Introduction to ABAP Programming

Lesson Objectives

After completing this lesson, participants will be able to –

- Understand The Need for ABAP
- Know the types of ABAP/4 Programs
- Create Reports
- Write the Program Code
- Test the Program
- Combine similar statements to one statement
- Illustrate Defining Data Types and Data Objects
- Recognize the System Variables

What is ABAP/4?

ABAP/4 is a 4 generation programming language that you can use multiple ways:

- You can select and edit data you want to process in traditional ways via the screens that guide you.
- You can write reports using the interactive reporting facility.

ABAP/4 (Advanced Business Application Programming-4) language was developed by SAP to provide optimal working conditions for application programmers.

It is the sole tool used by SAP to develop its own applications.

SAP customers use ABAP/4 to adapt R/3 standard solutions to specific problems.

Features of ABAP/4

Multi-Language Support

Supports business data types and operations

Open SQL

Use of sub routines

Introduction to Reporting

Purpose:

Reports are Programs that read data from the database, processes the data and displays the data in the required format.

Use:

Reports are used in day to day business environment.

Example:

- Displaying the purchase orders vendor wise.
- Displaying the balance of vendors to be paid till a particular date.

Program Types

Program type	Introductory statement
1 or E	Executable Program
I	INCLUDE Program
M	Module Pool Program
F	Function-Pool
S	Subroutine Pool
K	Class-Pool
J	Interface-Pool
T	Type Pool

Report Program

- The purpose of a report is to read data from the database and write it out.
- It consists of only two screens.
  - The first screen is called the selection screen.
    - It contains input fields allowing the user to enter criteria for the report.
  - The second screen is the output screen.
    - It contains the list.
- The list is the output from the report, and usually does not have any input fields.
- The selection screen is optional. Not all reports have one. However, all reports generate a list.

Creating Reports

- A report consists of individual statements that start with a reserved word and end with a period.

- E.g.

```
WRITE XYZ.
MOVE  SALES TO TOTAL_SALES.
```

- The first word of statement (the reserved word) determines the meaning of the whole statement.

Creating Reports

ABAP/4 report program can be created from the ABAP/4 editor (Transaction Code: SE38).

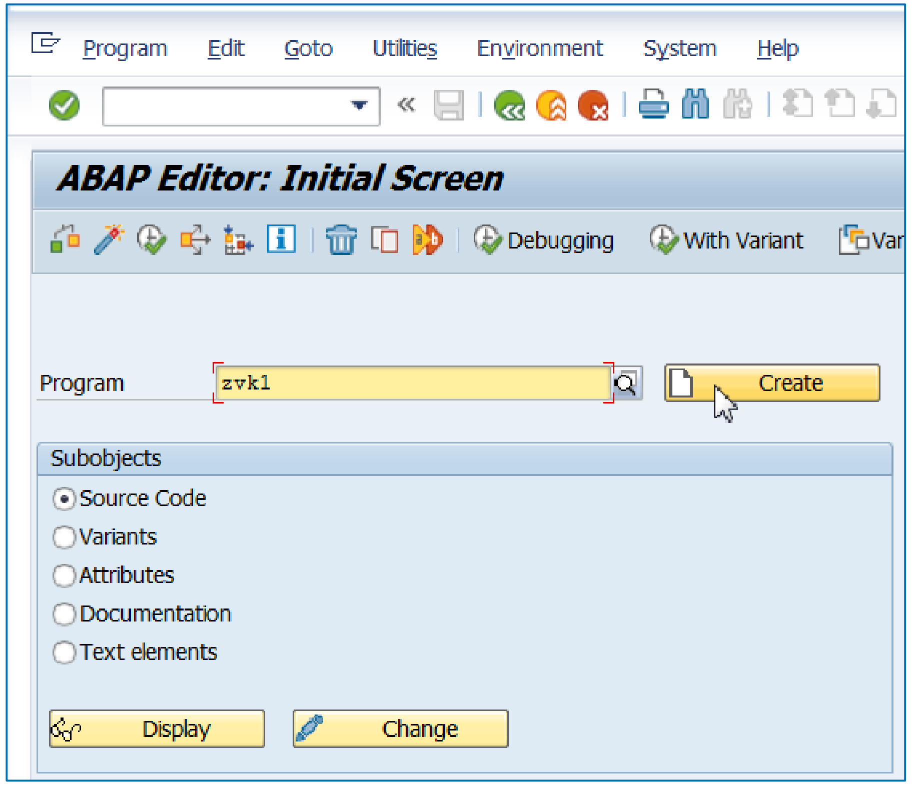
Creating a report program involves the following steps

- Creating the program
- Specifying the program attributes
- Writing the program code
- Testing the program

Creating Reports

Run transaction SE38 to go to the ABAP/4 Editor.

This enables to create report program.



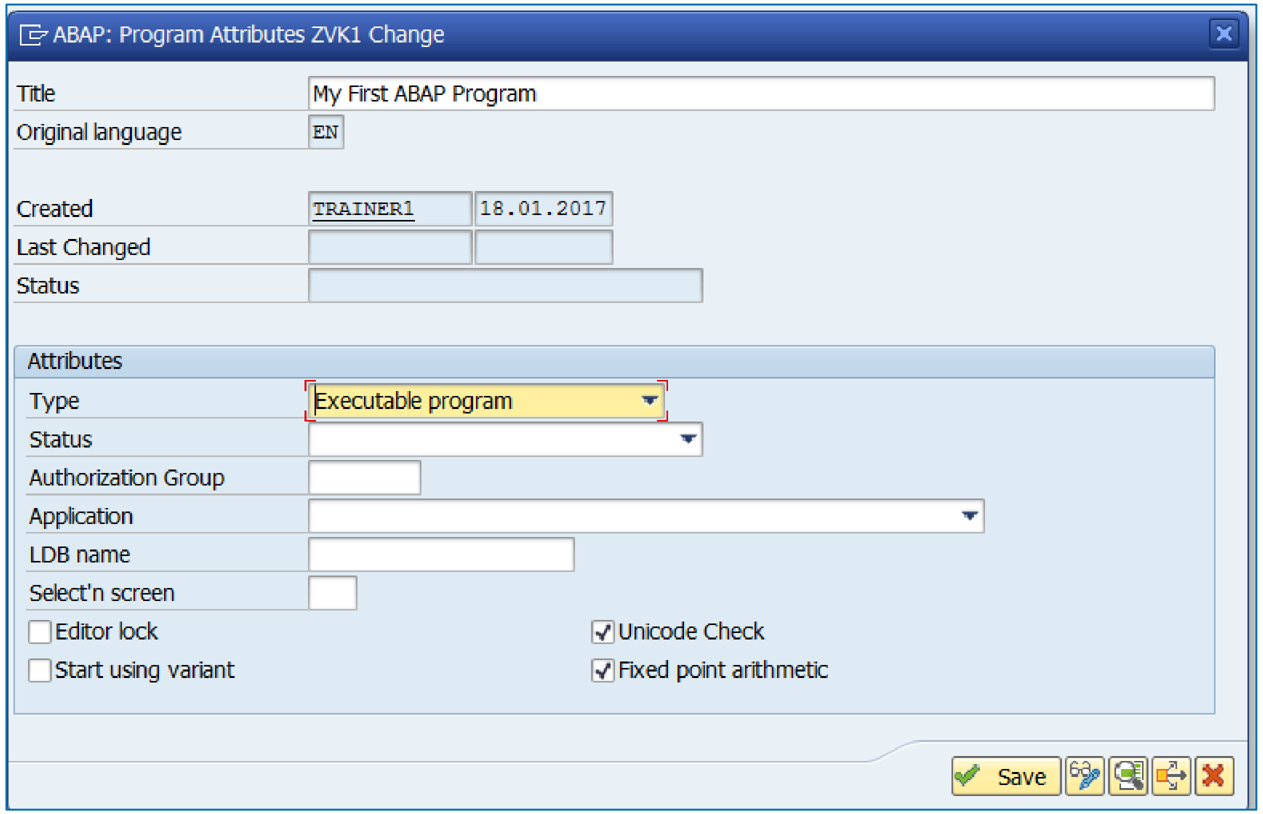
Program zvk1

Subobjects

(●) Source Code
( ) Variants
( ) Attributes
( ) Documentation
( ) Text elements

Creating Reports - Specifying program Attribute

Program attributes determine to which application a program belongs .


ABAP: Program Attributes ZVK1 Change

Title	My First ABAP Program
Original language	EN
Created	TRAINER1 18.01.2017
Last Changed	
Status	

Attributes

Type	Executable program
Status	
Authorization Group	
Application	
LDB name	
Select'n screen	

☐ Editor lock
☐ Start using variant
☑ Unicode Check
☑ Fixed point arithmetic

ABAP Syntax

Rules for ABAP Syntax

- The first word in the statement is the ABAP keyword
- Each statement ends with a period (.)
- Comment line is marked with a '*'
- Comments from the middle of a line begins with "
- The ABAP runtime system does not differentiate between uppercase and lowercase in keywords, additions and operands

Writing Program

ABAP/4 code is written from the ABAP/4 editor.

Test the Program

To test the program, select the ‘Execute’ button.

At runtime, the source code of the ABAP/4 program is compiled.

This compilation process is known as generation.

The generated form of the program is stored in the ABAP/4 repository.

As the program is automatically generated at run time while execution, one does not have to generate it separately.

The program will be regenerated at each run if some modifications have been made to the code.

The ABAP/4 also provides for various debugging mechanisms

Demo

Create first ABAP Program and execute it

Example.

```
REPORT Z.
DATA NAME(25) TYPE C VALUE 'Leena'.
DATA NUMBER   TYPE I VALUE 1.

WRITE NUMBER.
WRITE NAME.
```

Chained Statements

Used to Combine statements

The chain operator used is ‘:’

Example

Statement sequence:

```
        WRITE var1.
        WRITE var2.
        WRITE var3.
```

Chain Statement:

```
WRITE  :  var1, var2, var3.
```

Demo on Chain statement

```
REPORT  Z.
DATA: NUMBER   TYPE I VALUE 1,
        NAME(25) TYPE C VALUE 'Leena'.

WRITE: 'The Number is ', NUMBER.
WRITE: /'The Name is ', name.
```

Demo

Demo of chain statement

Data Types

A formal variable description is called a data type.

A data type characterizes the technical attributes of all data objects.

Data types can be divided into elementary, reference, and complex types.

Elementary types are not composed of other types.

They are further classified into elementary types of fixed length and of variable length.

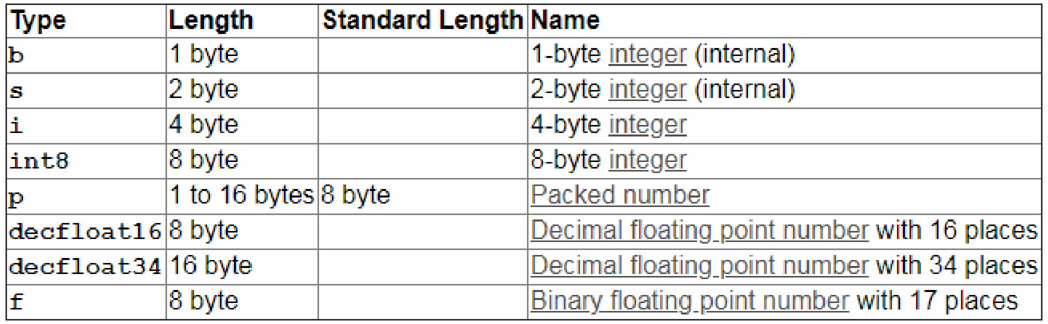
There are 13 predefined elementary data types of fixed length in ABAP.

Predefined ABAP Types

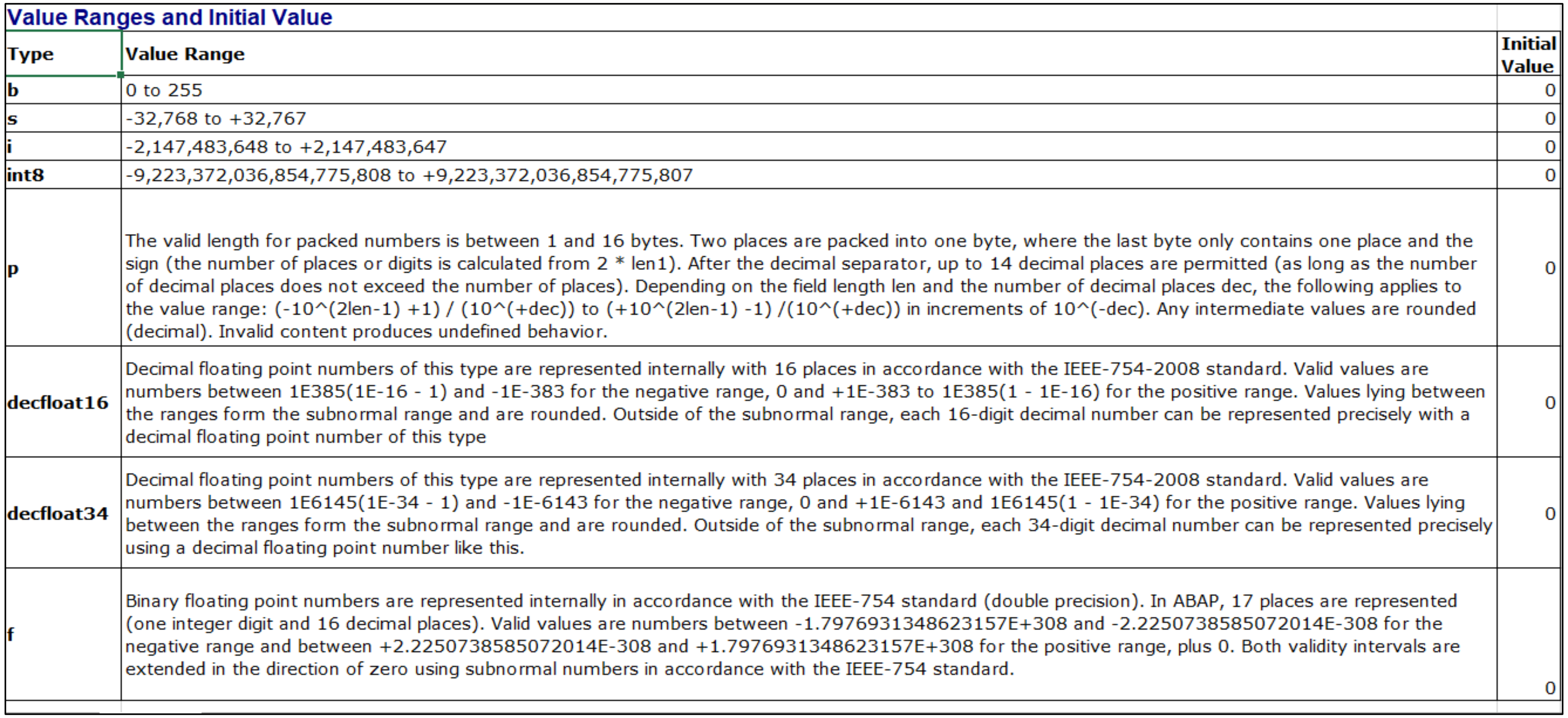
The following types are predefined in every ABAP program:

- Predefined Numeric Types
- Predefined Character-Like Types
- Predefined Byte-Like Types
- Predefined date types and time types

Predefined Numeric Types



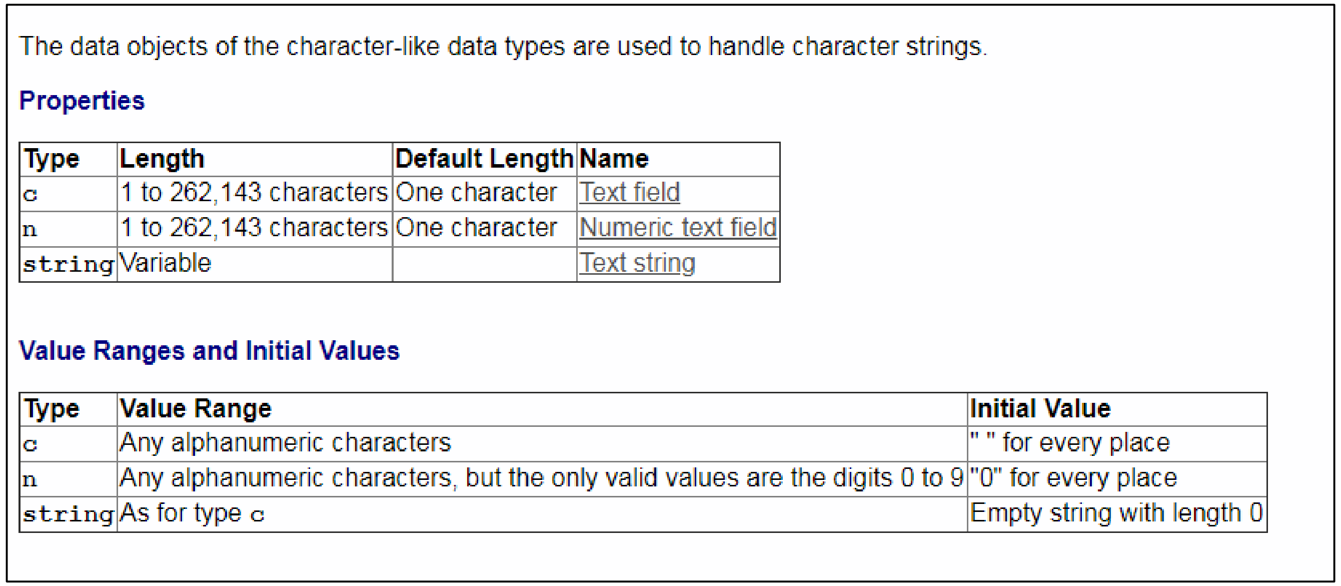
Type	Length	Standard Length	Name
b	1 byte		1-byte integer (internal)
s	2 byte		2-byte integer (internal)
i	4 byte		4-byte integer
int8	8 byte		8-byte integer
p	1 to 16 bytes	8 byte	Packed number
decfloat16	8 byte		Decimal floating point number with 16 places
decfloat34	16 byte		Decimal floating point number with 34 places
f	8 byte		Binary floating point number with 17 places



Value Ranges and Initial Value

Type	Value Range	Initial Value
b	0 to 255	0
s	-32,768 to +32,767	0
i	-2,147,483,648 to +2,147,483,647	0
int8	-9,223,372,036,854,775,808 to +9,223,372,036,854,775,807	0
p	The valid length for packed numbers is between 1 and 16 bytes. Two places are packed into one byte, where the last byte only contains one place and the sign (the number of places or digits is calculated from 2 * len1). After the decimal separator, up to 14 decimal places are permitted (as long as the number of decimal places does not exceed the number of places). Depending on the field length len and the number of decimal places dec, the following applies to the value range: (-10^(2len-1) +1) / (10^(+dec)) to (+10^(2len-1) -1) /(10^(+dec)) in increments of 10^(-dec). Any intermediate values are rounded (decimal). Invalid content produces undefined behavior.	0
decfloat16	Decimal floating point numbers of this type are represented internally with 16 places in accordance with the IEEE-754-2008 standard. Valid values are numbers between 1E385(1E-16 - 1) and -1E-383 for the negative range, 0 and +1E-383 to 1E385(1 - 1E-16) for the positive range. Values lying between the ranges form the subnormal range and are rounded. Outside of the subnormal range, each 16-digit decimal number can be represented precisely with a decimal floating point number of this type	0
decfloat34	Decimal floating point numbers of this type are represented internally with 34 places in accordance with the IEEE-754-2008 standard. Valid values are numbers between 1E6145(1E-34 - 1) and -1E-6143 for the negative range, 0 and +1E-6143 and 1E6145(1 - 1E-34) for the positive range. Values lying between the ranges form the subnormal range and are rounded. Outside of the subnormal range, each 34-digit decimal number can be represented precisely using a decimal floating point number like this.	0
f	Binary floating point numbers are represented internally in accordance with the IEEE-754 standard (double precision). In ABAP, 17 places are represented (one integer digit and 16 decimal places). Valid values are numbers between -1.7976931348623157E+308 and -2.2250738585072014E-308 for the negative range and between +2.2250738585072014E-308 and +1.7976931348623157E+308 for the positive range, plus 0. Both validity intervals are extended in the direction of zero using subnormal numbers in accordance with the IEEE-754 standard.	0

Built-In Character-Like Types



The data objects of the character-like data types are used to handle character strings.

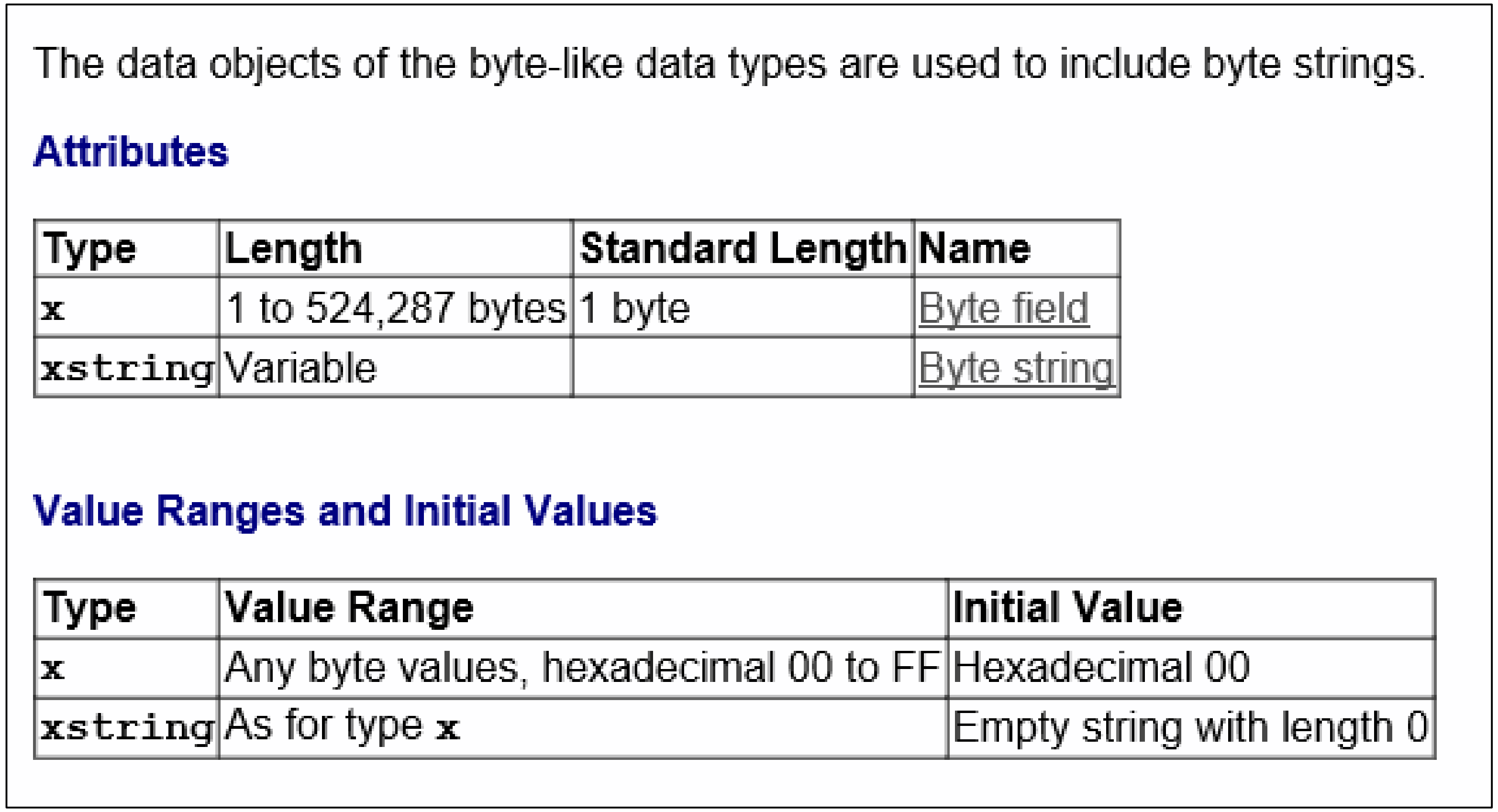
Properties

Type	Length	Default Length	Name
`c`	1 to 262,143 characters	One character	Text field
`n`	1 to 262,143 characters	One character	Numeric text field
`string`	Variable		Text string

Value Ranges and Initial Values

Type	Value Range	Initial Value
`c`	Any alphanumeric characters	" " for every place
`n`	Any alphanumeric characters, but the only valid values are the digits 0 to 9	"0" for every place
`string`	As for type `c`	Empty string with length 0

Built-In Byte-Like Types



The data objects of the byte-like data types are used to include byte strings.

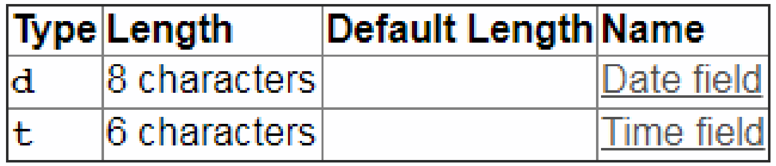
Attributes

Type	Length	Standard Length	Name
`x`	1 to 524,287 bytes	1 byte	Byte field
`xstring`	Variable		Byte string

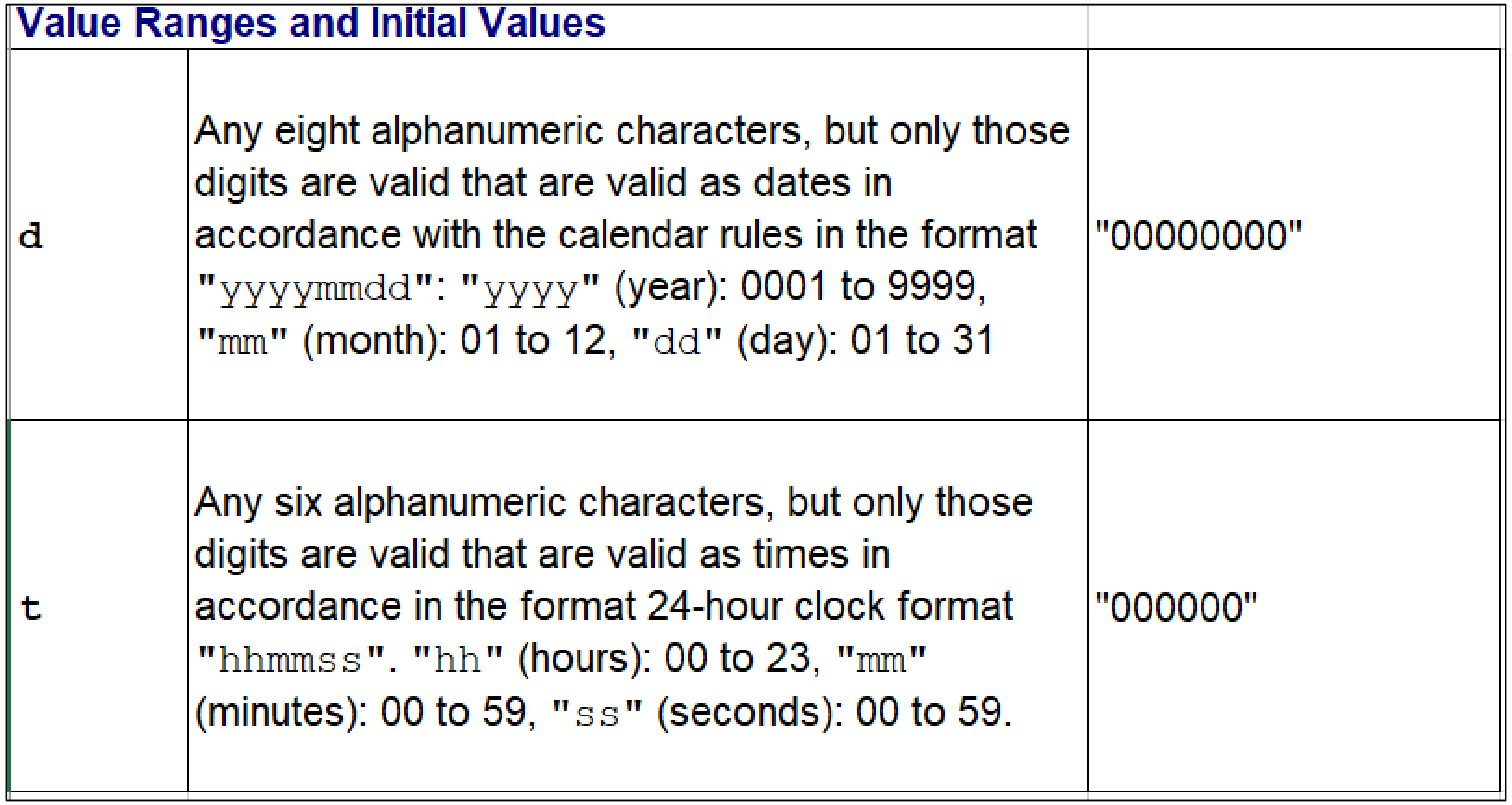
Value Ranges and Initial Values

Type	Value Range	Initial Value
`x`	Any byte values, hexadecimal 00 to FF	Hexadecimal 00
`xstring`	As for type `x`	Empty string with length 0

Built-In Date Types and Time Types



Type	Length	Default Length	Name
d	8 characters		Date field
t	6 characters		Time field



Value Ranges and Initial Values		
d	Any eight alphanumeric characters, but only those digits are valid that are valid as dates in accordance with the calendar rules in the format "yyyymmdd": "yyyy" (year): 0001 to 9999, "mm" (month): 01 to 12, "dd" (day): 01 to 31	"00000000"
t	Any six alphanumeric characters, but only those digits are valid that are valid as times in accordance in the format 24-hour clock format "hhmmss". "hh" (hours): 00 to 23, "mm" (minutes): 00 to 59, "ss" (seconds): 00 to 59.	"000000"

Example on Date and Time Data Types

```
REPORT z.
DATA DOJ(8) TYPE D VALUE '20181231'. "YYYYMMDD
DATA EXITTIME(6) TYPE T   VALUE '235945'. "HHMMSS
DOJ = SY-DATUM. “System data
EXITTIME = SY-UZEIT. “System Time
WRITE: 'The Date is ', DOJ. "DDMMYYYYY
WRITE:/'The Time is ', EXITTIME. "HHMMSS
```

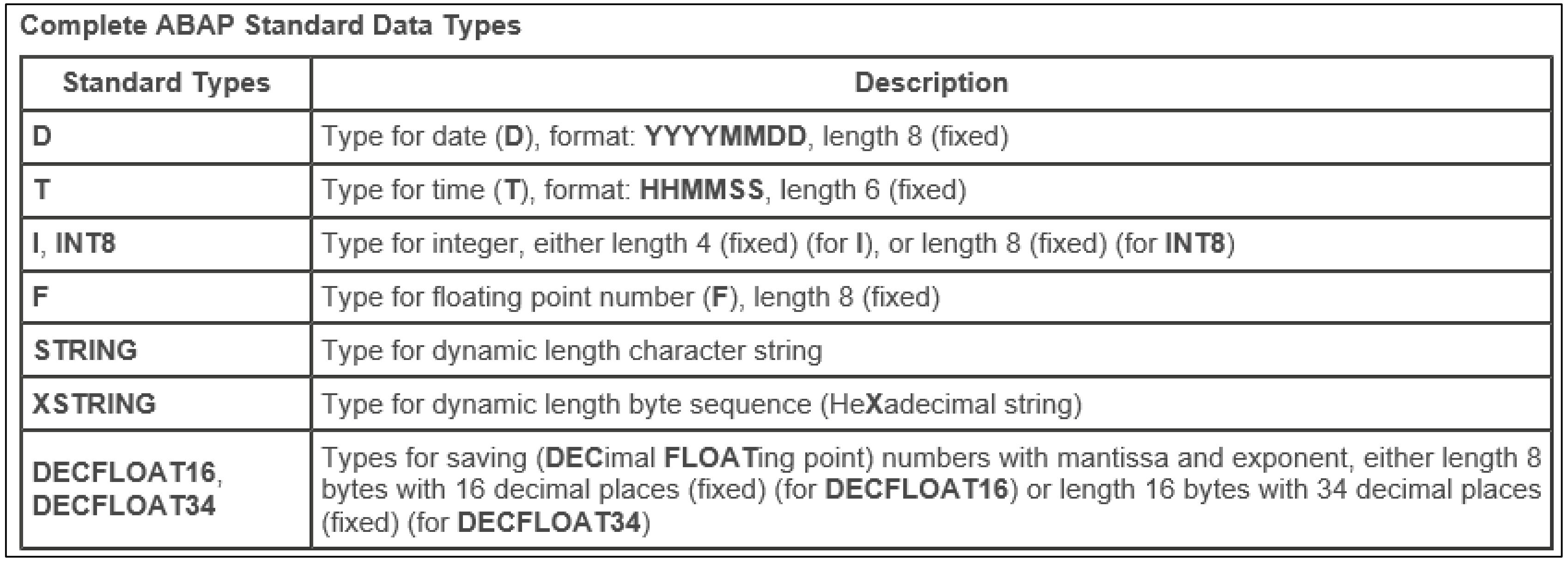
Groups of ABAP Standard Data Type

ABAP Standard Data Type(built-in) data types are divided into two groups:

- Complete
- Incomplete

Complete ABAP Standard Data Types

The built-in ABAP standard data types that already contain a type-specific, fixed-length byte sequence (Hexadecimal string)



Complete ABAP Standard Data Types

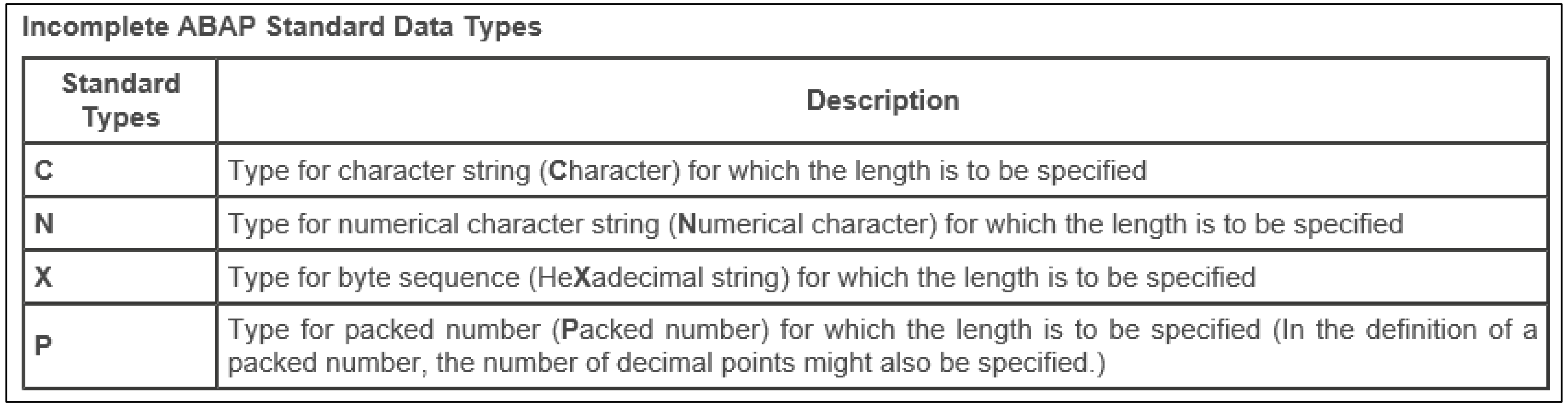
Standard Types	Description
D	Type for date (**D**), format: **YYYYMMDD**, length 8 (fixed)
T	Type for time (**T**), format: **HHMMSS**, length 6 (fixed)
I, **INT8**	Type for integer, either length 4 (fixed) (for **I**), or length 8 (fixed) (for **INT8**)
F	Type for floating point number (**F**), length 8 (fixed)
STRING	Type for dynamic length character string
XSTRING	Type for dynamic length byte sequence (He**X**adecimal string)
DECFLOAT16, **DECFLOAT34**	Types for saving (**DEC**imal **FLOAT**ing point) numbers with mantissa and exponent, either length 8 bytes with 16 decimal places (fixed) (for **DECFLOAT16**) or length 16 bytes with 34 decimal places (fixed) (for **DECFLOAT34**)

Demo

Demo of complete data types

Incomplete ABAP Standard Data Types

The standard types that do not contain a fixed length are considered incomplete data types. When they are used to define data objects, you need to specify the length of the variable.



Incomplete ABAP Standard Data Types

Standard Types	Description
C	Type for character string (**C**haracter) for which the length is to be specified
N	Type for numerical character string (**N**umerical character) for which the length is to be specified
X	Type for byte sequence (He**X**adecimal string) for which the length is to be specified
P	Type for packed number (**P**acked number) for which the length is to be specified (In the definition of a packed number, the number of decimal points might also be specified.)

Demo

Demo of incomplete data types

Categories of Data Types

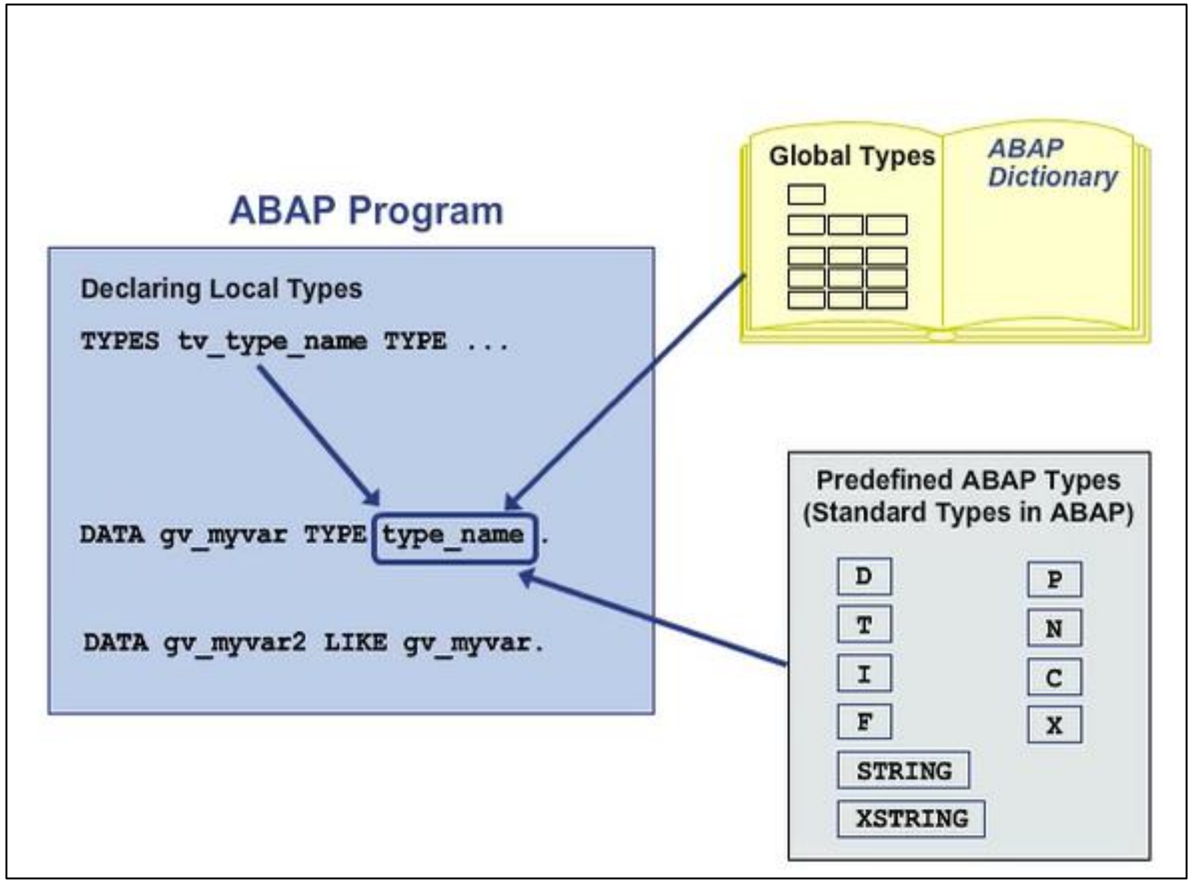
Categories of Data Types are as follows:

- Built-In
- Local
- Global

These types will be used to define variables (data objects).

Data objects are always defined with the DATA Keyword.

An ABAP standard type, local type or global type can be used to define the type for Data object.



ABAP Program
Declaring Local Types
TYPES tv_type_name TYPE ...
DATA gv_myvar TYPE type_name.
DATA gv_myvar2 LIKE gv_myvar.
Global Types
ABAP Dictionary
Predefined ABAP Types (Standard Types in ABAP)
D
T
I
F
STRING
XSTRING
P
N
C
X

Local Data Types

Using standard data types, you can declare local data types in the program

They are local to the program in which they are defined and cannot be reused in other programs

The declaration is made using the TYPES statement.

Example:

- TYPES number TYPE I.
  - DATA num1 TYPE number.
- TYPES length TYPE p decimals 2.
  - DATA mylen TYPE length
- TYPES code(3) TYPE c.
  - DATA mycode TYPE code.
- TYPES: text10 TYPE c LENGTH 10,
  ```
         text20 TYPE c LENGTH 20,
         number TYPE p LENGTH 8 DECIMALS 2.
  ```

Example – Creating user defined type

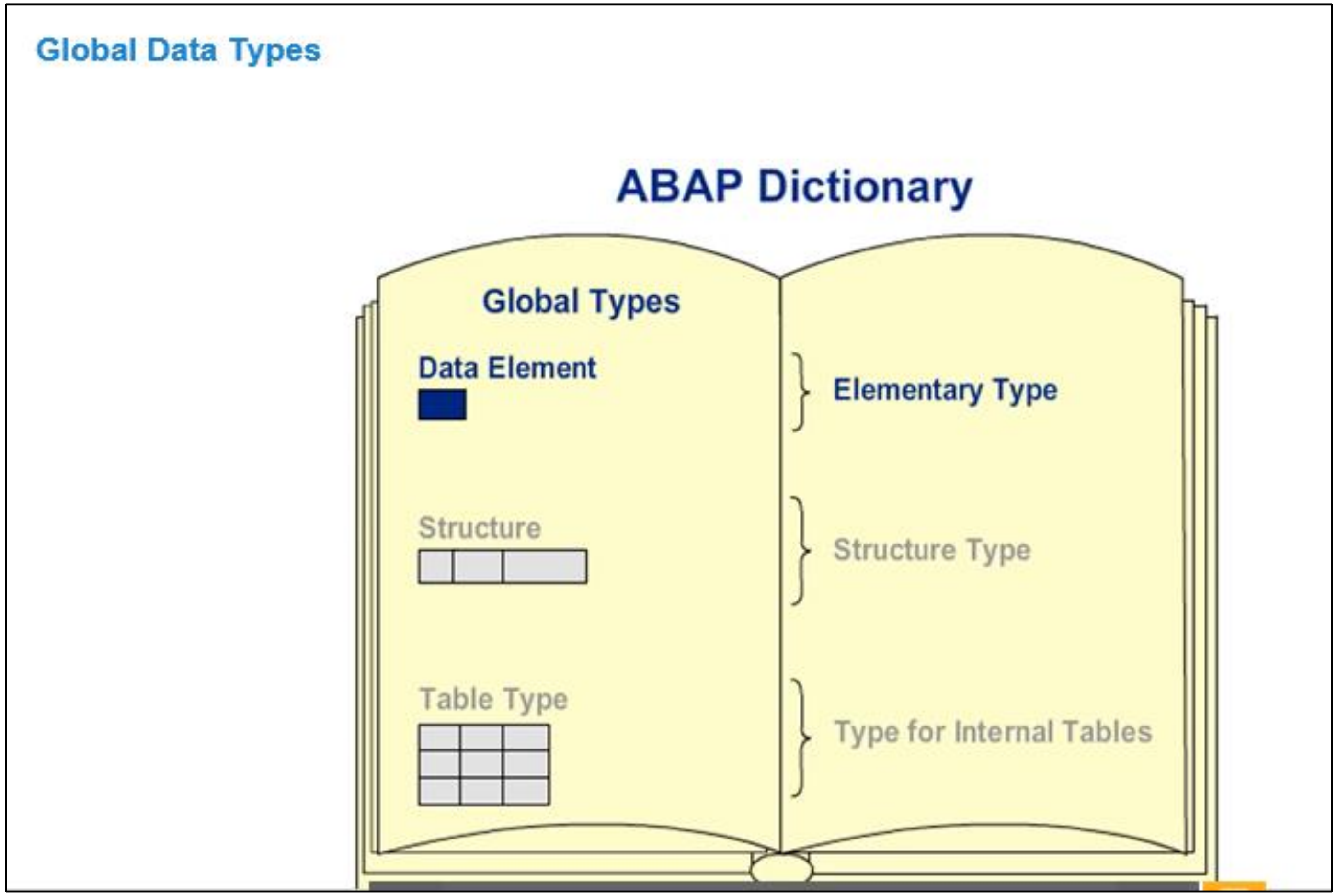
```
REPORT Z.
TYPES GENNAME(20) TYPE C. "USER DEFINED DATA TYPE
"cannot be assigned a value
DATA FIRSTNAME TYPE GENNAME VALUE 'Leena'.
DATA LASTNAME  TYPE GENNAME VALUE 'Agarwal'.
WRITE: / FIRSTNAME, LASTNAME.
```

Demo

Program on creating and using user defined Data Types

Global Data Types

A data type defined in the ABAP Dictionary is called global, as it can be used throughout the entire SAP system concerned.



Data Types (Contd.).

Reference Types

- Describes Data Objects that contain references to other objects
- No predefined references

Complex Types

- Are composed of other types
- Allow to manage and process related data under a single name
- No predefined complex Type in ABAP
- Further divided into
  - Structures – Field strings
  - Internal Tables

Data Types - Complex Data Types

Structures

- Sequence of any elementary types, reference types or complex data types
- Used in ABAP Programs to group work areas that logically belong together
- Example:

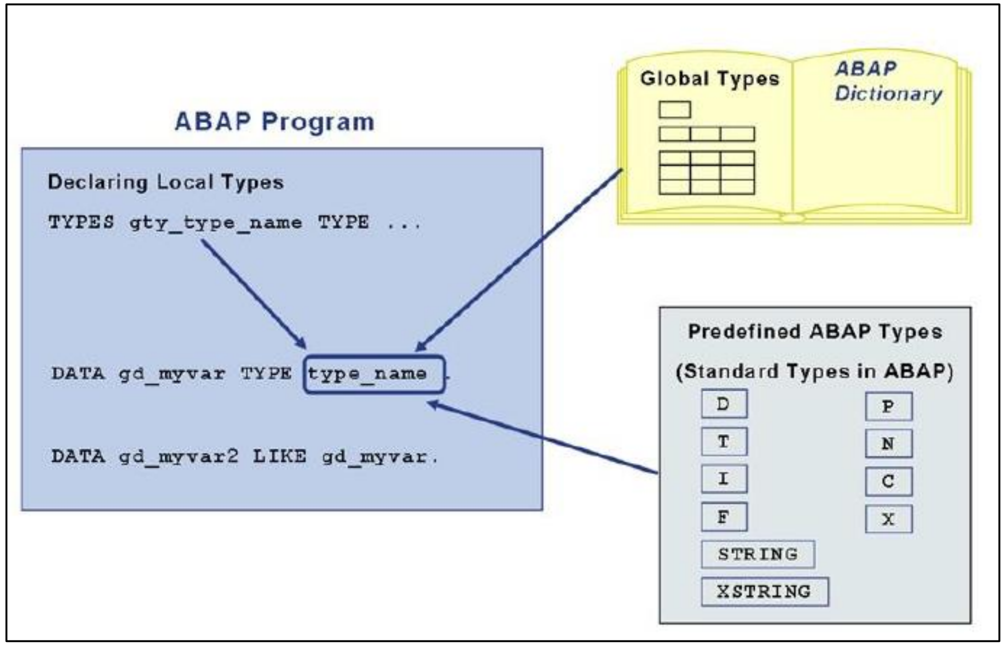
```
TYPES :  BEGIN OF address,
           name(20) TYPE c,
           street(30) TYPE c,
           city(20) TYPE c,
         END OF address.
```

Data objects

Data objects are usually defined with the DATA statement

After the name of the data object, a fully-specified type is assigned to it using the TYPE addition

They are local to the program in which they are defined, cannot be reused.



Data objects

ABAP contains the following kinds of data objects:

Literals

Named Data Objects

Data objects - Literals

Literals are not created by declarative statements.

Instead, they exist in the program source code.

Like all data objects, they have fixed technical attributes (field length, number of decimal places, data type), but no name.

They are therefore referred to as unnamed data objects.

Literals are unnamed data objects that you create within the source code of a program.

They are fully defined by their value.

You cannot change the value of a literal.

There are two types of literals: **numeric** and **character**.

Data objects - Literals

Numeric Literals

Examples of numeric literals:

```
123
-93
+456
```

Numeric literals in ABAP statements:

```
DATA number TYPE i VALUE -1234.
WRITE 6789.
MOVE 100 TO number.
```

Data objects - Literals

Character Literals

Character literals are sequences of alphanumeric characters in the source code of an ABAP program enclosed in single quotation marks.

Character literals enclosed in quotation marks have the predefined ABAP type **c** and are described as **text field literals**.

Examples of text field literals:

'Antony Smith'
'69190 Walldorf'

Data objects – Named data objects

Named Data Objects

Data objects that have a name that you can use to address the ABAP program are known as named objects. These can be objects of various types, including text symbols, variables and constants.

- Text Symbols are pointers to texts in the text pool of the ABAP program.
  - When the program starts, the corresponding data objects are generated from the texts stored in the text pool.
  - They can be addressed using the name of the text symbol.
- Variables are data objects whose contents can be changed using ABAP statements.
  - Variables are declared using the DATA, CLASS-DATA, STATICS, PARAMETERS, SELECT-OPTIONS, and RANGES statements.
- Constants are data objects whose contents cannot be changed.
  - They are declared constants using the CONSTANTS statement.

Data objects – Named data objects

A **text symbol** is a named data object of an **ABAP** program that is not declared in the program itself and instead is defined as a part of the **text elements** of the program.

Demo

Demo of Text Symbols

Data objects – Named data objects

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

Variables are declared using the DATA, CLASS-DATA, STATICS, PARAMETERS, SELECT-OPTIONS, and RANGES statements.

- Example:
- DATA name(20) TYPE c.
- Name = ‘Rupal’.
- Name = ‘Rohan’.

Data objects – Named data objects

Constants are data objects whose contents cannot be changed.

They are declared constants using the CONSTANTS statement.

- Example:
- CONSTANTS pi TYPE p DECIMALS 3 VALUE ‘3.141’.

Demo

Demo of Constants

Field Strings in ABAP

A field string is a series of fields grouped together under a common name.

It is equivalent of a structure in the DDIC but is defined within an ABAP/4 program.

The term structure in R/3 applies only to a Data Dictionary object containing a collection of fields.

The term field string applies to a collection of fields defined in an ABAP/4 program.

Two statements are usually used to define field strings in an ABAP/4 program:

- data
- tables

Demo

Program on using Field Strings

Data objects - Operations

Assigning Values

- MOVE
  - MOVE source TO destination

- Examples
  - MOVE ‘5.7’ TO number.
  - DATA : num1 TYPE i, num2 TYPE i.
  - num1 = 10.
  - MOVE num1 TO num2.

Data Objects – Operations (Contd.).

Assigning Values

- MOVE-CORRESPONDING
- MOVE-CORRESPONDING sourcestru TO deststru
  - Examples

```
DATA  : BEGIN OF  address,
                fname(15) TYPE c VALUE ‘Robert’,
                lname(15) TYPE c VALUE ‘David’,
              compname(30) TYPE c  VALUE ‘ CapGemini Tech’,
                number TYPE i VALUE ’72’,
              street(30) TYPE c VALUE ‘WhiteField’,
              city(10) TYPE c VALUE ‘BANGALORE’,
         END OF address.
DATA  : BEGIN OF name,
                lname(15) TYPE c ,
                fname(15) TYPE c ,
           END OF name.
MOVE-CORRESPONDING address TO name.
```

Demo

Program on Move Corresponding

Resetting Variables to Initial Values

CLEAR var.

- Resets var to appropriate initial value for its type
- Cannot use CLEAR to reset a CONSTANT
- Has different effect for different Data Types

CLEAR var.

Resetting Variables to Initial Values(Contd.).

- Example

```
DATA number TYPE i VALUE 10.
WRITE number.
CLEAR number.
WRITE  number.
```

- Output :   10   0

Arithmetic Operations

The following arithmetic operators are used in mathematical expressions:

Operato r	Meaning
+	Addition
_	Subtraction
*	Multiplication
/	Division
DIV	Integer division
MOD	Reminder of Integer Division
**	Powers

System Fields

ABAP system fields are always available in ABAP programs.

The runtime system fills them according to context.

They can then be used in programs to query the system status.

System fields are variables but they should be always treated as constants, and they should only be read.

The names and data types of the system fields are stored in the ABAP Dictionary in the SYST structure.

All system fields are addressed using SY field name.

System Fields

Few System Fields from Structure SY

SY-SUBRC

- Return code for ABAP statements
- 0 - if a statement is executed successfully

SY-UNAME

- Logon name of the user

SY-DATUM:

- Current System Date

SY-UZEIT:

- Current System Time

SY-TCODE

- Current Transaction

SY-INDEX

- Number of the current loop pass

Demo

Program on using System Fields

Write Statement

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

Syntax

```
WRITE {[AT] [/][pos][(len|*|**)]} dobj
      [UNDER other_dobj]
      [NO-GAP]
      [int_format_options]
      [ext_format_options]
      [list_elements]
      [QUICKINFO info].
```

Positioning Write on Output List

Syntax

WRITE AT [/][<POS>][(<LEN>)] <f>.

where '/' denotes a new line,

<pos> is a number or variable up to three digits long denoting the position on the screen.

<Len> is a number or variable up to three digits long denoting the output length.

Summary

In this lesson, you have learnt:

- The Need for ABAP
- The types of ABAP/4 Programs
- How to create, test and execute reports
- ABAP/4 Language Elements
- Chain Statement
- Data Types and Data Objects
- System Variables

Review Question

1. The _____ system variable displays the current ABAP program.
   - **Option 1:** SY-PRGNAME
   - **Option 2:** SY-REPID
   - **Option 3:** SY-PROG
   - **Option 4:** SY-PROG
2. _____ triggers a page break during list processing.