listlen > 0 AND listlen < 100 THEN

REM Read input into an array and compute the sum

FOR counter = 1 TO listlen

INPUT intlist(counter)

sum = sum + intlist(counter)

NEXT counter

REM Compute the average

average = sum / listlen

REM Count the number of input values that are > average

FOR counter = 1 TO listlen

IF intlist(counter) > average

THEN result = result + 1

NEXT counter

REM Print the result

PRINT "The number of values that are > average is:";

result

ELSE

PRINT "Error—input list length is not legal"

END IF

END

/\* PL/I PROGRAM EXAMPLE

INPUT: AN INTEGER, LISTLEN, WHERE LISTLEN IS LESS THAN

100, FOLLOWED BY LISTLEN-INTEGER VALUES

OUTPUT: THE NUMBER OF INPUT VALUES THAT ARE GREATER THAN

THE AVERAGE OF ALL INPUT VALUES \*/

PLIEX: PROCEDURE OPTIONS (MAIN);

DECLARE INTLIST (1:99) FIXED.

DECLARE (LISTLEN, COUNTER, SUM, AVERAGE, RESULT) FIXED;

SUM = 0;

RESULT = 0;

GET LIST (LISTLEN);

IF (LISTLEN > 0) & (LISTLEN < 100) THEN

DO;

/\* READ INPUT DATA INTO AN ARRAY AND COMPUTE THE SUM \*/

DO COUNTER = 1 TO LISTLEN;

GET LIST (INTLIST (COUNTER));

SUM = SUM + INTLIST (COUNTER);

END;

/\* COMPUTE THE AVERAGE \*/

AVERAGE = SUM / LISTLEN;

/\* COUNT THE NUMBER OF VALUES THAT ARE > AVERAGE \*/

DO COUNTER = 1 TO LISTLEN;

IF INTLIST (COUNTER) > AVERAGE THEN

RESULT = RESULT + 1;

END;

/\* PRINT RESULT \*/

PUT SKIP LIST ('THE NUMBER OF VALUES > AVERAGE IS:');

PUT LIST (RESULT);

END;

ELSE

PUT SKIP LIST ('ERROR—INPUT LIST LENGTH IS ILLEGAL');

END PLIEX;

{Pascal Example Program

Input: An integer, listlen, where listlen is less than

100, followed by listlen-integer values

Output: The number of input values that are greater than

the average of all input values }

**program** pasex (input, output);

**type** intlisttype = **array** [1..99] **of** integer;

**var**

intlist : intlisttype;

listlen, counter, sum, average, result : integer;

**begin**

result := 0;

sum := 0;

readln (listlen);

**if** ((listlen > 0) **and** (listlen < 100)) **then**

**begin**

{ Read input into an array and compute the sum }

**for** counter := 1 **to** listlen **do**

**begin**

readln (intlist[counter]);

sum := sum + intlist[counter]

**end;**

{ Compute the average }

average := sum / listlen;

{ Count the number of input values that are > average }

**for** counter := 1 **to** listlen **do**

**if** (intlist[counter] > average) **then**

result := result + 1;

{ Print the result }

writeln ('The number of values > average is:',

result)

**end** { of the then clause of if (( listlen > 0 ... }

**else**

writeln ('Error—input list length is not legal')

**end.**

-- Ada Example Program

-- Input: An integer, List\_Len, where List\_Len is less

-- than 100, followed by List\_Len-integer values

-- Output: The number of input values that are greater

-- than the average of all input values

**with** Ada.Text\_IO, Ada.Integer.Text\_IO;

**use** Ada.Text\_IO, Ada.Integer.Text\_IO;

**procedure** Ada\_Ex **is**

**type** Int\_List\_Type **is array** (1..99) **of** Integer;

Int\_List : Int\_List\_Type;

List\_Len, Sum, Average, Result : Integer;

**begin**

Result:= 0;

Sum := 0;

Get (List\_Len);

**if** (List\_Len > 0) **and** (List\_Len < 100) **then**

-- Read input data into an array and compute the sum

**for** Counter := 1 .. List\_Len **loop**

Get (Int\_List(Counter));

Sum := Sum + Int\_List(Counter);

**end loop;**

-- Compute the average

Average := Sum / List\_Len;

-- Count the number of values that are > average

**for** Counter := 1 .. List\_Len **loop**

**if** Int\_List(Counter) > Average **then**

Result:= Result+ 1;

**end if;**

**end loop;**

-- Print result

Put ("The number of values > average is:");

Put (Result);

New\_Line;

**else**

Put\_Line ("Error—input list length is not legal");

**end if;**

**end** Ada\_Ex;