Objective

You get a feel of ABAP/4 Programs with Basics covered in detail. In the Program Data Objects, Data type, Case , Write and use of Tables statement. You can see the use of all the statements and control words in the given example.

Output can be seen with a write statement in the end.

You can also see the use of Predefined Data type and reference type.

You see creation of reports in the program.

Creating and changing ABAP/4 Programs:

ABAP/4 Programs are objects of the R/3 Repository, and therefore maintained just like any other Repository objects (example ABAP/4 Dictionary Tables or user interface screens) using a tool of the ABAP/4 workbench, the ABAP/4 Editor.

To start the ABAP/4 Editor to create or change ABAP/4 Programs, R/3 System offers three possibilities:

- **Open programs in object browser**
- **Open programs using the ABAP/4 Editor**
- **Open Programs via forward navigation**

Open programs in the object browser

The object browser offers a hierarichal overview of all R/3 Repository objects,order by development classes, user name of the programmers, object types and so on. By selecting a program, the object browser directs access to all components of a program, such as main program, subroutines or global data.

The procedure is suited for complex (Interactive) reports or module pool for transaction, since in object browser you have an overview of all the components like user interfaces screens or Dynpros.

To open the ABAP/4 Programs in the object browser choose object browser of the ABAP Development Workbench
(or start Transaction SE80).

The initial Screen appear. Here you can enter a Program name directly or list of all programs of a certain development class.

Enter the Program name directly

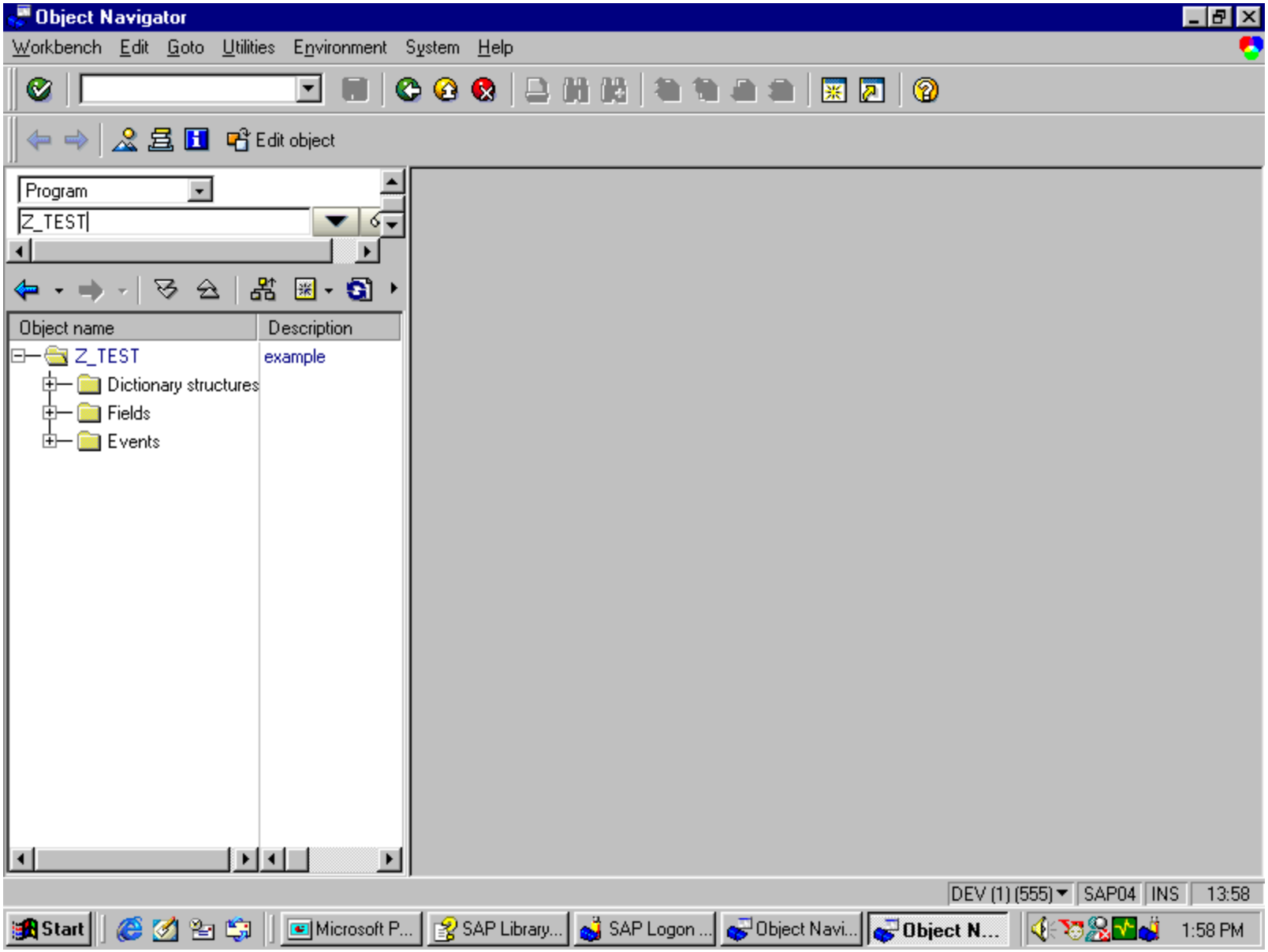
Enter a program name as per naming conventions into the object list and choose display.
If the program does not exist, a dialogue screen appears, asking you whether to create the program.
Otherwise object browser opens the specified screen.

Creating a New Program:
1) The dialogue box Create Program appears, which allows you to create a TOP INCL(top include program)
2) The screen ABAP/4:Program attributes appears where you maintain program attributes.
3) Save the program attributes
4) Use the ABAP/4 Editor for editing your program.

Displaying or Changing an existing program:
1) The browser displays the overview of all the components of a program

2) Position the cursor on the program name and choose display or change or double click.

Opening programs by object browser



Open programs using the ABAP/4 Editor

To open ABAP programs directly using the ABAP Editor, choose *ABAP Editor* in the *ABAP Workbench* screen or start Transaction SE38, and enter a program name.

Maintaining an Existing Program

To edit an existing program, enter its name on the initial screen of the ABAP Editor (Transaction SE38), select one of the following components, and then choose *Display* or *Change*.

***Source Code* - Starts the ABAP Editor.**

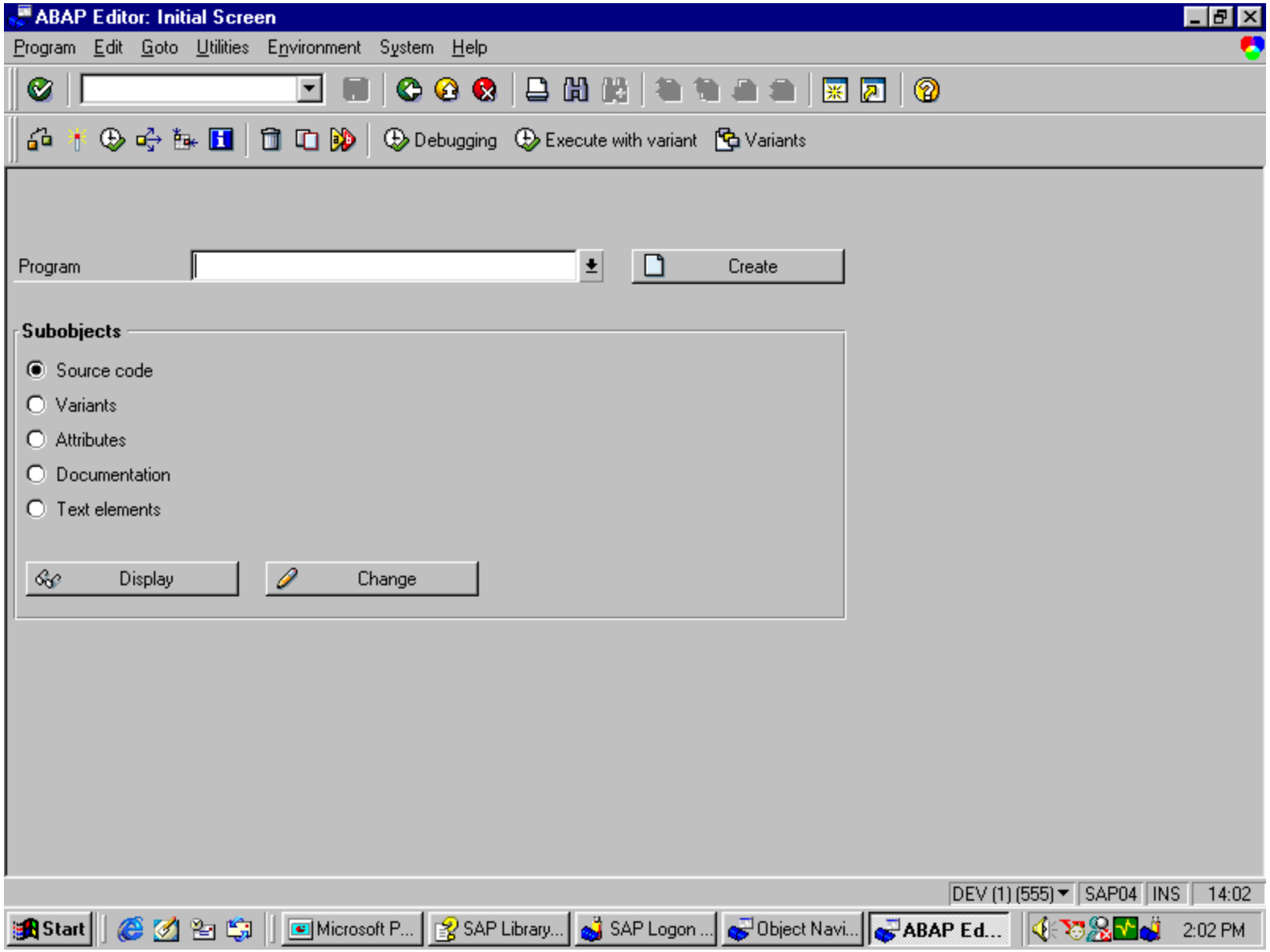
***Variants* - Starts the variant maintenance tool. Variants allow you to define fixed values for the input fields on the selection screen of a report.**

***Attributes* - Allows you to change the program attributes.**

Text elements- Appears on output screen of the report

***Documentation* - Allows you to write documentation for a particular executable program (report).**

ABAP/4 Editor screen



Open Programs via forward navigation

Whenever you work with a tool of the ABAP/4 Development Workbench and you position the cursor on a name of an R/3 Repository object and select the object the system opens the object together with the corresponding tool.

This procedure is suited whenever ABAP/4 programs are called from within objects such as screen flow logic or from within ABAP/4 Program.

Naming Conventions

A) Reports (Stand-alone programs):

Customer reports must follow this naming convention:Yaxxxxxx or Zaxxxxxx. Replace ‘a’ with the short form of the corresponding application area. Replace x with any valid character. Sap reports adhere with the similar naming convention: Raxxxxxx.

B) Module pool programs (for transactions):

Customer Dialogue programs must follow this naming convention: SAPMYxxx or SAPMZxxx. Replace x with any valid character. SAP standard programs adhere to a similar naming convention: SAPMaaxxx, where ‘a’ is the application area.

C) Valid Characters:

The name consist of at least one character,can have up to 8 characters.

Do not use following characters:

Period(.) equal sign(=) double quote(“) Comma(,) blank ( ) asterisk(*) parenthesis ’(‘ ’) percent(%) and underscore(_)

ABAP Syntax

The syntax of the ABAP programming language consists of the following elements:

- **Statements**
- **Keywords**
- **Comments**

Statements:

An ABAP program consists of individual ABAP statements. Each statement begins with a keyword and ends with a period.

Keywords:

A keyword is the first word of a statement. It determines the meaning of the entire statement.

Comments:

Comments are text elements which you can write between the statements of your program to explain it’s purpose to a reader.

Types of Keywords

•Declarative keywords:

These keywords define data types or declare data objects.

Example: TYPES, DATA, TABLES

•Modularization keywords:

These keywords define the processing blocks in an ABAP program. The modularization keywords can be further divided into:

*** Event Keywords:**

Example:

AT SELECTION SCREEN, START-OF-SELECTION, AT USER-COMMAND.

*** Defining keywords:**

Example:

FORM ..... ENDFORM, FUNCTION ... ENDFUNCTION,

MODULE ... ENDMODULE.

Keywords

•Control keywords:

You use these keywords to control the flow of an ABAP program within a processing block.

Example: IF, WHILE, DO, FOR, CASE.

•Calling keywords:

You use these keywords to call processing blocks that you have already defined.

Example: PERFORM, CALL, SET USER-COMMAND, SUBMIT, LEAVE TO

•Operational keywords:

These keywords process the data that you have defined using declarative statements.

Example: WRITE, MOVE, ADD.

Comments:

Comments are text elements which you can write between the statements of your ABAP/4 Program to explain it's purpose to the reader.Comments are flagged by special characters which causes system to ignore them. You should comment the document your program internally. Comments help other users to understand or change the program.

Structure of comments:

There are two ways to insert comment in the program

A) For entire line to be comment enter (*) at the beginning of the Line.

B) If you want to enter part of line as comment, enter a double quotation mark before the comment.

Syntax Structure

An ABAP program consists of different statements which have a particular structure. Each statement begins with a keyword and ends with a period.

Example:

This example shows the structure of an ABAP statement.

WRITE	**SPFLI-CITYTO**	**UNDER**	**SPFLI-CITYFROM.**
(Keyword)	**(Operand)**	**(Addition)**	**(Operand)**

Formatting ABAP Statements:

ABAP has no format restrictions. You must separate words within a statement with at least one space. You can write several statements on one line, or spread a single statement over several lines. Chained Statements:

You can concatenate consecutive statements with an identical first part into a chain statement by writing the identical part only once and placing a colon (:) after it.

Example:

```
WRITE: SPFLI-CITYFROM, SPFLI-CITYTO, SPFLI-AIRPTO.
```

Types of ABAP/4 Programs

- **Report programs**
- **Dialog programs**

Report programs:

A report program generates a list from database tables in a user defined format . It does not alter the data in the database but only analysis(reads) them.The results which can be displayed on the screen or sent to a printer.

A report program can either be an online or background program.

Dialog programs

Dialog programs read and change database tables. They are also called as Module pool programs. Dialog programs accept user information,process the information and update the database. For this reason module pool programs cannot be executed in background.

Workbench Tools

The ABAP/4 development workbench contains tools you need to create and maintain ABAP/4 programs.

• Object browser:
This utility offers a hierarchical overview of all R/3 repository objects.

•ABAP/4 Dictionary:
The ABAP Dictionary centrally describes and manages all the data definitions used in the system.

•ABAP/4 Editor:
You use the ABAP Editor to create and edit your programs and their components.

Workbench tools

•Function library:

The function library allows you to create and maintain the function modules.

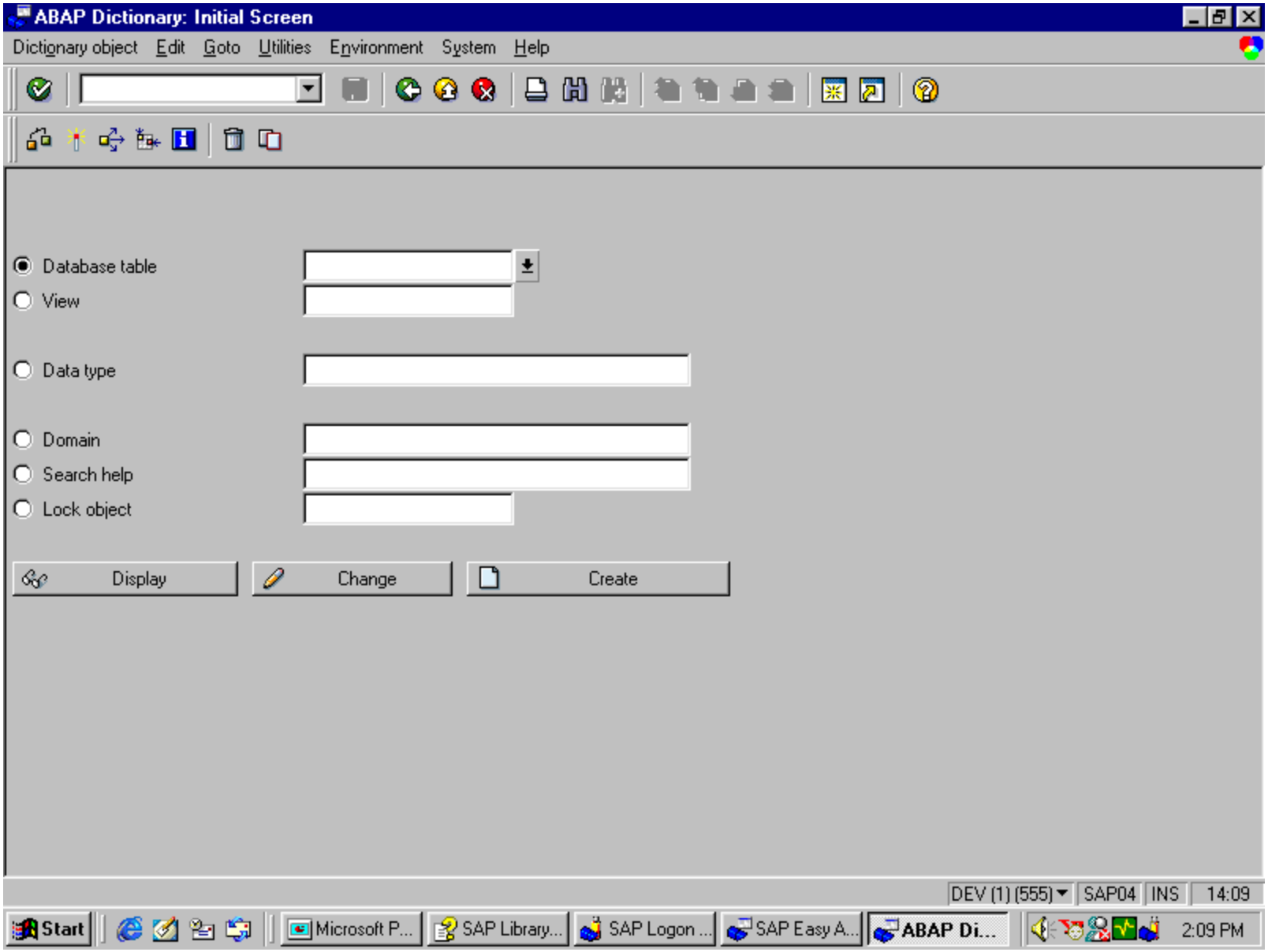
•Screen painter:

This ABAP Workbench tool allows you to create screens for your transactions. SAP R/3 provides two modes (graphical and alphanumeric modes) of the Screen Painter.

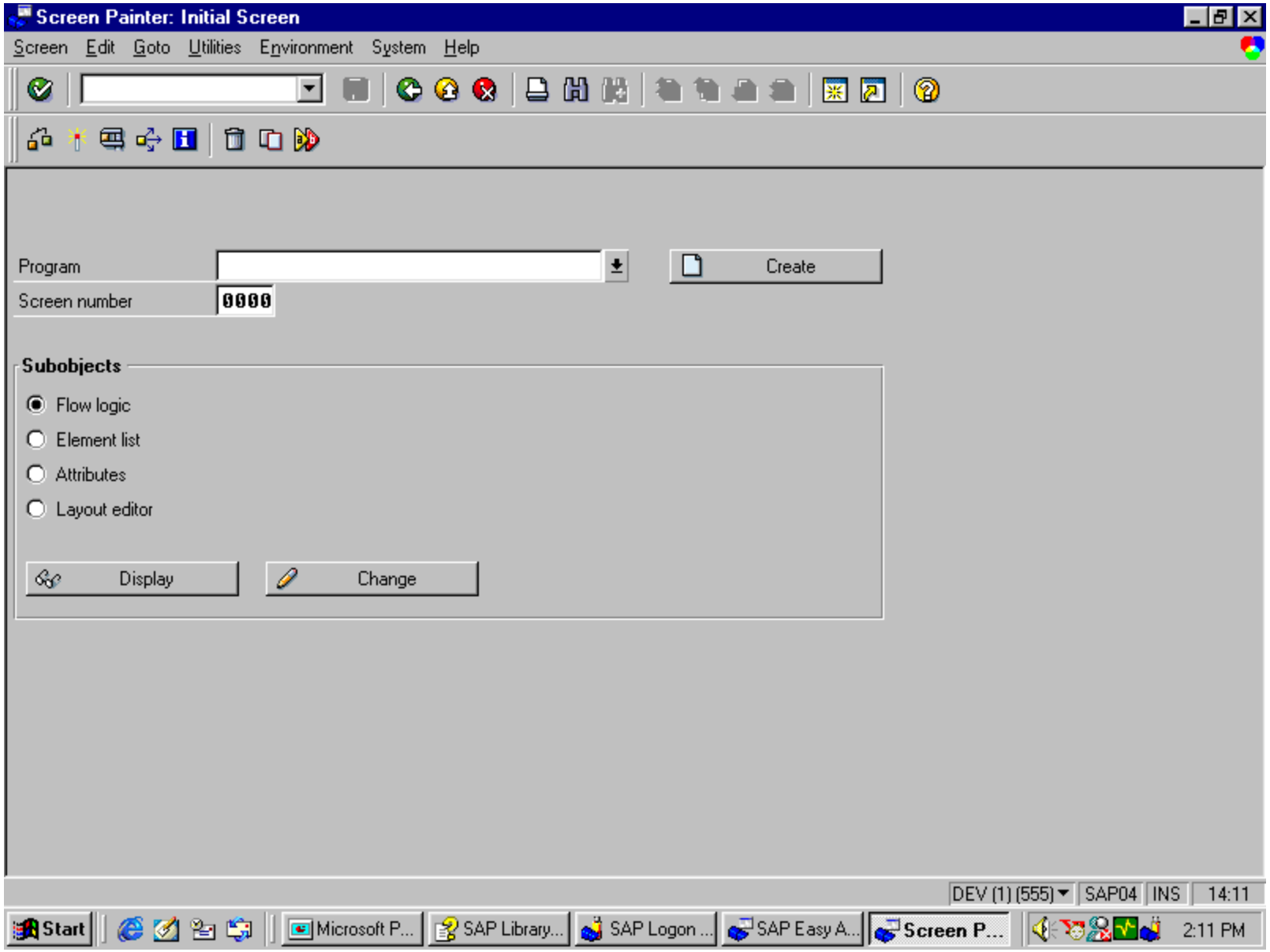
•Menu painter:

The Menu Painter is a tool with which you design user interfaces.

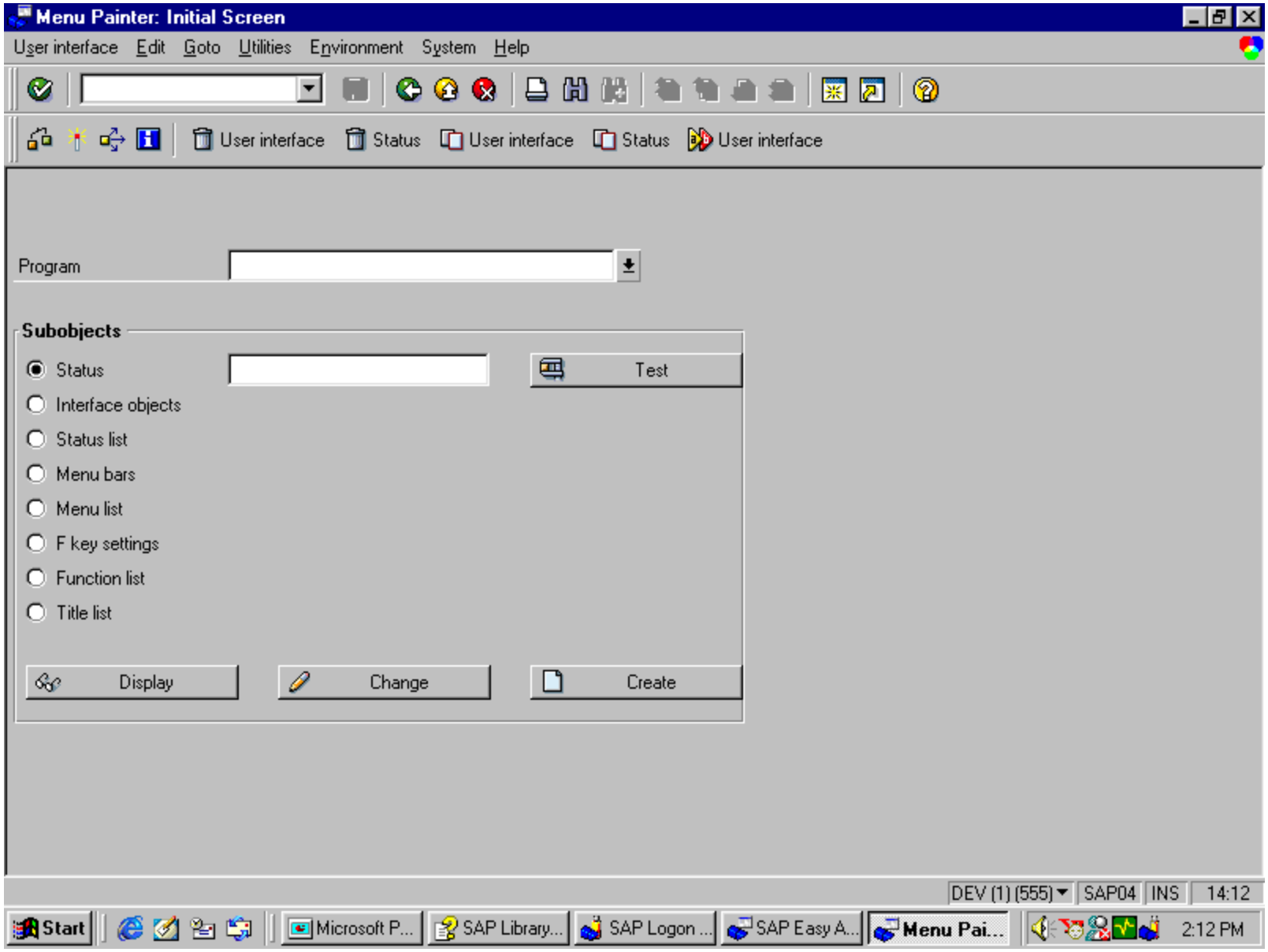
ABAP/4 Dictionary : Initial screen



Screen Painter - Initial Screen



Menu Painter - Initial screen



Data Types

The following list is an overview of the main features of data types and objects:

•Data types: A data type describes the technical attributes of all the objects with that type. There is no memory associated with data types.

Data types	Predefined	User-defined
Elementary	C,D,F,I,N,P,T,X ABAP/4 Contains eight predefined elementary data types	User defined elementary data typesare based on the predefined Elementary data types
Structured	TABLE :	Field Strings and internal tables :
	This predefined structured data type is used for the typing of formal parameters and field symbols	These structured data types can be used for data objects and are user defined.
Reference	Reference types describe data objects You will have to create your own that contain references (pointers) to references other objects (data objects and objects in ABAP Objects).	

Elementary Data Types- Predefined:

•Predefined:

Predefined elementary types are the smallest indivisible unit of types. They can be grouped as those with fixed length and those with variable length.

•Fixed-Length Elementary Types:

There are eight predefined types in ABAP with fixed length:

A)Four character types:	**Character (C), Numeric character (N), Date (D), and Time (T).**
B)One hexadecimal type:	**Byte field (X).**
C)Three numeric types:	**Integer (I), Floating-point number (F) and Packed number (P).**

•Variable-Length Elementary Types:

There are two predefined types in ABAP with variable length:

STRING for character strings **XSTRING for byte strings**

Elementary Data Types - User Defined

User-defined:

elementary data types are based entirely on predefined elementary data types. To define your own elementary data types, you use the TYPES statement

Example:

```
TYPES: NUMBER TYPE I.
DATA: NO_FLIGHTS TYPE NUMBER
```

Structured Data Types:

Structured Data types are made up of other types and generally are user defined. There are two kind of structured data types.

•Field Strings:

A field string is a collection of other data types.You Define Field string with the TYPES statement or with the DATA statement. structures as components.

Internal Tables:

Internal tables consists of a several lines that all have the same data type. Unlike field strings, which extends only horizontally, Internal Table also extends vertically.You define Internal Table with occurs parameter of the types or Data statement

Internal tables are characterized by:

The line type
The key and
The access method.

ABAP/4 Program Layout

To write a high quality program you should keep certain Layout standards for ABAP/4 Programs. You should observe these standards as soon as you start defining your data. Follow when structuring your program flow, and use as many as informative comments as possible. If you follow these suggestions your program will be

- **more readable**
- **easier to test and change**
- **more reliable**
- **To improve the quality of your programs, use the following Techniques:**

    Indenting statement blocks

    Using Modularization Tools

    Inserting Program comments

ABAP/4 Program Layout

Indenting statement blocks

You should combine statements that belong together into a single block . Indent each block by at least two columns.

Using Modularization Tools

To produce good programs you should use modularization tools. If you write larger processing blocks as subroutines, the logical structure of your program becomes easier to identify.It also allows you to sort the subroutines according to the tasks they perform.

Inserting Program comments Correctly

You should avoid placing comments on statement line. Placing them on separate comment lines improves the readability of the program.To insert subroutine headings and comments in your program use the ready-made structures available in ABAP/4 Editor.

Pretty Printer:

The ABAP/4 Editor includes a tool which helps you to design the Layout of your program more easily. It follows ABAP/4 Layout guidelines

Inserting ready-made structures

Ready-made structures simplify the coding of ABAP/4 Programs. They provide the exact syntax and follow the ABAP/4 layout guidelines.

You can insert two kinds of ready-made structures into your program code when using the ABAP/4 Editor:

A) Ready-Made Keyword Structures

To insert a ready-made keyword structure into your code, proceed as follows:

1. Place the cursor on the line where you want to insert the structure.

2.Choose EDIT --> Insert statement or select Pattern.

3. In the dialogue Box that appears, choose a statement with radiobutton or enter it in the other instruct . Field: To display a list of all available ready-made keyword structures, place the cursor in other instruct. field and click the possible entry button to the right of the input field.

Inserting ready-made Comment Lines

B) Inserting Ready-Made Comment Lines:

To insert ready-made comment lines into your code, proceed as follows:

1.Follow steps 1 to 2 in inserting Ready-Made keyword structures.

2.Select a structure with an asterisk(*) as a first structure from the other instruct. field.

3.The system inserts comment lines into your program

Data Objects

Data objects contain the data with which ABAP programs work at runtime.

ABAP contains the following kinds of data objects:

A) Literal

They are not created by declarative statements. Instead, they exist in the program code. They have fixed technical attributes but no name. Hence they are also referred to as unnamed data objects.

B) Named Data Objects

You declare these data objects either statically or dynamically at runtime. Their technical attributes are always fixed. ABAP contains the following kinds of named data objects:

Data Objects

Text symbols are pointers to texts in the text pool of the ABAP program.

Variables are data objects whose contents can be changed using ABAP statements.

Constants are data objects whose contents cannot be changed. Interface work areas are special variables that serve as interfaces between programs, screens, and logical databases.

Predefined Data Objects

They do not have to be declared explicitly - they are always available at runtime.

Dynamic Data Objects

You create them dynamically using data references. They do not have a name.

Internal Data objects

•Internal data objects are created for use in one particular program. They have no validity outside that program.

•Internal data objects include:

A)Literal

- Text literals (using type c)

Example: Data name(10) type c value ‘SATYAM’.

- Numeric literals (using type n)

Example: Data Pincode(10) type n value '600024'.

B)Variables

Example:

```
DATA: S1  TYPE I.
SUM = S1 + 10.
```

C)Constants

Example:

```
CONSTANTS  PI   TYPE  P  DECIMALS  10  VALUE   '3.1415926536’.
```

Data Objects

External data objects

External data objects exist independent of programs. You cannot work directly with them, but you can copy them to internal data objects and write them back when you have finished. External data objects can be used globally throughout the system environment.

ABAP/4 stores external data objects in tables defined in the ABAP/4 Dictionary. To access this data from within a program, you declare the tables in the program with the TABLES statement .

Data Objects : Contd..

System-defined data objects

Besides user-defined data objects, some data objects are defined automatically by the system. When you start an ABAP/4 program, some data objects are available automatically and do not need to be declared. These are called System-defined data objects. They include:

Space:

The data Object SPACE is a constant of type C. It is one character long and contains a space.

System fields:

All System fields have names with the format SY-<name>, where <name> specifies an individual field. To display a list of available system fields in the ABAP/4 Editor, type SHOW SY in the command line.

Example:

SY-UNAME : login name of the user
SY-DATUM : current date
SY-UZEIT : current time

Special data objects

ABAP/4 includes some data objects with special features, namely:

- **Parameters**

Parameters are variables which are linked to a selection screen. They can accept values after a program is started.

- **Selection criteria**

Selection criteria are special internal tables used to specify value ranges. They are linked to a selection screen.

Creating Data Objects and Types

Apart from literals, you must declare each data object with a declarative statement . In declarative statements, you must specify the data type of all data objects

You define the data type of an object in the declarative statement, either

directly, using <declaration>……TYPE<datatype>……..

Indirectly, using <declaration>like<dataobject>……..

- **Type and like are optional additions to most of the data declaration statements listed below.**
- **With the TYPE option, you assign the data type<datatype>directly to the declared data object.**
- **With the like option,you assign the data type of another data object<data object> to the declared data object.**

Creating Data Objects and Data Types:

ABAP/4 includes the keywords for creating Data objects and Data types statically:

The Data statement	**for creating variables**
The Constants statement	**for creating constants**
The Statics statement	**for creating variables which exist as long as the program runs, but are only visible in a procedure**
The Tables system	**for creating table work areas**
The Type statement	**for creating user-defined data types**

In the context of Internal Tables, you use the operational statements APPEND,COLLECT and INSERT to create lines of an Internal Tables dynamically.

In case of selection screens,you use the additional statements PARAMETERS SELECT-OPTIONS to create Data objects with special function

Basic Form of the DATA Statement

- **Data Statement**

- **Syntax**

DATA <f>[(<length>)] <type> [<value>] [<decimals>].

In its basic form, the keyword DATA has the following parameters:

<f>	**Naming a Variable**
<length> <type>	**Specifying the Data Type and the Length of the Variable**
<value>	**Specifying a Start Value**
<decimals>	**Specifying the Number of Digits after the Decimal Point**

Constants,Static and Tables

If you use a constant frequently in a program, you can declare it as a Fixed value variable with the CONSTANTS statement as follows:

Syntax: Constants<c>[<length>]<type><value>[<decimals>]

Statics Statement
If you want to retain the value of variable beyond the runtime of a procedure, you define the variable with STATICS Statement in that procedure.

Syntax: Statics<s>[<Length>]<type>[<Value>][<decimals>]

Tables statement
With the TABLES statement, you can create a data object called a table work area. A Table work area is a field string which refers to ABAP/4 dictionary objects.

Syntax: Tables<dbtab>

The Types Statement

You use the TYPES statement to create user-defined elementary data types and structured data types. You can use data types defined by the TYPES statement in the same way you use predefined data types for declaring data objects.

Syntax: Types<t>[<length>]<<type>[<decimals>]

TYPE GROUPS

You use Type-Groups to store user-defined data types or constants in the ABAP/4 Dictionary for cross program use.In your ABAP/4 program, you declare type groups with the TYPE-POOLS statement as:

Syntax: Type-Pools<name>

Determining the Decimal Places

To determine the number of decimals for a Type P field, you use the DECIMALS parameter with the DESCRIBE FIELD statement as follows:

Syntax: Describe Field<f>Decimals<d>.

Determine the conversion routine

To determine whether a conversion routine exists for a field in ABAP/4 Dictionary use the EDIT MASK parameter with the Describe Field statement as follows:

Syntax: Describe Field<f>EDIT MASK<m>

If a conversion routine exists for the field <f> in the ABAP/4 Dictionary, the system writes it to the field <m> and sets the return code value in the System Field SY-SUBRC equal to 0.

You can then use the field <m> directly as a format template in a write statement , as below:

Write<f>using EDIT MASK<m>

Naming a Variable

•**The variable name <f> may be up to 30 characters long. You can use any alphanumeric characters except those listed below.**

- **Do not use the following characters:**
  - **plus sign +**
  - **period .**
  - **comma ,**
  - **colon :**
  - **parentheses ( )**
- **Do not create a name consisting entirely of numeric characters.**

The write statement

The basic ABAP/4 statement for outputting on the screen is WRITE.

Syntax: WRITE<f>.

This statement outputs the field <f> to the current list in its standard output format.

The field <f> can be:

A) Any Data Object

B) A Field Symbol

C) A Text Symbol

Basic Form of the WRITE TO Statement

•To write a value (literal) or the contents of a source field to a target field, you use the WRITE TO statement:

Syntax

WRITE <f1> TO <f2> [<option>].

Example

```
DATA: NUMBER TYPE F VALUE '4.3',
            TEXT(10).
WRITE NUMBER TO TEXT EXPONENT 2.
WRITE / TEXT.
```

Positioning WRITE Output on the Screen

You can position the output of a write statement on the screen by making a format specification before the field name as follows:

Syntax
WRITE AT [/][<pos.>][(<len>)] <f>.

A) the slash / denotes a new line.
B) <pos.> is a number or variable up to three digits long denoting the position on the screen.
C)<len> is a number or variable up to three digits long denoting the output length.

Formatting options

You can use formatting options with the write statement.

Syntax: Write……..<f><option>

Option	Purpose
LEFT-JUSTIFIED	Output is left-justified.
CENTERED	Output is centered.
RIGHT-JUSTIFIED	Output is right-justified.
UNDER <g>	Output starts directly under the field <g>.
NO-GAP	The blank after the field <f> is omitted.

Formatting options for numeric fields

NO-SIGN	The leading sign is not output.
DECIMALS <d>	<d> defines the number of digits after the decimal point.
EXPONENT <e>	In type F fields, the exponent is defined in <e>.

Outputting Symbols and Icons on the Screen

•You can output symbols or R/3 icons on the screen by using the following syntax:

Syntax

```
WRITE <symbol-name> AS SYMBOL.

WRITE <icon-name>   AS ICON.
```

Example:

```
INCLUDE <LIST>.

WRITE: /  'Phone Symbol:', SYM_PHONE AS SYMBOL.

SKIP.

WRITE: /  'Alarm Icon:  ', ICON_ALARM AS ICON.
```

Lines and blank lines on the output screen:

A) Horizontal Lines:

You can generate horizontal lines on the output screen by using the following syntax:

Syntax: ULINE[AT[/][<pos.>][(,len.)].

Horizontal lines are output as specified.

B) Vertical lines:

You generate vertical lines on the output screen by using the following syntax: Write[AT[l][pos.]] SY-VLINE.

C) Blank Lines

Syntax: SKIP[<n>]

Starting on current line , this statement generates <n> blank lines on the output screen.

To position the output on a specific line on the screen use:

Syntax: SKIP TO LINE <n>

Outputting Field contents as check boxes

You can output the first character of a field as a checkbox on the output screen by using the following

syntax:

WRITE <f> AS CHECKBOX.

In the first character of the field <f> is an “X”, the checkbox is displayed filled. If the first character is a SPACE, the checkbox is displayed blank. With other words user can fill or empty the checkbox by mouse clicks.

Using Write via a Statement Structure

The R/3 System includes a useful facility for trying out all the options and output formats of the write statement and inserting them into your program.

Syntax:

Edit ===> Insert statement…… in the ABAP/4 Editor

Then select WRITE in the relevant dialogue box.

When you have confirmed your selection with a enter, you see the screen where you can:

A) determine the output format of an internal field by entering its name or a literal in the field FLD.

B) generate the WRITE statements for symbols,icons,and checkbox simply by selecting the appropriate field

C)generate the write statements for components of structures defined in the data dictionary.

Working with text elements and concept

The ABAP/4 programming environment allows you to create and maintain programs in several languages. You can store all texts the program outputs to the screen as text elements in text pools.

Text elements comprise

A)The title of the program.

B)List headings and column headings for page headers of output lists

C)Selection texts which appear on a selection screen

D)Text symbols which can be used in ABAP/4 statements instead of literals.

Creating and changing text elements

Steps to create or change program specific text elements:

A) Choose Tools===> ABAP workbench==>Development==>Programming environment ==>Text Elements

B) In the object browser of the ABAP/4 Workbench, choose program objects for a specific program and then Text elements.

C)Choose GOTO=> Text Elements in the ABAP/4 Editor.

D)On the ABAP/4 Editor: Initial screen (SE38), enter in the program field the name of the program you want to maintain the text elements.

E) Select Text Elements and choose display or change.

F)All actions take you to the ABAP/4 Text Elements screen.

G)You can now choose to maintain the program.

Titles and Headers

A)Titles:

You enter the program title when specifying the program attributes. You may change this title as you wish.You can enter a title of up to 70 characters in the Title field.

Save your changes by choosing save.

B)Headers:

You can create or change the header line for the output list of your program and the column headers for the different columns in the list..

To create or change the titles in the output screen,select titles and headers from the ABAP/4 Text Elements screen and choose change.

You can enter a list characters up to 70 characters in the List header field and column headers up to 255 characters in the four lines of the Column Header Field.

Save your changes by choosing save.

Selection Texts

Replace the standard Text:

You can replace the standard texts that appear for parameters and selection criteria on the selection screen by text elements.

A)To change the text on the selection screen,select Selection texts from the ABAP/4 Text elements screen and choose change.

B)On the following screen,the column name contains the names of parameters and selection criteria of your program. Now you can enter a selection text up to 30 characters for each parameter and selection criterion.

Text Symbols

Text symbols are text constants which you enter and maintain outside a program.You should use text symbols instead of text literals in the final version of your program to keep it language-independent and easy to maintain.

Maintain Text symbols:

To create or change text symbols , select Text symbols on the ABAP/4 Text Elements screen and choose change.For each text symbol,you must specify a three-character identifier which contains no blanks and does not begin with the character '%'.You can assign a text up to 132 character to each text symbol.You specify 3 character identifier in column Sym and the text in column Text.

Using Text Symbols in ABAP/4 Programs:

You use text symbols exactly as literals in your ABAP/4 programs. At each position in a statement where you can write a literal, you can write also a text symbol. If the text symbol<idt> does not exist in the text pool, the system treats TEXT<idt>like the constant SPACE.

Syntax: …TEXT-<idt>...

Comparing Text Elements,Symbols,Selection Texts,

A)Comparing Text Elements:

By choosing the function compare on the ABAP/4 Text elements screen, you can adapt the text elements of the program to the source code. This function is possible for Selection texts and Text symbols but not for Titles and Headers.

B)Comparing Selection Texts:

When you choose Selection Texts and compare the system supports you by finding missing or unused selection texts.

C)Comparing Text Symbols:

If you insert new text symbols or change existing ones in the program code. These text symbols are not automatically copied to the Text Pool.To update this list and to eliminate any error,choose text symbols and the function compare on the ABAP/4 Text Elements Screen.

Arithmetic operations

ABAP/4 supports the four basic arithmetic operations, as well as power calculation. You can specify the following arithmetic operators in a mathematical expression:

+	Addition
-	Subtraction
*	Multiplication
/	Division
DIV	Integer division
MOD	Remainder of integer division
**	Exponentiation

SUMMARY

You must have understood the basics after doing above exercises. Practice more and you will be able to command on the language the most. ABAP/4 Basics is sufficiently covered in the above presentation.

Example 1

Name of your report: ZSHU017
ID Number :

TASK : Create the List as per the output below:
Output:

10	1	XX	1
20	2	BB	2
30	3	XX	3
40	4	DD	4
50	5	XX	5

Solution

```
REPORT ZSHU017 .
data: begin of it occurs 3,
        f1(2) type n,
        f2    type i,
        f3(2) type c,
        f4    type p,
        end of it.

 it-f1 = '40'. it-f3 = 'DD'. it-f2 = it-f4 = 4. append it.
 it-f1 = '20'. it-f3 = 'BB'. it-f2 = it-f4 = 2. append it.

 sort it by f1.
```

Example 1 Solution

```
do 5 times.
    it-f1 = sy-index * 10.
    it-f3 = 'XX'.
    it-f2 = it-f4 = sy-index.
    read table it
       with key f1 = it-f1
       binary search
       transporting no fields.
    if sy-subrc <> 0.
       insert it index sy-tabix.
       endif.
    enddo.

 loop at it.
    write: / it-f1, it-f2, it-f3, it-f4.
    endloop.
```

Example 2

Example of parameters , Data, Predefined Data type, Events,write.

Name of your report: Z10393_10
ID Number :

Task : To find whether a given number is a palindrome
Output :

Input string = 20
Output string = 02
The given text is not Palindrome

Example 2 - Solution

Solution

```
report z10393_10 .
* Variables declaration.
data : v_text1(100) type c.
data : v_text2(100) type c.
data : v_text3(100) type c.
data : v_length type i.
********** **********
* Selection screen
parameters : p_input(100) type c.

**********
* Start of selection....
v_text1 = p_input.
```

Example 2 - Solution

```
* Get the string length.
v_length  = strlen( v_text1 ).

* Start a while loop...
while v_length >= 0.

* select last letter,last-1...
  v_text2 = v_text1+v_length(1).

* lastletter+last-1letter+.....
  concatenate v_text3 v_text2 into v_text3.  ...

  v_length = v_length - 1.
endwhile.
```

Example 2 - Solution

```
* Output....
write :/1 'Input string    = ', 20 v_text1.
write :/1 'Reverse string  = ', 20 v_text3.
if v_text3 = v_text1.
  write :/ 'The given text is palindrome'.
else.
  write :/ 'The given text is not a palindrome'.
endif.
```

Example No. 3

Name of your report: Z31212
ID number :
TASK : To retrieve material master information
Output :

MBLNR	MJAHR	ZEILE
Material Number: 2		Desc: 567

MBLNR	MJAHR	ZEILE
5000000000	2001	0001
4900000000	2001	0001
5000000001	2001	0001
5000000002	2001	0001
4900000001	2001	0001

Solution-Example 3

```
REPORT Z31212_EXERCISE2_MM NO STANDARD PAGE HEADING .
TABLES: MARA,
        MSEG,
        MAKT.
DATA: BEGIN OF I_MARA OCCURS 0,
        MATNR LIKE MARA-MATNR,
        MTART LIKE MARA-MTART,
        MATKL LIKE MARA-MATKL,
      END OF I_MARA,
      BEGIN OF I_MSEG OCCURS 0,
        MATNR LIKE MSEG-MATNR,
        MBLNR LIKE MSEG-MBLNR,
        MJAHR LIKE MSEG-MJAHR,
        ZEILE LIKE MSEG-ZEILE,
```

Solution-Example 3

```
 END OF I_MSEG,
      BEGIN OF I_MAKT OCCURS 0,
        MATNR LIKE MAKT-MATNR,
        MAKTX LIKE MAKT-MAKTX,
      END OF I_MAKT.
SELECTION-SCREEN BEGIN OF BLOCK B1.
SELECT-OPTIONS: S_MATNR FOR MARA-MATNR,
                S_MTART FOR MARA-MTART,
                S_MATKL FOR MARA-MATKL.
SELECTION-SCREEN END OF BLOCK B1.

SELECT MATNR MTART MATKL
FROM MARA
INTO TABLE I_MARA
WHERE MATNR IN S_MATNR
AND MTART IN S_MTART
```

Solution-Example 3

```
AND MATKL IN S_MATKL.
IF NOT I_MARA[] IS INITIAL.
  SELECT MATNR MBLNR MJAHR ZEILE
  FROM MSEG
  INTO TABLE I_MSEG
  FOR ALL ENTRIES IN I_MARA
  WHERE MATNR EQ I_MARA-MATNR.
  SELECT MATNR MAKTX
  FROM MAKT
  INTO TABLE I_MAKT
  FOR ALL ENTRIES IN I_MARA
  WHERE MATNR EQ I_MARA-MATNR.
ENDIF.
LOOP AT I_MARA.
  AT NEW MATNR.
    WRITE :/5 'Material Number:',I_MARA-MATNR.
  ENDAT.
```

Solution-Example 3

```
LOOP AT I_MARA.
  AT NEW MATNR.
    WRITE :/5 'Material Number:',I_MARA-MATNR.
  ENDAT.
LOOP AT I_MAKT WHERE MATNR = I_MARA-MATNR.
    WRITE :  35 'Desc:', I_MAKT-MAKTX.
  ENDLOOP.

  SKIP.
  WRITE :/5 'MBLNR', 30 'MJAHR', 55 'ZEILE'.
  LOOP AT I_MSEG WHERE MATNR = I_MARA-MATNR.
      WRITE :/5 I_MSEG-MBLNR , 30 I_MSEG-MJAHR, 55 I_MSEG-ZEILE.
  ENDLOOP.
    ULINE.
ENDLOOP.
```

Example 4

Name of your report: Z31212

ID Number :

Task : Learn ABAP/4

Output:

The number is bad 39 number is good

sum of dates*

Solution-Example 4

```
REPORT Z31212_LEARN .
*---------------------------------------------------------------------

*example for between.

DATA NUM TYPE I.
NUM = 34.
IF NUM BETWEEN 30 AND 33.
  WRITE 'number is good'.
ELSE.
  WRITE 'number is bad'.
ENDIF.

*---------------------------------------------------------------------
```

Solution-Example 4

```
*example for add.
NUM = 34.
IF NUM BETWEEN 30 AND 35.
  ADD 5 TO NUM.
  WRITE NUM.
  WRITE 'number is good'.
ELSE.
  WRITE 'number is bad'.
ENDIF.
*---------------------------------------------------------------------
DATA : DD1 TYPE D VALUE '19900101'.
DATA : DD2 TYPE D VALUE '19990000',SUM.
SUM = DD1 + DD2.
WRITE :/ 'sum of dates', SUM.
*---------------------------------------------------------------------
```

Solution-Example 5

Example 5

Name of your report: ZSHU006
ID number :

Task : Do loop exercise.
Output:

1	**2**	**3**	**4**	**5**	**99**
2	**4**	**6**	**8**	**10**	**6**

Solution-Example 5

```
REPORT ZSHU006 .
data: f1 type i,
        begin of s,
            c1 type i value 1,
            c2 type i value 2,
            c3 type i value 3,
            c4 type i value 4,
            c5 type i value 5,
            c6 type i value 6,
           end of s.
 field-symbols <f>.

 write / ''.
```

Solution - Example 5

```
do 6 times varying f1 from s-c1 next s-c2.
    if sy-index = 6.
        s-c6 = 99.
    else.
        f1 = f1 * 2.
        endif.
    assign component sy-index of structure s to <f>. "<f> now points to
    write <f>.                                    "a component of s
    enddo.

write / ''.
do 6 times varying f1 from s-c1 next s-c2.
    write f1.
    enddo.
```