

FOR expression in ABAP

Version 1.0

Author – Prateek Shirod

Date - 04 May 2018

Purpose

This document covers how to use FOR expression in ABAP. This is high level document and assume resource has basic knowledge of ABAP.

As we all know, the In-line declarations, operators and expressions available in 7.4 SP02 onward are taking the abap world by storm for last few years. For the new or the experienced abap-ers this is almost like fall in line or fall apart. The main idea is to write code easily, keep it lean and compact as much as possible. Let us discuss about some simple best-case scenarios of FOR expression which can seriously reduce no of lines of code and make it more compact. I feel this is one of the best addition among the new features.

Situation 1: To move values from a source internal table to a target internal table which has all the fields of source internal table + some additional fields (say).

*Define structure

TYPES:

```
BEGIN OF ty_struct1,  
  field1 TYPE i,  
  field2 TYPE string,  
END OF ty_struct1,  
BEGIN OF ty_struct2,  
  field1 TYPE i,  
  field2 TYPE string,  
  field3 TYPE i,  
END OF ty_struct2.
```

*Define table types

```
TYPES: gtt_struct1 TYPE STANDARD TABLE OF ty_struct1 WITH DEFAULT KEY,  
       gtt_struct2 TYPE STANDARD TABLE OF ty_struct2 WITH DEFAULT KEY.
```

* Initialize source table with some random values

```
DATA(lt_source) = VALUE gtt_struct1(  
  ( field1 = 1 field2 = 'A' )  
  ( field1 = 2 field2 = 'B' ) ).
```

*Use like simple MOVE CORRESPONDING

```
DATA(lt_target1) = VALUE gtt_struct2( FOR lwa_source IN lt_source ( CORRESPONDING #(  
  lwa_source ) ) ).  
cl_demo_output=>display( lt_target1 ).
```

```

*Populate sy-tabix in the additional fields within the for loop
DATA(lt_target2) = VALUE gtt_struct2( FOR lwa_source IN lt_source
    INDEX INTO index
    LET base = VALUE ty_struct2( field3 = index )
    IN ( CORRESPONDING #( BASE ( base ) lwa_source ) ) ).
cl_demo_output=>display( lt_target2 ).

```

```

*Populate any value or call custom method in the additional fields within the for loop
DATA(lt_target3) = VALUE gtt_struct2( FOR lwa_source IN lt_source
    LET base = VALUE ty_struct2( field3 = 10 ) "<<< Custom method/any value
    IN ( CORRESPONDING #( BASE ( base ) lwa_source ) ) ).
cl_demo_output=>display( lt_target3 ).

```

Points to Note:

i) VALUE (Type of the internal table) is used for the in-line declaration of the internal table. Here structure and Type declaration is required before using VALUE operator. If you want to use VALUE # then type of the table must be already available and defined in the program i.e. the internal table need to be declared before.

iii) LT_TARGET3-FIELD3 is populated with a hard-coded value 10 but in real life a custom method call can also be done in this place to pass the actual value of field3

Situation 2: Suppose we have an internal table and we want to create another range table with the values of one field from that internal table

Sample code

```

*Get details from DB table
SELECT * FROM
sflight
INTO TABLE @DATA(lt_sflight)
WHERE connid IN (17,555).

```

IF sy-subrc = 0.

```

*Prepare a range table
DATA: lr_carrid TYPE RANGE OF s_carr_id.
lr_carrid = VALUE #( FOR ls_value IN lt_sflight ( sign = 'I'
    option = 'EQ'
    low = ls_value-carrid ) ).

```

```

SORT lr_carrid BY low.
DELETE ADJACENT DUPLICATES FROM lr_carrid
COMPARING low.
cl_demo_output=>display( lr_carrid ).
ENDIF.

```

Points to Note:

i) '@' need to be used in the SELECT for in-line declaration of the internal table

Situation 3: There are 2 tables and based on which a third table need to be constructed

*Define the structures of header & item table

TYPES:

```
BEGIN OF comp,  
  ebeln TYPE ebeln,  
  buhrs TYPE buhrs,  
END OF comp.
```

*Declare table type

```
TYPES: gtt_comp TYPE STANDARD TABLE OF comp WITH DEFAULT KEY,  
       gtt_header TYPE STANDARD TABLE OF ekko WITH DEFAULT KEY,  
       gtt_item TYPE SORTED TABLE OF ekpo WITH NON-UNIQUE KEY ebeln  
              WITH NON-UNIQUE SORTED KEY key_combi COMPONENTS ebeln.
```

*Populate dummy values

```
DATA(lt_header) = VALUE gtt_header(  
  ( ebeln = '4500000027' )  
  ( ebeln = '4500000028' )  
  ( ebeln = '4500000029' ) ).
```

```
DATA(lt_item) = VALUE gtt_item( ( ebeln = '4500000027' ebelp = '000010' netwr = '100' buhrs = 'IND'  
)  
  ( ebeln = '4500000027' ebelp = '000020' netwr = '200' buhrs = 'IND' )  
  ( ebeln = '4500000027' ebelp = '000030' netwr = '300' buhrs = 'IND' )  
  ( ebeln = '4500000028' ebelp = '000010' netwr = '999' buhrs = 'USA' )  
  ( ebeln = '4500000029' ebelp = '000010' netwr = '25' buhrs = 'GB' )  
  ( ebeln = '4500000029' ebelp = '000020' netwr = '50' buhrs = 'GB' )  
  ( ebeln = '4500000029' ebelp = '000030' netwr = '100' buhrs = 'GB' )  
  ( ebeln = '4500000029' ebelp = '000040' netwr = '150' buhrs = 'GB' ) ).
```

*Now populate the values of Comp from item table to a new table

```
DATA(lt_comp) = VALUE gtt_comp(  
  FOR ls_header IN lt_header  
  FOR ls_item IN lt_item WHERE ( ebeln = ls_header-ebeln )  
  ( ebeln = ls_header-ebeln  
    buhrs = ls_item-buhrs ) ).
```

```
SORT lt_comp BY ebeln.
```

```
DELETE ADJACENT DUPLICATES FROM lt_comp  
COMPARING ebeln.
```

```
cl_demo_output=>display( lt_comp ).
```

Points to note:

i) VALUE (Type of the internal table) is used for the in-line declaration of the internal table.

Situation 4: Summation of item values and populate the total in header table

⦿ Use the code from situation 3 for the declaration part

*Now sum up all the netwr for a purchase order

LOOP AT lt_comp ASSIGNING FIELD-SYMBOL(<lf_comp>).

```
  <lf_comp>-netwr = REDUCE netwr( INIT lv_netwr TYPE netwr
    FOR ls_item IN
    FILTER #( lt_item
    USING KEY key_combi
    WHERE ebeln = <lf_comp>-ebeln )
    NEXT lv_netwr = lv_netwr + ls_item-netwr ).
```

ENDLOOP.

cl_demo_output=>display(lt_comp).

Points to note:

i) LT_ITEM must be sorted table for using the FILTER option

ii) REDUCE with FOR is like a nested loop only difference is that you can't debug all the iterations

Situation 5: Append new values into an internal table which already have some values

Use the code from situation 3 for the declaration part

⦿*Append new records in item table

```
lt_item = VALUE #( BASE lt_item
  ( ebeln = '4500000030' ebelp = '000010' netwr = '1099' bukrs = 'CAN' ) ).
```

cl_demo_output=>display(lt_item).

Result

SAP ABAP Development, SAP ABAP Guides, SAP ABAP Live

Points to Note:

i) Here VALUE with # has been used since type of LT_ITEM is already defined