Former Member

September 11, 2017 | 4 minute read

Dynamic Programming in ABAP – Part 2 – Introduction to Data Reference

Hi,

In my last blog I explained about field symbols, below is the link for same:

https://blogs.sap.com/2017/09/05/dynamic-programming-in-abap-part-1-introduction-to-field-symbols/

In this blog I am going to explain about data references and its significance in dynamic programming.

According to SAP documentation, ***Data references can point to any data objects or to their parts (components, rows of internal tables, or sections specified by offsets and lengths).***

So data references are nothing but pointers. It stores the memory address of any data object. But to access the actual data object which data reference is pointing to, we first need to deference it using dereferencing operator **->***.

Difference between field symbol and data reference:

Field symbol is a placeholder for data object to which it is assigned and points to the content of data object hence it can be used at any operand position (no need to dereference it) and works with the content of the referenced memory area (value semantics).

Data references are pointers to data objects and it contains the memory address of data object (reference semantics). Data reference cannot be used at operand position directly; it should be dereferenced first.

Working with data reference:

There can be two types of data references:

- Typed Data Reference
- Generic Data Reference

Typed Data Reference:

Typed data reference variable can be declared as:

```
DATA lr_num TYPE REF TO i.
CREATE DATA lr_num.
```

Here first statement declares a reference variable **lr_num** which can point to any data object of type **"i"**. And second statement creates an anonymous data object of type **"i"** and assigns the reference of this data object to **lr_num**. Now if we

want to change the value of data object, then it can be done by dereferencing **lr_num** by using dereference operator **->*** as shown below:

```
DATA lr_num TYPE REF TO i.
CREATE DATA lr_num.

lr_num->* = 2.
WRITE: / lr_num->*.
```

The output will be 2.

NOTE:

- With ABAP 7.40, instead of **CREATE DATA**, the **NEW** operator can also be used to create an anonymous data object and assigns its reference to a data reference variable.

```
DATA lr_num TYPE REF TO i.
lr_num = NEW #( ).
```

Assigning existing data object to data reference:

If you want to assign the reference of an existing data object to a data reference, you can use **GET REFERENCE** statement.

```
DATA: lv_num TYPE i VALUE 2.
DATA: lr_num TYPE REF TO i.

GET REFERENCE OF lv_num INTO lr_num.
lr_num->* = 4.
WRITE: / lv_num.
```

Here **lv_num** is an existing data object (not anonymous data object). The output would be 4.

NOTE:

- With ABAP 7.40, instead of **GET REFERENCE**, the **REF** operator also can be used to assign the reference of an existing data object to a data reference.

Working with structures:

```
DATA: lr_mara TYPE REF TO mara.

CREATE DATA lr_mara.
lr_mara->matnr = '1111'.
lr_mara->matkl = '03'.
```

Here individual components of the structure can be accessed with **->** operator on data reference variable.

Working with internal tables:

While processing internal table row, we can use **REFERENCE INTO** statement to set references to table rows as shown below:

```
DATA: lr_mara TYPE REF TO mara.
DATA: lt_mara TYPE TABLE OF mara.

SELECT * FROM mara INTO TABLE lt_mara UP TO 10 ROWS.

LOOP AT lt_mara REFERENCE INTO lr_mara.
  WRITE: / lr_mara->matnr.
ENDLOOP.
```

Generic Data Reference:

Generic data reference can be declared as:

```
DATA: lr_num TYPE REF TO data.
CREATE DATA lr_num TYPE i.
```

Here first statement declares a generic data reference **lr_num** which can point to any data object. And second statement creates an anonymous data object of type "i" and assigns its reference to **lr_num**.

'**data**' in ABAP is a generic data type.

Now since **lr_num** is generic, **lr_num->*** cannot be directly used at operand position. Hence the below statement would not be allowed.

```
lr_num->* = 2.
```

So in case of generic data reference, it can only be dereferenced using a field symbol, and this field symbol can be used at any operand position to manipulate the value of data object as shown below:

```
DATA: lr_num TYPE REF TO data.
FIELD-SYMBOLS: <num> TYPE any.

CREATE DATA lr_num TYPE i.
ASSIGN lr_num->* TO <num>.

<num> = 3.
```

NOTE:

- After **ASSIGN** statement you should check **sy-subrc** If field symbol assignment is successful, **sy-subrc** will be 0 otherwise it will be 4.

Working with structures:

```
DATA: lr_str TYPE REF TO data.
FIELD-SYMBOLS: <str> TYPE any.
FIELD-SYMBOLS: <data> TYPE any.
CREATE DATA lr_str TYPE mara.

ASSIGN lr_str->* TO <str>.
ASSIGN COMPONENT 'MATNR' OF STRUCTURE <str> TO <data>.
<data> = '112'.
```

Here **CREATE DATA** statement creates an anonymous data object (MARA structure) and assigns its reference to the generic data reference **lr_str**, which then can be dereferenced into a generic field symbol **<str>**. Now, to access individual component of MARA structure, **ASSIGN COMPONENT** statement can be used.

Dynamically create data objects:

Requirement: Selection screen parameter "**Table Name**" will take a table name as input and display the corresponding table entries as output.

Solution:

```
PARAMETERS: p_tname TYPE tabname.
DATA: lr_tab TYPE REF TO data.
FIELD-SYMBOLS: <tab> TYPE ANY TABLE.

CREATE DATA lr_tab TYPE TABLE OF (p_tname).
ASSIGN lr_tab->* TO <tab>.
IF sy-subrc EQ 0.
  SELECT * FROM (p_tname) INTO TABLE <tab> UP TO 10 ROWS.
```

```
  cl_demo_output=>display( <tab> ).
ENDIF.
```

Explanation:

Here **lr_tab** is a generic data reference and **<tab>** is a generic field symbol for internal table. In **CREATE DATA** statement, the type of data object is mentioned in parenthesis which means that the type will be determined at runtime based on the value of parameter **p_tname**. After that we have dereferenced the data reference **lr_tab** into a generic field symbol **<tab>.** Now this field symbol can be used to do any valid operation on the internal table.

Difference between data reference and object reference:

There are two types of reference variable:

- Data reference and
- Object reference

Data reference variable can store the reference to any data object (variable, structures, internal tables etc.) whereas Object reference variable can store the reference to any class object.

For data reference variables, either the generic data type or a completely specified data type can be specified. For object reference variables, either a class or an interface can be specified.

Thank you for reading.

In next blog, I have put one example of dynamic programming approach:

https://blogs.sap.com/2017/09/29/dynamic-programming-in-abap-part-3-an-example-abap-rtts/

Credits:

https://help.sap.com/http.svc/rc/abapdocu_751_index_htm/7.51/en-US/abendata_reference_type.htm

Alert Moderator

Assigned Tags

ABAP Development

SAP NetWeaver Application Server for ABAP

abap

data

data reference

dynamic

introduction

View more...

Similar Blog Posts

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

By Former Member Sep 05, 2017

Evolution of ABAP

By Karl Kessler Sep 01, 2022

New kinds of ABAP expressions, Part II

By Kilian Kilger Mar 28, 2022

Related Questions

---

dynamic internal table

By Former Member Jul 28, 2007

OOP

By Former Member Jan 31, 2008

Difference between Normal ALV and ALV OOPs

By Former Member Dec 05, 2007

18 Comments

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

Jelena Perfiljeva
September 11, 2017 at 3:39 pm

Can I ask you - what exactly is the point of these blogs?

I have quite a bit of ABAP experience but, to be honest, after reading this I'm none wiser on why would I need this if I'm not planning to re-invent SE16. If someone has trouble understanding ABAP Help, I doubt this would be of much help to them either.

Also posting the old syntax and then adding a note of 7.4 syntax could have been justified few years ago when it just became available. But these days I believe it should be the other way around: post current syntax and make a note of the old one. This is, however, all available in ABAP Help so, again, not sure what the point of this blog is...

Like 1 | Share

Former Member | Blog Post Author
September 11, 2017 at 6:05 pm

Hi Jelena,

I observed that sometimes, ABAP help lacks proper examples, so anyone who is very new in ABAP development may not be very comfortable in understanding ABAP help. Please correct me if i am wrong.

My point in this blog was to help the readers by showing that dynamic programming in ABAP is achieved using field symbols and data reference by putting some examples. Also the table display problem was just an example to show how to create data object dynamically.

I agree that i should have used 7.4 syntax, will keep this in mind in my future posts, thanks for your guidance 🙂

Regards,

Rahul

Like 1 | Share

Jelena Perfiljeva
September 12, 2017 at 5:39 pm

This may have been true many years ago but currently ABAP keyword documentation is available online and it has very good examples and thorough explanations, from what I see. Unfortunately, I can't find how to get a link to a specific article but this is a general link for documentation and a sample screenshot:

Syntax

```
CREATE DATA dref [area_handle].
```

Effect

If neither of the additions TYPE or LIKE is specified, the data reference variable dref mus

Example

Creates an internal table and a data object of type i. The data objects are created directly
objects are accessed by dereferencing the data references.

```
TYPES t_itab TYPE TABLE OF i WITH NON-UNIQUE KEY table_line.

DATA: tab_ref TYPE REF TO t_itab,
      i_ref   TYPE REF TO i.

IF tab_ref IS INITIAL.
   CREATE DATA tab_ref.
ENDIF.

tab_ref->* = VALUE #( FOR j = 1 UNTIL j > 10 ( j ) ).
```

If someone is using ABAP Help in their outdated SAP system I'd encourage them to use online documentation to supplement it.

Like 0 | Share

Michelle Crapo

September 19, 2017 at 5:59 pm

Another nice t-code for help is ABAPDOCU. It comes complete with documented program examples.

Like 2 | Share

John Voorhees
July 23, 2019 at 10:15 pm

Jelena, So I strongly disagree with you!! I am a new ABAPer. After working with several other languages C++, Java, PHP and BPMS.

While I can read the documentation which can be verbose , I found this blog and his one on Field Symbols most useful to get up to speed quickly

Keep it up! 🙂

Like 4 | Share

Jay Desai
November 1, 2019 at 7:43 am

@Jelena - Humbly disagree.

SAP help language and examples can be tougher to understand for beginners. This is much more simpler and i benefited from it.

Thanks Rahul for the blog.

Like 2 | Share

Jelena Perfiljeva
November 1, 2019 at 4:06 pm

It's OK to disagree, different people - different opinions.

Yes, there are some ABAP Help (not to be confused with SAP Help) articles that are difficult to understand. But in this particular case personally I find ABAP documentation and examples quite adequate and language in this blog doesn't seem any simpler to understand to me (maybe I'm stupid, dunno 🙂). And I'm scratching my head on what **exactly** causes difficulties for the beginners in this case. So far the comments just state disagreement but it'd be nice to offer some specific examples, like what exactly is not clear in ABAP Help. (CC Horst Keller)

Like 0 | Share

Saptarshi Ray Chaudhuri

July 31, 2021 at 10:28 pm

"So in case of generic data reference, it can **only** be dereferenced using a field symbol, and this field symbol can be used at any operand position to manipulate the value of data object as shown below" - could not find this clarity in the help documentation .

Also this single blog post summarizes data referencing and dereferencing in a more easy to understand manner than the help documentation, placing the concept in a single page.

Like 1 | Share

Jelena Perfiljeva

August 2, 2021 at 5:03 pm

This is stated in the documentation for TYPE REF TO data:

"A data reference variable typed generically with TYPE REF TO data can be dereferenced only in the statement ASSIGN with the dereferencing operator ->."*

https://help.sap.com/doc/abapdocu_753_index_htm/7.53/en-US/index.htm?file=abaptypes_references.htm

If information in this blog helps you to understand something, that's great. But please always check the dates on these blogs (this one is 4 years old) and make sure to refer to the current documentation for the release you're working on to get most accurate information.

Like 0 | Share

Jacques Nomssi
September 12, 2017 at 8:37 am

Hello Rahul,

you can pass references to routines/methods as IMPORTING parameters, and still be able to change the referenced object. In this use case the IS BOUND predicate expression is useful to check if a data reference is valid.

best regards,

JNN

Like 1 | Share

Former Member | Blog Post Author
September 12, 2017 at 8:48 am

Correct Jacques, I forgot to put this point.

We should always check if the data reference is valid using IS BOUND predicate, just like we use IS ASSIGNED for validating field symbol.

Thanks.

Like 0 | Share

Michelle Crapo
September 19, 2017 at 6:30 pm

Hi Rahul,

Again this is a nice start. I know that a lot of us are still on the older systems. (SAP GUI) So I tend to look for these "older" blogs. With that thought - it would help if this read less like a index from a book or a help doc. Perhaps a nice program to illustrate your points would be helpful instead of the small snippets. Show that one program that "encapsulates" what you are trying to do. I know you've written a dynamic program. Show it! Then maybe in pieces explain it.

Think more about what the business problem/demand caused you to write a dynamic program. Why? Easier to understand? Quicker to code? Multiple uses?

This really reads more like a help document.

Just a thought!

Michelle

Do Keep blogging! Very few are brave enough to use the "older" technology and blog about it.

Like 3 | Share

Former Member | Blog Post Author
September 20, 2017 at 6:48 am

Thanks Michelle for your suggestions, I will try to put a good example demonstrating dynamic programming in next blog.

Regards,

Rahul

Like 0 | Share

Angel Sabogal
September 4, 2019 at 10:21 pm

It's the best explanation I've ever seen. To be honest I prefer this snippets that seeking in a program the way to understand the topic. Thanks for posting.

Like 0 | Share

Sandra Rossi
September 5, 2019 at 7:25 am

Thank you for blogging about this concept often not well understood, and to summary the important notions all in one.

Just to nitpicking:

- It's always good to propose the link to the "texts referenced", so that to help readers learn more about the topic -> ABAP docu 7.53 - Data References ("*Data references can point to any data objects or to their parts (components, rows of internal tables, or sections specified by offsets and lengths).*")
- Typo: deference -> dereference
- "Data reference cannot be used at operand position directly": not entirely true if taken out of the context of your post, because you may pass a data reference as a parameter without dereferencing it (more frequent with object references/okay I admit you're only talking about data references), and the procedure will dereference it later.
- other possible syntax: dref = NEW i(555).
- "After **ASSIGN** statement you should check **sy-subrc**": no, it is set only for the "dynamic variants" of ASSIGN. ASSIGN dref->*, ASSIGN dobj, etc., don't set SY-SUBRC. But ASSIGN (var), ASSIGN COMPONENT, etc., do set SY-SUBRC.
- ABAP docu 7.53 - ASSIGN
- ABAP docu 7.53 - CREATE DATA
- ABAP docu 7.53 - NEW
- ABAP docu 7.53 - REF

Like 3 | Share

Sriram Kompell
March 20, 2021 at 7:10 am

Hi Rahul,

Excellent Blog indeed. Though there are several new syntaxes which has come up with SAP ABAP the business logic remains the same in SAP. Though we don't use new syntaxes as mentioned by Jelena Perfiljeva still we can write few blogs which are quite useful as per our requirements. We work for different customers for different scenario's i don't think all the requirements would be one and the same. in such cases our Blogs will be much useful for some one who are desperately looking for such scenario's . i wrote a blog for Inbound Delivery creation previously [Which you can't find any solution anywhere] and didn't used New ABAP Syntax. But still there are hundreds of users which found that useful. Jelena Perfiljeva though it useless and asked me the same question what's the use of that blog for her. i may be wrong here. but My intention is to help some one and share the knowledge according to our experience. so in this particular scenario we don't need to worry about new ABAP Syntax or other stuff, just that we're writing a blog which is useful for some needy.

Once Again Excellent Blog here. Much Appreciated.

Regards

Ram

Like 1 | Share

Jelena Perfiljeva
August 2, 2021 at 5:25 pm

Just reading this comment because it mentioned my old account. It seems your account is now inactive but for the record, I never made such derogatory comments on your blog. Anyone can read the actual comment text here together with your response. I'm quite surprised to read this statement here because your response in the blog comment was completely different.

Like 0 | Share

Shivam Bedwal

November 20, 2021 at 10:07 am

Hello Experts,

Can someone please explain, how would the following statement be written with "NEW" keyword -

```
CREATE DATA lr_tab TYPE TABLE OF (p_tname)
```

I understand that the below statement would be incorrect -

lr_tab = NEW (ptab)( ).

Thanks & Regards,

Shivam

Like 0 | Share

Find us on

Privacy	Terms of Use
Legal Disclosure	Copyright
Trademark	Cookie Preferences
Newsletter	Support