Former Member

September 5, 2017 | 4 minute read

Dynamic Programming in ABAP – Part 1 – Introduction to Field Symbols

Field symbol is a placeholder for data object, which points to the value present at the memory address of a data object. It does not reserve any physical memory space when we declare them. It only points to a data object at run time. Field symbols are of two types:

- Typed Field Symbol
- Generic Field Symbol

Typed Field Symbol – Typed field symbol can be declared as:

```
DATA: var TYPE i VALUE 2.
FIELD-SYMBOLS: <fs_num> TYPE i.
```

```
ASSIGN var TO <fs_num>.
WRITE: / <fs_num>.
<fs_num> = 4.
WRITE: / var.
```

The output will be 2 and 4.

NOTE:

- Typed field symbols can point to the data objects of specified type only.
- After assigning a data object to a field symbol, if we make any changes to the field symbol value, then the value of corresponding data object is also updated.

Field symbol as a replacement of Work area:

Modifying internal table records – We can declare a field symbol of type any structure, which we can use while looping through an internal table.

```
DATA: lt_mara TYPE STANDARD TABLE OF mara.
FIELD-SYMBOLS: <fs_mara> TYPE mara.
SELECT * FROM mara INTO TABLE lt_mara UP TO 10 ROWS.
LOOP AT lt_mara ASSIGNING <fs_mara>.
  <fs_mara>-matkl = 'DEMO'.
ENDLOOP.
```

NOTE:

- If we change any field of structure in field symbol, the corresponding field in internal will get updated. We don't need to write the **MODIFY** statement which we would have written if we had used work area. This is because work area stores a copy of the internal table row, whereas field symbol directly references the internal table row.

- Hence processing of internal table with field symbol is faster than the processing of internal table with work area.

Appending to internal table – Now suppose we want to append some values to one internal table, then we can use field symbol as below:

```
DATA: lt_mara TYPE STANDARD TABLE OF mara.
FIELD-SYMBOLS: <fs_mara> TYPE mara.

APPEND INITIAL LINE TO lt_mara ASSIGNING <fs_mara>.
IF <fs_mara> IS ASSIGNED.
  <fs_mara>-matnr = 'MAT1'.
  <fs_mara>-matkl = '001'.
  UNASSIGN <fs_mara>.
ENDIF.

APPEND INITIAL LINE TO lt_mara ASSIGNING <fs_mara>.
IF <fs_mara> IS ASSIGNED.
  <fs_mara>-matnr = 'MAT2'.
  <fs_mara>-matkl = '001'.
  UNASSIGN <fs_mara>.
ENDIF.
```

After executing this, the internal table will hold two rows.

NOTE:

- Always perform a check on field symbol that if it is assigned before doing any operation to avoid short dump. Also after doing the operation, unassign the field symbol.

Reading internal table – We can read a record of internal table using field symbol as below:

```
READ TABLE lt_mara ASSIGNING <fs_mara> WITH KEY matnr = 'MAT1'.
```

Generic Field Symbol:

Dynamic programming is actually implemented using generic field symbols. The most commonly used generic types are **TYPE ANY** and **TYPE ANY TABLE**.

```
FIELD-SYMBOLS: <fs_str> TYPE ANY.
FIELD-SYMBOLS: <fs_tab> TYPE ANY TABLE.
```

Here we can assign any data object to **TYPE ANY** field symbol whereas **TYPE ANY TABLE** field symbol is used for assigning any internal table.

TYPE ANY:

Let us assign a work area of type **MARA** to a TYPE ANY field symbol and then populate the work area using field symbol.

```
FIELD-SYMBOLS: <fs_str> TYPE ANY.
FIELD-SYMBOLS: <fs_data> TYPE ANY.
DATA: lw_mara TYPE mara.

ASSIGN lw_mara TO <fs_str>.
IF <fs_str> IS ASSIGNED.
  ASSIGN COMPONENT 'MATNR' OF STRUCTURE <fs_str> TO <fs_data>.
  IF <fs_data> IS ASSIGNED.
    <fs_data> = 'MAT001'.
    UNASSIGN <fs_data>.
  ENDIF.
  UNASSIGN <fs_str>.
ENDIF.
```

NOTE:

- After assigning **lw_mara** to **<fs_str>**, we cannot directly use the ‘-‘ operator on field symbol to access the fields of MARA structure i.e. **<fs_str>-matnr** would produce syntax error. This is because the field symbol type is declared only at runtime not at compile time.
- So to access the **matnr** field with field symbol, first we need to assign that field component to a different field symbol and then use the new field symbol to update the **matnr** field as show in above code snippet.
- After execution of above code snippet, the value of **lw_mara-matnr** would be **MAT001**.

TYPE ANY TABLE:

We can assign any internal table to this field symbol. Let us analyze the below code snippet to understand how we could use such field symbol.

```
FIELD-SYMBOLS: <fs_tab> TYPE ANY TABLE.
FIELD-SYMBOLS: <fs_str> TYPE any.
FIELD-SYMBOLS: <fs_data> TYPE any.
DATA: lt_mara TYPE STANDARD TABLE OF mara.
DATA: lw_mara TYPE mara.

ASSIGN lt_mara TO <fs_tab>.
SELECT * FROM mara INTO TABLE lt_mara UP TO 10 ROWS.

LOOP AT <fs_tab> ASSIGNING <fs_str>.
  IF <fs_str> IS ASSIGNED.
    ASSIGN COMPONENT 'MATKL' OF STRUCTURE <fs_str> TO <fs_data>.
    IF <fs_data> IS ASSIGNED.
```

```
      IF <fs_data> EQ '01'.
*********** Do some processing *********
      ENDIF.
      UNASSIGN <fs_data>.
    ENDIF.
  ENDIF.
ENDLOOP.
```

Reading internal table using generic field symbol:

```
FIELD-SYMBOLS: <fs_tab> TYPE ANY TABLE.
FIELD-SYMBOLS: <fs_str> TYPE any.
DATA: lt_mara TYPE STANDARD TABLE OF mara.

ASSIGN lt_mara TO <fs_tab>.
SELECT * FROM mara INTO TABLE lt_mara UP TO 10 ROWS.

READ TABLE <fs_tab> ASSIGNING <fs_str> WITH KEY ('MATNR') = 'MAT001'.
```

NOTE:

- Since **<fs_tab>** is a generic field symbol, its type will be known only at runtime, hence we cannot directly write the fields of MARA structure after WITH KEY, instead we have to write the field name within parenthesis as shown above.
- In ABAP, this parenthesis indicates the compiler that the value of the operand will be decided at runtime, hence we don't get any compilation error.

In my next blog i have explained about data references and its significance in dynamic programming. Below is the link for same.

https://blogs.sap.com/2017/09/11/dynamic-programming-in-abap-part-2-introduction-to-data-reference/

Alert Moderator

Assigned Tags

ABAP Development

SAP NetWeaver Application Server for ABAP

abap

dynamic

field

field symbol

introduction

View more...

Similar Blog Posts

New kinds of ABAP expressions, Part II

By Kilian Kilger Mar 28, 2022

Creating Dynamic Table and Dynamic Select

By Dogu Ozan Kumru Aug 17, 2022

New kinds of ABAP expressions

By Kilian Kilger Oct 19, 2021

Related Questions

dynamic internal table

By Former Member Jul 28, 2007

How to read calculation from field symbol in ABAP.

By Amanjot Kaur Oct 21, 2018

loop using field symbols

By Pavan Prasad Feb 10, 2020

20 Comments

You must be Logged on to comment or reply to a post.

Matthew Billingham

September 5, 2017 at 7:43 am

I'm not a fan of prefixes related to typing generally, but having FS before each field symbol seems particularly useless. The names are embedded in angle brackets - we know it's an FS without the need for any prefix. Concentrate on meaningful names rather than prefixes that add zero value.

Like 18 | Share

Bhakti joshi

April 3, 2020 at 7:05 am

its for readability.we shud not invent new things just for the heck of it. readability in a program is very important for the portability and management of code.

Like 0 | Share

Sandra Rossi
April 3, 2020 at 8:14 am

What “should not invent new things” means, here?

Like 1 | Share

Bhakti joshi
April 3, 2020 at 5:03 pm

i mean the idea of ditching fs_ from the name of field symbol is against universally accepted standards and its better not to divert bcz it might reduce readability and clarity in complex coding.

Like 0 | Share

Sandra Rossi
April 3, 2020 at 5:08 pm

I've been telling developers to not use <FS_...> for 20 years. I think all the projects I was working in did not prefix them with FS. Sorry, it's not a universal rule. I don't know who invented this rule of prefixing field symbols with FS. I think those people who invented the prefix FS for field symbols were crazy 😉

Like 3 | Share

Matthew Billingham
April 3, 2020 at 6:09 pm

"i mean the idea of ditching fs_ from the name of field symbol is against universally accepted standards"

With the greatest possible respect - this is not the case. But just to be sure I checked - twice - and it turns out - I am right. It really isn't the case..

I've been programming in ABAP since 1997. I've also defined the standards for numerous multinationals. Never was <fs_...> permitted. In code reviews, the code would be sent back and the programmer told not to be so silly. Precisely because clarity and readability are so important.

I've also managed to get the current SAP and DSAG recommendations for variable naming included at a couple of global companies. These recommendations follow the general consensus across the *entire* IT current programming community that clarity and meaningful names are vital to readability, **and prefixes reduce clearly and readability and so should be avoided**.

Do you also prefix forms with **F_**

Like 4 | Share

Bhakti joshi

April 3, 2020 at 7:48 pm

Thanks for sharing ur experience. I've always seen field symbols with fs_ in more than 15 years of experience seeing multitude of global MNC systems across geographies . If there are any std sap blogs / document / examples without this convention then it will be good to refer. Maybe within scn site itself . tat is in case it's worth spending any more time. thanks

Like 1 | Share

Sandra Rossi

April 4, 2020 at 6:25 am

Avoid prefixes (documentation "Clean ABAP", by SAP): https://github.com/SAP/styleguides/blob/master/clean-abap/CleanABAP.md#avoid-encodings-esp-hungarian-notation-and-prefixes

Like 3 | Share

Patrick Van Nierop
April 4, 2020 at 10:19 am

Personally I don't even consider the 'fs_' part to be a prefix. It's completely useless in my eyes as the '<' & '>' clearly indicate that you're looking at a field-symbol.

A prefix in a field-symbol would be adding a 's_', 't_',.. etc to the name of the field-symbol. At the very least that would add some meaning :).

Also, with regards to the Clean ABAP styleguide, no prefixes works if you adhere to a functional style of programming (smaller 'functions' that do only one thing). Sadly, I still see a lot off wall-of-text code, with both global & local variables.. And anyway, I prefer to see meaningful names.. so rather LV_COMPANY_CODE than LV_BUKRS or gods forbid BUKRS ;).

Last but not least, for a lot of us, it is the customer that decides upon the naming conventions, and it is not that easy to convince them to change these..

Like 1 | Share

Matthew Billingham
April 5, 2020 at 5:15 pm

Or lv_bukrs for a global variable... (But I only use it locally within methods... ?)

Like 0 | Share

Matthew Billingham
April 5, 2020 at 5:17 pm

> *I've always seen field symbols with fs_*

This is exactly how wrong practices get perpetuated. Nobody challenges the status quo, nobody thinks. And then it becomes the standard - and it's still wrong.

Like 4 | Share

Ryan Crosby

September 13, 2022 at 12:25 pm

Spot on... I challenge it all the time because that's the road to continuous improvement. One of my least favorite sayings of all time... "That's the way we've always done it".

Like 0 | Share

Jacques Nomssi
September 5, 2017 at 8:09 pm

Hello Rahul,

1. there is a static and a dynamic version of the ASSIGN statement. If you recognize a dynamic case, then check of SY-SUBRC after ASSIGN (0 if ok, 4 if error) is enough to make sure the field symbol valid, so a IS ASSIGNED or subsequent UNASSIGN is not mandatory.
2. On newer releases, typed field symbols can be declared implicitely, e.g.

```
ASSIGN lw_mara TO FIELD-SYMBOL(<lw_mara>).
```

I think the VALUE operator will make the use of ASSIGNING as in the idiom APPEND INITIAL LINE TO .. ASSIGNING less popular.

regards,

JNN

Like 5 | Share

Former Member | Blog Post Author
September 6, 2017 at 2:18 am

Hi Jacques,

Thanks for putting new points.

Like 0 | Share

Tamit Kumar Das Sharma
September 7, 2017 at 6:29 pm

Also Rahul, you can refer to this quick reference to understand the latest advancements in ABAP – https://blogs.sap.com/2015/10/25/abap-740-quick-reference/

The full documentation is available at https://help.sap.com/doc/abapdocu_740_index_htm/7.40/en-US/index.htm

With these advancements programming is fun.

Like 2 | Share

Former Member | Blog Post Author
September 8, 2017 at 7:57 am

Thanks Tamit

Like 0 | Share

Gopi Srinivasan
February 2, 2018 at 1:16 pm

Good to begin with field symbols..! Rahul. Cheers.

Like 1 | Share

Francesco Allocca
July 2, 2020 at 9:29 am

I ask for information,

rather than

```
FIELD-SYMBOLS: <fs_str> TYPE ANY.
FIELD-SYMBOLS: <fs_data> TYPE ANY.
DATA: lw_mara TYPE mara.

ASSIGN lw_mara TO <fs_str>.
IF <fs_str> IS ASSIGNED.
  ASSIGN COMPONENT 'MATNR' OF STRUCTURE <fs_str> TO <fs_data>.
```

Isn't this better?

```
FIELD-SYMBOLS: <fs_str> TYPE ANY.
DATA: lw_mara TYPE mara.
ASSIGN lw_mara-matnr TO <fs_str>.
```

Like 0 | Share

Paul Hammond
February 16, 2022 at 6:39 pm

I still don't see what the advantage of using field symbols is.

For example, instead of

```
DATA: lt_mara TYPE STANDARD TABLE OF mara.
FIELD-SYMBOLS: <fs_mara> TYPE mara.
```

```
APPEND INITIAL LINE TO lt_mara ASSIGNING <fs_mara>.
IF <fs_mara> IS ASSIGNED.
  <fs_mara>-matnr = 'MAT1'.
  <fs_mara>-matkl = '001'.
  UNASSIGN <fs_mara>.
ENDIF.
```

what's wrong with

```
DATA: lt_mara TYPE STANDARD TABLE OF mara,
      foo TYPE mara.

foo-matnr = 'MAT1'.
foo-matkl = '001'.
APPEND foo TO lt_mara.
```

Like 1 | Share

Vera Kulish
February 16, 2022 at 8:11 pm

Paul,

I like your example and agree to use that simple code instead of that confusing code with assigning and unassing field symbol.

Like 0 | Share