

## Contents

<b>WRITE statement</b> .....	2
<b>Syntax:</b> .....	2
<b>Examples:</b> .....	2
<b>Write statement additions</b> .....	2
<b>New line</b> .....	2
<b>Chain Operator ':' and Separator ','</b> .....	2
<b>Output location and size</b> .....	3
<b>Data Types and Data Objects</b> .....	3
<b>Elementary Data Types and Variables</b> .....	3
<b>Built in Data type:</b> .....	4
<b>Declare Variables using Built in Types</b> .....	4
<b>Declare Variables using Data Elements</b> .....	5
<b>Declare Variables using Table Fields</b> .....	6
<b>Declare Variables using User Define Types</b> .....	7
<b>Structured Types and Work Areas:</b> .....	7
<b>Declare Work area based on Database Table</b> .....	8
<b>Declare Work area based on Dictionary Structure</b> .....	8
<b>Structured data types and data objects</b> .....	9
<b>Table Types and Internal Tables</b> .....	10
<b>Internal tables</b> .....	10
<b>Types of internal tables:</b> .....	10
<b>Operations on Internal tables</b> .....	11
<b>Examples</b> .....	11

## WRITE statement

This statement is used to print static text or content of variable on screen. It can also be used to copy the content of one variable to another variable.

### Syntax:

```
WRITE {[AT] [/] [pos]/[(len/*/**)]} dobj  
    [/UNDER other_dobj]  
    [/NO-GAP]  
    [/int format options]  
    [/ext format options]  
    [/list elements]  
    [/QUICKINFO info].
```

### Examples:

<pre>REPORT zdemo_write_01. WRITE 'SAP ABAP Demo'. WRITE 'Welcome to ABAP World'.</pre>	<b>Output</b> <div>Demo on WRITE Statement</div> <div>SAP ABAP Demo Welcome to ABAP World</div>
Write the Same Program using Chain Operator <pre>WRITE: 'SAP ABAP Demo',       'Welcome to ABAP World'.</pre>	<b>Output</b> <div>Demo on WRITE Statement</div> <div>SAP ABAP Demo Welcome to ABAP World</div>

## Write statement additions

### New line

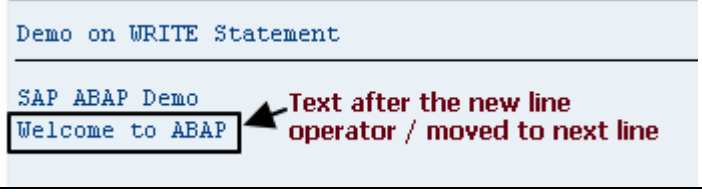
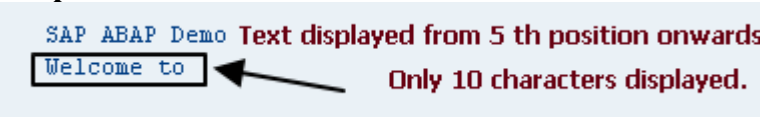
'/' character initiates output on next line.

### Chain Operator ':' and Separator ','

Chain operator is used to combine several similar statements which have same prefix. In the above example WRITE statement. So we can combine above two statements.

## Output location and size

We can specify location from where output should start printing and length of the output.

Output printed on different lines.  <pre>REPORT zdemo_write_02. WRITE 'SAP ABAP Demo'. WRITE / 'Welcome to ABAP'.</pre>	<b>Output</b>  
<pre>WRITE 5 'SAP ABAP Demo'. WRITE /5(10) 'Welcome to ABAP'.</pre>	<b>Output</b>  

## Data Types and Data Objects

**Data Types** defines technical attributes of data. Whereas **Data Objects** which are declared with help of Data Types

Based on its complexity we can classify into following types

- Elementary Data Types
- Structured Types
- Table Types

## Elementary Data Types and Variables

Elementary data types are used to declare variables which hold single value. There different types of Elementary Types

- Built In Data Type
- Data Elements
- Table Fields
- User Defined Types

### Built in Data type:

SAP has given some data types called as built in data types.

Type	Length	Standard length	Description
B	1 Byte		1 byte integer (internal)
C	1 to 65,535 characters	1 character	Text field
Cursor	as i	As i	Database cursor
D	8 characters		Date field
F	8 bytes		Floating point number
I	4 bytes		4 byte integer
N	1 to 65,535 characters	1 character	Numeric text
P	1 to 16 bytes	8 bytes	Packed number
String	Variable		Text string
S	2 bytes		2 byte integer (internal)
T	6 characters		Time field
X	1 to 65,535 bytes	1 byte	Byte field
xstring	Variable		Byte string

### Declare Variables using Built in Types

#### Program

#### Output

##### Employee Details

```
Employee No:      101
Employee Name:    Mr Apple
Employee Salary:  5.000,00
Employee Address: 1-9176/A, Building#60, Rose Street
```

```
" Declare Variables
DATA emp_num TYPE i.
DATA emp_name TYPE c LENGTH 30.
DATA emp_sal TYPE p DECIMALS 2.
DATA emp_addr TYPE c LENGTH 100.
```

Declared 4 Variables  
with Different technical  
Properties

```
" Populate Variables
emp_num = 101.
emp_name = 'Mr Apple'.
emp_sal = '5000.00'.
emp_addr = '1-9176/A, Building#60, Rose Street'.
```

Populate Values to the  
variables

```
" Display Variables
WRITE: 'Employee Details'.
WRITE: / (30) sy-uline.
WRITE: /2 (20) 'Employee No:', 22 emp_num.
WRITE: /2 (20) 'Employee Name:', 22 emp_name.
WRITE: /2 (20) 'Employee Salary:', 22 emp_sal.
WRITE: /2 (20) 'Employee Address:', 22 emp_addr.
```

Output Display

## Declare Variables using Data Elements

```
" Declare Variables
DATA salesorder TYPE vbeln.
DATA customer TYPE kunnr.
DATA amount TYPE netwr.
DATA curr TYPE waerk.
DATA product TYPE matnr.
DATA quantity TYPE menge_d.
DATA uom TYPE meins_d.
```

Data Elements

```
" Populate Variables
salesorder = 101.
customer = 'Mr Apple'.
amount = '5000.00'.
curr = 'USD'.
product = 'MT50981'.
quantity = '499.99'.
uom = 'EA'.
```

```
" Display Variables
WRITE: 'Sales Order Details'.
WRITE: / (30) sy-uline.
WRITE: /2 (20) 'Sales Order:', 22 salesorder.
WRITE: /2 (20) 'Customer:', 22 customer.
WRITE: /2 (20) 'Amount:', 22 amount, curr.
WRITE: /2 (20) 'Product:', product.
WRITE: /2 (20) 'Quantity:', quantity, uom.
```

## Output

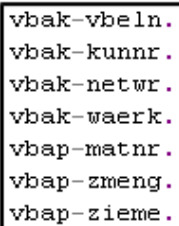
### Sales Order Details

Sales Order:	101		
Customer:	Mr Apple		
Amount:		5.000,00	USD
Product:	MT50981		
Quantity:		499,990	EA

## Declare Variables using Table Fields

*" Declare Variables*

```
DATA salesorder TYPE vbak-vbeln.  
DATA customer   TYPE vbak-kunnr.  
DATA amount     TYPE vbak-netwr.  
DATA curr       TYPE vbak-waerk.  
DATA product    TYPE vbap-matnr.  
DATA quantity   TYPE vbap-zmeng.  
DATA uom        TYPE vbap-zieme.
```



*" Populate Variables*

```
salesorder = 101.  
customer   = 'Mr Apple'.  
amount     = '5000.00'.  
curr       = 'USD'.  
product    = 'MT50981'.  
quantity   = '499.99'.  
uom        = 'EA'.
```

Fields from the Table.  
Technical properties of the  
table field will be used

*" Display Variables*

```
WRITE: 'Sales Order Details'.  
WRITE: / (30) sy-uline.  
WRITE: /2 (20) 'Sales Order:', 22 salesorder.  
WRITE: /2 (20) 'Customer:', 22 customer.  
WRITE: /2 (20) 'Amount:', 22 amount, curr.  
WRITE: /2 (20) 'Prodcuct:', product.  
WRITE: /2 (20) 'Quantity:', quantity, uom.
```

## Output

### Sales Order Details

Sales Order:	101		
Customer:	Mr Apple		
Amount:		5.000,00	USD
Prodcuct:	MT50981		
Quantity:		499,990	EA

## Declare Variables using User Define Types

We can declare our own data types using **TYPES** statement. Structures, Tables, Data elements defined in Data Dictionary will be used as data types to declare variables.

```
" Declare Type
TYPES t_name TYPE c LENGTH 30.
```

```
" Declare Variable
DATA emp1 TYPE t_name.
DATA emp2 TYPE t_name.
DATA emp3 TYPE t_name.
DATA emp4 TYPE t_name.
```

```
" Populate
emp1 = 'Mr Apple'.
emp2 = 'Mr Red'.
emp3 = 'Mr While'.
emp4 = 'Mr Blue'.
```

Variables declared  
with user defined  
type

```
"Display
WRITE 'Employees'.
WRITE / (15) sy-uline.
WRITE / emp1.
WRITE / emp2.
WRITE / emp3.
WRITE / emp4.
```

### Output

Employees

Mr Apple  
Mr Red  
Mr While  
Mr Blue

## Structured Types and Work Areas:

Structured data types are composed of several simple data types or other structure data types are used to declare work areas which holds group of values

Following are the Structure Types used to declare work areas

- Tables
- Dictionary Structures
- User Defined Structures

Declare Work area based on Database Table

Output

Customer Details			
Customer No:	CUST-01		
Customer Name:	Mr.	Red	White
City:	NewYork		

```
" Declare Workarea
DATA wa_customer TYPE kna1.

"Populate Values
wa_customer-kunnr = 'CUST-01'.
wa_customer-anred = 'Mr.'.
wa_customer-name1 = 'Red'.
wa_customer-name2 = 'White'.
wa_customer-ort01 = 'NewYork'.
" Display
WRITE 'Customer Details'.
WRITE / (20) sy-uline.
WRITE: /2 (15) 'Customer No:',
        20 wa_customer-kunnr.
WRITE: /2 (15) 'Customer Name:',
        20 wa_customer-anred,
        25 wa_customer-name1,
        32 wa_customer-name2.
WRITE: /2 (15) 'City:',
        20 wa_customer-ort01.
```

Declared work area with DB Table

Declare Work area based on Dictionary Structure

Output

Customer Details			
Customer Name:	Mr.	Red	White
PO Box:	12198		



```
" Declare Workarea
DATA wa_customer TYPE si_kna1.
```

Declare Workarea  
with Dictionary  
Structure

```
"Populate Values
wa_customer-anred = 'Mr.'.
wa_customer-name3 = 'Red'.
wa_customer-name4 = 'White'.
wa_customer-pfach = '12198'.

" Display
WRITE 'Customer Details'.
WRITE / (20) sy-uline.
WRITE: /2 (15) 'Customer Name:',
        20 wa_customer-anred,
        25 wa_customer-name3,
        32 wa_customer-name4.
WRITE: /2 (15) 'PO Box:',
        20 wa_customer-pfach.
```

## Structured data types and data objects

### Syntax

```
TYPES : BEGIN OF <struct_type>,
        <field_1> type <type_1>,
        <field_2> type <type_2>,

        <field_n> type <type_n>,
END OF <struct_type>
```

Above is the user defined structured data type, which used to declare multiple structured data objects

### Output

#### Customer Details

Customer No:	CUST-01	
Customer Name:	Mr. Red	White
City:	NewYork	

<pre> " Declare Workarea TYPES: BEGIN OF ty_customer,     kunnr TYPE knal-kunnr,     anred TYPE knal-anred,     name1 TYPE knal-name1,     name2 TYPE knal-name2,     ort01 TYPE knal-ort01, END OF ty_customer. DATA wa_customer TYPE ty_customer. "Populate Values wa_customer-kunnr = 'CUST-01'. wa_customer-anred = 'Mr.'. wa_customer-name1 = 'Red'. wa_customer-name2 = 'White'. wa_customer-ort01 = 'NewYork'. " Display WRITE 'Customer Details'. WRITE /(20) sy-uline. WRITE: /2(15) 'Customer No:',         20 wa_customer-kunnr. WRITE: /2(15) 'Customer Name:',         20 wa_customer-anred,         25 wa_customer-name1,         32 wa_customer-name2. WRITE: /2(15) 'City:',         20 wa_customer-ort01. </pre>	<p>Declare work area type / Local structure type TY_CUSTOMER and declare work area with Local work area type</p>
---	--

## Table Types and Internal Tables

### Internal tables

Internal tables are multi line structured types/objects. We can store several lines of similar structured data in an internal table. While defining internal table, we can specify some characteristics, so that it behaves in that way. We can specify key (Unique/Non Unique) for internal table. We can declare internal table by referring ABAP dictionary structure, Local structure or type pool structures.

### Types of internal tables:

- Standard - Most frequently use
- Sorted - data will be stored in sorted order so no need to sort again
- Hashed -

## Operations on Internal tables

### Filling

- Append/ Append lines of
- Insert / Insert lines of
- Collect

### Retrieving

- Loop - loop through entire internal table record by record
- Read - search internal table

### Delete

- Clear/Refresh - delete all records without specifying any condition
- Delete - delete records based on condition

### Sort

Sort internal table

### Describe

Count number of lines in internal table

## Examples

### Declare Internal Table

Dictionary Structure and Table	<pre>" Declare Internal Table Based on Dictionary Table/Structure DATA it_material1 TYPE STANDARD TABLE OF mara.</pre> <p><b>Declare Table Type and Declare Internal Table</b></p> <pre>" Declare Internal Table based on Local Table Type TYPES ty_material_tab TYPE STANDARD TABLE OF mara. DATA it_material2 TYPE ty_material_tab.</pre>
Internal Table based on Local Structure Type and Internal Table Type	

	<div><div><pre>" Declare Internal Table Based on User Defined Work Area Type TYPES:   BEGIN OF ty_material,     matnr TYPE matnr,     mtart TYPE mtart,     mbrsh TYPE mbrsh,     matkl TYPE matkl,   END OF ty_material. DATA it_material1 TYPE STANDARD TABLE OF ty_material.  " Declare Internal Table based on Local Table Type TYPES ty_material_tab TYPE STANDARD TABLE OF ty_material. DATA it_material2 TYPE ty_material_tab.</pre></div><div><div>Local Work area Type Declaration</div><div>Local Internal Table Type Declaration</div></div></div>
Fill Internal Table	
Append Statement	<div><div><pre>" Declare Work Area and Internal Table DATA wa_material TYPE ty_material. DATA it_material TYPE STANDARD TABLE OF ty_material.  " Fill Work area and Add Record to the Table wa_material-matnr = 'MAT1'. wa_material-mtart = 'HALB'. wa_material-mbrsh = 'M'. wa_material-matkl = '100'. APPEND wa_material TO it_material. CLEAR wa_material.  " Fill Work area and Add Record to the Table wa_material-matnr = 'MAT2'. wa_material-mtart = 'FERT'. wa_material-mbrsh = 'I'. wa_material-matkl = '101'. APPEND wa_material TO it_material. CLEAR wa_material.</pre></div><div><div>- Fill Workarea - Append workarea to Internal Table - Clear Workarea</div></div></div>
Insert Statement	

	<pre> " Declare Work Area and Internal Table DATA wa_material TYPE ty_material. DATA it_material TYPE STANDARD TABLE OF ty_material.  " Fill Work area and Add Record to the Table wa_material-matnr = 'MAT1'. wa_material-mtart = 'HALB'. wa_material-mbrsh = 'M'. wa_material-matk1 = '100'. INSERT wa_material INTO it_material INDEX 1. CLEAR wa_material.  " Fill Work area and Add Record to the Table wa_material-matnr = 'MAT2'. wa_material-mtart = 'FERT'. wa_material-mbrsh = 'I'. wa_material-matk1 = '101'. INSERT wa_material INTO it_material INDEX 2. CLEAR wa_material. </pre> <div> - Fill Workarea  - Insert workarea to Internal Table  - Clear workarea </div>
Collect Statement	<pre> " Declare Workarea Type TYPES:   BEGIN OF ty_stock,     matnr TYPE matnr,     werks TYPE werks_d,     labst TYPE labst,   END OF ty_stock. " Declare Table Type TYPES ty_stock_tab TYPE STANDARD TABLE OF ty_stock. " Declare Work Area and Internal Table DATA wa_stock TYPE ty_stock. DATA it_stock TYPE ty_stock_tab. " Fill Work area and Add Record to the Table wa_stock-matnr = 'MAT1'. wa_stock-werks = '1000'. wa_stock-labst = 200. COLLECT wa_stock INTO it_stock. CLEAR wa_stock.  " Fill Work area and Add Record to the Table wa_stock-matnr = 'MAT1'. wa_stock-werks = '1000'. wa_stock-labst = 200. COLLECT wa_stock INTO it_stock. CLEAR wa_stock. </pre> <div> - Fill Workarea  - Collect workarea to Internal Table  - Clear Workarea </div>

### *Read Internal Table*

- Read Single Record
  - By Index
  - By Key
- Loop Throug all the Records

### Read Record By Index

```
" Read Record By Index
READ TABLE it_material INTO wa_material INDEX 1.
IF sy-subrc = 0.
    WRITE: 'Material', 20 wa_material-matnr,
           / 'Material Type', 20 wa_material-mtart,
           / 'Industry Sector', 20 wa_material-mbrsh,
           / 'Material Group', 20 wa_material-matk1.
ENDIF.
```

### Read By Key

```
" Read Record By Key
READ TABLE it_material INTO wa_material WITH KEY matnr = 'MAT1'.
] IF sy-subrc = 0.
    WRITE: 'Material', 20 wa_material-matnr,
           / 'Material Type', 20 wa_material-mtart,
           / 'Industry Sector', 20 wa_material-mbrsh,
           / 'Material Group', 20 wa_material-matk1.
ENDIF.
```

### Existence Check

```
" Existence Check
READ TABLE it_material TRANSPORTING NO FIELDS WITH KEY matnr = 'MAT1'.
] IF sy-subrc = 0.
    WRITE / 'Material Exists'.
ENDIF.
```

### Binary Search

### *Looping Internal Table*

```
" Loop each record in the Internal Table
WRITE: /2(20) 'Material', 23(10) 'Material Type',
       34(7) 'IndSect', 42(10) 'Material Group'.
] LOOP AT it_material INTO wa_material.
    WRITE: /2(10) wa_material-matnr,
           23(10) wa_material-mtart,
           34(7) wa_material-mbrsh,
           42(10) wa_material-matk1.
ENDIF.
```