Title : ABAP/4 @ A GLANCE

Compiled by : Ramani N, *ennar91@yahoo.com*

I N D E X

1 **Working With Programming Elements**

0 TCODE: **SE38**

1 Data Types: n , F, P, D, T, X, String
Data: out type p decimals 2 value '100,20'.
Move 10 to a.
Add a to b.
Multiply a by c
Divide b by a
Subtract a from c

2 DATA: A(5) TYPE I/C/STRING.

3 WRITE : / 10(3) 'Anna University', 25 'Student'.
WRITE at 10

4 System Date/Time: sy-datum, sy-useit
Write: '222222' USING EDIT MASK '__:__:__'.

Some Important String Comparisons
CO Contains Only
CN Contains Not only
CA Contains Any
NA contains Not Any
CS Contains String
NS contains No String
CP Contains Pattern
NP contains No Pattern

WRITE: <var> NO-GAP
LEFT-JUSTIFIED
RIGHT JUSTIFIED
CENTERED
CURRENCY c
ROUND r
DECIMALS d
'DD/MM/YYYY'
Where: c='INR': r=3: d =1

STRLEN(<Var>) Split <var> at delimeter into A B.
CONCATENATE <c1> ... <cn> INTO <c> [SEPARATED BY <s>].
CONDENSE <c> [NO-GAPS].
When you use CO,CN, CA, NA, CS, NS, CP, and NP, offset values are assigned to SY-FDPOS depending on the search result.
SEARCH … FOR … sets SY-FDPOS to the offset of the search string.

MATHEMATICAL FUNCTIONS:

CEIL : Smallest integer value not smaller than the argument.
FLOOR: Largest integer value not larger than the argument.
TRUNC: Integer part of argument.
FRAC: Fraction part of argument.

FLOATING POINT FUNCTIONS:

ACOS, ASIN, ATAN; COS, SIN, TAN Trigonometric functions.
COSH, SINH, TANH : Hyperbolic functions.
EXP : Exponential function with base e (e=2.7182818285).
LOG : Natural logarithm with base e.
LOG10: Logarithm with base 10.
SQRT: Square root.

1a MESSAGE..(SE91) I -> Information: S->Status: E->Error: A->Abend: X->Exit

2 ABAP Control Structures

1 CASE….WHEN….ENDCASE
2 DO <N> TIMES. ENDDO.
3 DO. ENDDO. { Control to broken by using EXIT}
4 WHILE <expression>……ENDDO.
5 IF…(exit)....ENDIF.
6 LOOP AT <ITAB>….ENDLOOP. *Use: sy-subrc, sy-Tablx*
7 CHECK <expr>. {If FAILS, Rest of the programme won't be processed}

3 Events of Ordinary Reports

1 Initialization
2 Start-of-Selection
3 At Selection-Screen
4 At Selection-Screen Output
5 At Selection-Screen on <Field>
6 At User-Command.

---

4 Macros

```
Define cal.    Out = &1 &2  &3.  Write out.
End-of-definition.
cal 12 + 32.
```

---

5 Include Programming

---

8 Structure…. Internal Table… Table Introduction.

```
Nested Structure.      Data: Begin of emp,   data sn type I,
                                             Data: Begin of Dob,
                                             Data day type I,
                                             end of dob,  end of emp.
```

---

6 SUBROUTINE

```
PERFORM <> using ….                    {Calling  Programme}

1   Form <>.  EndForm.
2   Form <> Using <v1>.   EndForm.
3   Form <> Using  value(v) type I.  EndForm.
4   Form <>  Using <v> Changing <v>.   Endform.
5   Form <>  Using  <Str>  structure  <wa>.   EndForm.
6   Form <> Table  <Itab>  structure  <wa>.    EndForm.
```

---

7 Fn. Modules (SE37)

---

9 Internal Table - Data Accessing (Also Refer: ANNEX-2 back side)

```
1   Data: itab like  <wa>  occurs 0  with header line.   Loop at itab…Write… Endloop.
2   Loop at ….. Endloop.
3   Wa-no =5.  Read table itab.
4   Read Tbl emp into wa  with key ename = <>
5   Read Tbl emp into wa  index 1.
6   Read Table Itab from wa.     (wa-eno = 3)
7   Sort <Itab>.
8   Append wa  to  Itab.
9   sy-subrc.
10  Clear WA              {Header data will be removed}
11  Free  (itab)          {Removes Header,Detail as well as DEALLOCATES memory}
12  Refresh  (itab)       {Removes Header and Detail Content only}
13  COLLECT               COLLECT <wa> into ITAB.
                          1 It checks for the Existance of the data given in the WorkArea
                            from the Body of ITAB.
                          2 If Not Exists.
                                  Appends Data into ITAB.  <valid for P,I,F types only>
                          3 If Exits.    Updates the data of the Changed Fields only

14  DESCRIBE              table itab LINES num.
                          table itab DECIMALS num.
                          table itab COMPONENTS num.
15  itab-eno = 2.  MODIFY  itab TRANSPORTING eno where name = 'rose'.
16  DELETE                If <>.   Delete itab.  Endif.
                          Delete itab index 2.
                          Delete itab where eno = 3.
                          Delete adjacent duplicates from  itab.
17  Insert [wa into ]  ITAB  [index id]
```

Compiled by: Ramani N

10 DATA EXTRACTS

```
Data: Rno Type I,   Name(20) type c.
FIELD-GROUPS: HEADER, myDetail.        Where Header is Mandatory
Rno = 1.  Name = 'DAS SOFTWARE'.
INSERT RNO    into  HEADER.
INSERT Name  into  myDETAIL.
Sort.
LOOP.
        At HEADER.    Write:  /  Rno.                EndAt.
        At MYDETAIL.  Write:  /  Name.               EndAt.
        At FIRST.     Write: / 'Ist Rec'.            EndAt.
        At LAST.      Write: 'Last Rec'.             EndAt.
        At NEW Rno.   Write: 'New Value Starts'.  EndAt.
ENDLOOP.
```

---

11 Dynamic User Inputs

```
Write:  as checkbox
PARAMETERS ... AS CHECKBOX | RADIOBUTTON GROUP ... USER-COMMAND

   PARAMETERS: A AS CHECKBOX,
PARAMETERS <p> ..... RADIOBUTTON GROUP <radi>.....
SELECT-OPTIONS:  <fld>   for   <object-fld>  NO INTERVAL  NO-EXTENSION
                                (Refer: ANNEX-2 <back side>)
```

---

12 Data Dictionary

Domain, Data Element, Table, Structure, Search Help

---

13 DB Programming

1 Client Dependant/ independent [MANDT field]

MANDT: If thie field is added in our zTable, our table is called **Client Dependant**
Because, the data in one client can't be accessed from other client.
Such, Client Dependant data can be accessed from other client as follows:

```
Select * from spfli  CLIENT  SPECIFIED '800'  into spfli.
EndSelect.
```

Else, our table is called **Client Independent**. Means that, the data in our Table is visible to all client.

```
2 TABLES:  <TBL>
3 SELECT *  FROM  <TBL>  into wa.   ENDSELECT.
4 SELECT SINGLE *  FROM  <TBL>  into wa.   ENDSELECT.
  WHERE....ORDER BY  ASCENDING/DESCENDING ,  GROUP BY
5 SY-DBCNT,   SY-TABIX.
6 Data:  Begin of mFlight.
     INCLUDE  Structure Spfli.
  Data:  End of mFlight.
7 INSERT  <tbl>  from  wa.
  INSERT  into  <tbl>  values  wa.
  UPDATE
        wa-itno = 1.    Update <tbl> from wa.
        tbl-itno  =1.Update <tbl>
        Update <Tbl>  SET itname = 'xxx' where itno = 2.

  APPEND   wa-itno=1. Append WA.
  MODIFY   wa-itno = 1.    Modify <tbl> from wa.

  DELETE   wa-itno=1. DELETE <tbl> from wa.
                      Delete <tbl> where itno = 3.

  DYNAMIC:  NAME  ASSIGNMENT
        name = 'sflight'.     Update (name)  from wa.
```

13A MEMORY

Export <val> into memory id 'sym'. Import <val> from memory id 'sym'.	ABAP	**(ABAP Memory can't be extended to other session)**
Get Parameter id 'sym' field txt. Set Parameter id 'sym' field txt.	SAP	**(SAP Memory can be Viewed at other session also.)**

Compiled by: Ramani N

```
SUBMIT  zPrg   [VIA SELECTION-SCREEN]        SAP
       [USING SELECTION-SET <var>]
       [WITH <sel> <criterion>]
       [WITH FREE SELECTIONS <freesel>]
       [WITH SELECTION-TABLE <rspar>].
```

14 FILE HANDLING

```
1    DATA FNAME(60) VALUE 'myfile'.

2    OPEN DATASET FNAME FOR INPUT  [IN TEXT MODE]
     READ DATASET FNAME INTO NUM.     if sy-subrc <> 0.   ….endif.
3    OPEN DATASET FNAME FOR OUTPUT [IN TEXT MODE ]  MESSAGE MESS.

     IF SY-SUBRC <> 0.
      WRITE: 'SY-SUBRC:', SY-SUBRC,
          / 'System Message:', MESS.
     ENDIF

4    OPEN DATASET FNAME FOR APPENDING.
     NUM = NUM + 10.
       TRANSFER NUM TO FNAME.
```

EG:

```
Data: Begin of itab,  sn type I,  nm type c, End of Itab.
Data fname(10) type c value 'Ram.txt'.
Open DataSet  fname for output in text mode.
if sy-subrc = 0.
        Write: 'No File Error'.
Endif.

Itab-sn = 1.  Itab-nm = 'A'.  Perform FileWrite Using itab.
Itab-sn = 2.  Itab-nm = 'B'.  Perform FileWrite Using itab.

Close DataSet fname.

Form FileWrite using IT structure Itab.
        Transfer IT to fname.
EndForm.
```

```
5 If FileFound.    DELETE DATASET fname. Endif.
6 UPLOAD,  DOWNLOAD                  Data Stored in P-Server
  WS_UPLOAD,  WS_DOWNLOAD            Data Stored in A-Server


     CALL FUNCTION 'DOWNLOAD'
     EXPORTING.
        CodePage = 'IBM'
        FileName  = 'c:\ram.txt'.
        FileType   = 'ASC'.
     TABLES.
        DATATAB = ITAB.
```

15 CLASSICAL/ STANDARD PROGRAMMING.

SELECTION-SCREEN: BEGIN OF SCREEN….
SELECTION-SCREEN: END OF SCREEN…

Selection-Screen: Begin of block <> with frame title <>
Selection-Screen: Begin of line……End of line.
Parameters: <> as checkbox. <> RadioButton Group <>. MODIF ID <FN>
Selection-Screen: PUSHBUTTON

@Selection-Screen output.
Loop at screen.
Screen-name = <FN>. Screen-Input = 0.
EndLoop.
Modify Screen.

Events: Initialization
Start-of-selection. End-of-Selection
At Selection-Screen Output.
At Selection-Screen.
At User-Command.
LEAVE TO LIST-PROCESSING
SET PF-STATUS <>.

DIALOG PROGRAMMING SE51

1 Events: **PBO** Process Before Output
PAI Process After Input
POV Process on Value Request
POH Process on Help Request

FlowLogic
Process On Value-Request
FIELD matnr MODULE ModVal.
Report
MODULE ModVal.
Call Fn. 'F4IF-INT-TABLE-VALUE_REQUEST'
<……>
EndModule.

2 **List of Screen-Controls**
1) Input Box 4) TableView Control
2) Text Box 5) DropDown List Box
3) Frame 6) SubScreen

---

16 List/Interacting Reporting

No Report Page Heading
Line-Size =
Line-Count =

Events. At Line-Selection
At PF<>
At User-Command
Top-of-Page.
Top-of-Page During Line-Selection.

Sy: Sy-Lsind
Sy-Lilli
Sy-LiSel
Sy-Listi.
Set User-comand <>.

Compiled by: Ramani N

17 SAP Scripts

Page, Window, PageWindow, PgFormat, ChrFormat
99-Max Windows.

```
Call Function "Open_Form"
        Exporting
                Application = 'TX'
                Device = 'Printer'
                Dialog = 'X'
                Form = 'zForm'
                Language = sy-langu.
Call Function "Write_Form"
        Exporting
                Element = ''
                Function = 'set'
                type = 'body'
                window = 'MAIN'.
Call Function "Close_form."
```

18 Batch Data Communication see ANNEX-2

Compiled by: Ramani N

INTERNAL TABLE

Clear ITAB : Will Clear Header
Clear ITAB[] : Will clear Body
Refresh ITAB: Will remove contents from both Header & Body
FREE ITAB: Will deallocate memory associated with Internal Table

COLLECT COLLECT <wa> into ITAB.
It checks for the Header Data Existence. If Exists, Updates the data from changed fields. If Not Exists, Appends Data into Table. <valid for P,I,F type only)

CONTROL BREAK STATEMENT
At First.
At New <Itab-Field>.
At End of <Itab-Field>.
At Last.
At End of <Itab-Field>
Sum. {will sum all Numeric Field)
EndAt.

REPORTS

1) SELECT-OPTION: fld as mara-matnr.
No-Extension No-Intervals.
where , <fld> has four standard fields:

SIGN I or E
LOW [Lower Value]
HIGH [Higher Value]
OPTION

- If LOW value only given:
  [EQ,GT,LT,GE, LE]
- if LOW & HIGH value given:
  [BT] -> Between

2) CLASSICAL REPORTS / STANDARD REPORTS
Events:
- INITIALIZATION
- At Selection-Screen.
- At Selection-Screen On <Field>
- Start-Of-Selection
- End-Of-Selection.

3) INTERACTIVE REPORT
Events:
- At Line-Selection
- At User-Command
- At PF<key>
- TOP-OF-PAGE
- END-OF-PAGE

4) HIDE <Fields>.
Temporarily stores the content of clicked field in system area. So as to access the FIELD Value in the NEXT LIST.
.

5) System Fields for Interaction Listing

Sy-title	-Title of Report
Sy-linct	-Total Line No. of Page
Sy-Linno	-Current Line No
Sy-Lsind	-Index of List Created
Sy-Listi	-Index of Previous List
Sy-Lilly	-Line No.of List
Sy-Lisel	-Contents of Line Selected
Sy-Ucomm	-Fn.Code of Clicked Menu Item
Sy-pfkey	-Status of Displayed Item
Sy-Linsz	-Line Sz. of Report
Sy-Colno	-Cur. Col. No.
Sy-Pagno	-Current Page No.
Sy-Scols	-No. of Cols in Window
Sy-Srows	- No. of Lines in Window
Sy-CPage	-Cur. Page No.
Sy-CuCol	-Cursor Position Col #
Sy-CuRow	-Cursor Position Lin #
Sy-LStat	-Sts.Info. for Each List Level
Sy-Msgli	-Content of the Message Line
Sy-StaCo	-No.of First Displayed Col.
Sy-StaRo	-No. of First Displayed Row
Sy-RepId	-Name of Cur. Report

6) MENU PAINTER: TCODE> SE41

7) DIALOG PROGRAMMING / MODULE-POOL-PROGRAMMING

DYNPRO = Screen No. + Flow Logic

SCREEN-PAINTER: (TCODE: SE51)
EVENTS:
a) PBO :- (Process Before Output)
This event is triggered before the screen is displayed for filling of any default values in the screen fields.
b) PAI :- (Process After Input)
This event is responsible for processing of screen after the user enters the data and clicks the pushbutton. OKCODE plays an important role in this operation.
c) POV:- (Process on Value Request)
Is triggered when the user clicks F4 function key for list of possible values.
d) POH:- (Process on Help Request)
When the user places the cursor the field and presses F1 function key, the system displays its /our own help document for that field.

Compiled by: Ramani N

DATABASE-PERFORMANCE-TUNING

1
- Read Header Table First
- And, Then Read the Line Item Table Using SELECT.

2 Avoid using Tables with much number of rows.

3 Use as much as possible of PRIMARY-KEY

4 Avoid using CHECK statement, unless they can not be incorporated into SELECT Condition.

5 Avoid using MOVE-CORRESSPONDING eventhough it is convenient. Because, It makes server to think more.
Instead, individual (MOVE) statement can be used. It is faster in execution.

Eg: MOVE mMatnr into Itab-Matnr.

6 If we have 1 or 2 Tables only for a Query.
Better we can use Nested SELECT Statement.
Else if more than 2 Tables used.
Move all values into INTERNAL TABLE and then process.

7 WHILE instead of DO....ENDDO.
If there is any logical expression, WHILE condition can be used.

8 IF TOTAL-RECORD IN A QUERY > 1 LAC
We must go for FIELD-GROUPS instead of using INTERNAL TABLE.

By avoiding Internal Table, we can free-up Memory for other Applications.

9 If we feel that one of our program will hinder the system by taking more processing time, we can do a CHECK at the beginning of the program to make sure that the BATCH-MODE is TURNED ON. Otherwise, send an proper Error Message as given below:

```
IF SY-BATCH <> ' '.
  Message E999 with 'PROGRAM TO BE RUN AT THE BACKGROUND'.
ENDIF.
```

BDC (BATCH DATA COMMUNICATION)

1) S E S S I O N - METHOD

3-IMPORTANT FUNCTIONS IN BDC :-

(a) CALL FUNCTION 'BDC_OPEN_GROUP'
EXPORTING
GROUP = 'RAMANI'
* HOLDDATE = sy-datum
KEEP = ''
USER = 'SAPUSER'.

(b) CALL FUNCTION 'BDC_INSERT'
EXPORTING
TCODE = TCODE
TABLES
DYNPROTAB = MBDCDATA.

(c) CALL FUNCTION 'BDC_CLOSE_GROUP'.

2) T R A N S A C T I O N - METHOD

CALL TRANSACTION <TCODE>
USING <BDCDATA>
MODE <A / E / N>
UPDATE <SYNCHROUNUS /
ASYNCHRONOUS / LOCAL>

Where: A -> All Display
E -> Error Display Only
N -> No Display

Difference Between :

SESSION	CALL TRANSACTION
Data Updation Only After Session is processed	Immediate Updation into database
No Sy-Subrc is returned	Sy-Subrc is returned
Error Log created for error records	Error Need to be handled explicitly with SY-SUBRC
UPDATION is SYNCHRONOUS	Can be set to either SYNCH or ASYNCH
It is a Collection os Transaction Datas	No Data is collected, but updated immly.
Session can be processed at our required time	Immediate only
Processing is Synchronous	Processing is Asynchronous

LSMW – (Legacy Migration Workbench)

14-Import Steps in LSMW

1. Object Attribute

Standard Batch / Direct Input

{for Example}

Object :- 020 {Eg.for Material Master

Method :- 000

Program Name

Program Type

Batch Input Recording

BAPI

IDOC

2. Maintain Source Structure
3. Maintain Source Fields
4. Maintain Structure Relations

{for Example}

BGR00 -> Session Data

BM00 -> Batch Input Data

BMMH1-> Transfer of Main Data

5. Maintain Field Mapping & Conversion Rule
6. Main Fixed Values, Transaction, User-Defined Routines
7. Specify Files
8. Assign Files
9. Read Data
10. Display Read Data
11. Convert Data
12. Display Convert Data
13. Create Batch Input Session {for Batch input recording}
14. Start Direct Input Program

RULES Available In Step(5)

Initial, Constant, Transfer (Move), Fiexed Value, Translation, Prefix, Suffix, Concatenation, Left Trim, ABAP Coding, User Defined Routine.

SAP-SCRIPTS

```
Call Function 'OPEN_FORM'
    Exporting
            Application = 'TX'
        Device                  = 'PRINTER'
        Dialog                  = 'X'
        Form                    = 'ZFORM'
        Language    = sy-langu
Call Function 'WRITE_FORM'
    Exporting
Element                 = ' '
Function                = 'SET'
Type                    = 'BODY'
Window                  = 'MAIN'
Call Fn. 'START_FORM'    {If Required}
Call Fn  'END_FORM'
Call Fn  'CLOSE_FORM'.
```

Compiled by: Ramani N