SAP ABAP Data Types -

What is Data Type?

Generally, several types of data available in the external world. Any language can't able to differentiate the data directly until the type was specified to the operating system. Every language has its own predefined data types to categorise and process the data. Like other languages, ABAP also has its predefined data types to differentiate and process each type separately.

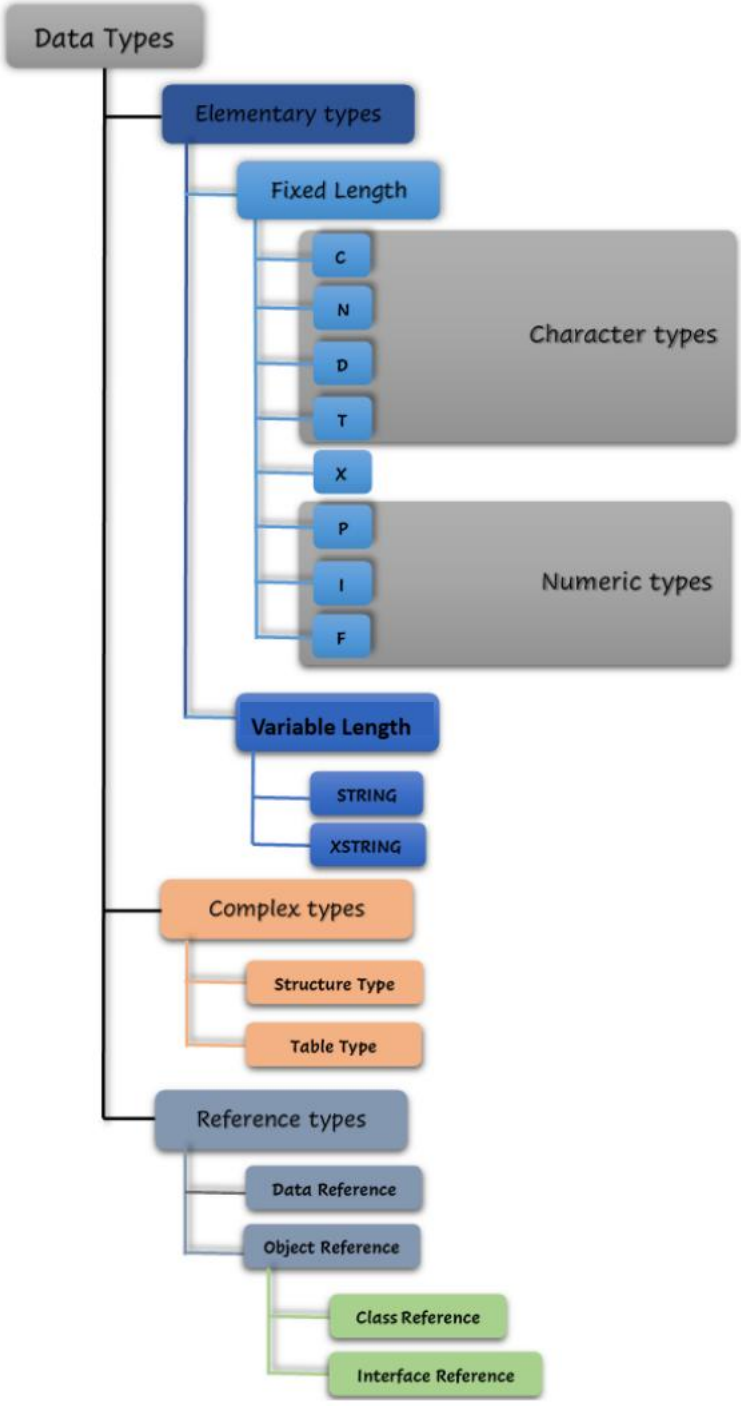
Data type is used to describe the technical characteristics the data. The data type is declared in the program using variables. A variable is a name that points to reserved memory location to save the data value. Based on the data type specified during the variable declaration, operating system allocates the memory and decide how the data can be stored in the memory.

Data type classification?

Data types are classified into three categories in ABAP. Those are -

Data Type	Description
Elementary types	Elementary types are the smallest individual units of types. Elementary type is a single data type used to define the data. In elementary type, only one data type is used to declare a variable for the data.
Complex types	Complex data types are created with the combination of elementary types. ABAP supports complex data types. Complex types allow us to manage and process conceptually-related data under a single name. Complex types can be processed as a whole or individually.
Reference types	Reference types specifies data objects that contains references/pointers to other objects. There are no predefined references. We must define them in a program.

Below tree representation specifies the detailed data types classification.



SAP ABAP Elementary Types -

Elementary types are the smallest individual units of types. Elementary type is a single data type used to define the data. In elementary type, only one data type is used to declare a variable for the data.

Elementary data types are divided into two types based on the legth of the declaration. Those are -

- Elementary fixed length data types
- Elementary variable length data types

Below table specifies the list of elementary data types -

Type	Description	Keyword
Fixed length data types	Text/Character field	C
	Numeric text	N
	Integers	I
	Packed number	P
	Floating point	F
	Date	D
	Time	T
	Byte/Hexadecimal field	X
Variable length data types	Text string	STRING
	XSTRING	XSTRING

Below table specifies the data type, length, range and description of each data type.

Data type	Length	Range	Description

Keyword			
C	1 character = 1 character	1 to 65535	Used for regular text information. Left justified and spaces padded to right. Default data type when none specified.
N	1 character = 1 character	1 to 65535	Used for set of digits. Right justified and zeroes padded to left.
I	4 bytes	-2147483648 to 2147483647	Used for integers. Right justified and zeroes padded to left.
P	8 bytes	[- 10 ^ (2len -1) + 1] to [+ 10 ^ (2len -1) + 1] (where length = fixed length)	Numbers stored in compressed format. Right justified and zeroes padded to left. Can be used for all calculations.
F	8 bytes	1E-307 to IE+307 positive or negative	Specified as floating point. Can be used for all calculations.
D	8 characters	8 characters	Used for internal representation of YYYYMMDD using Georgian calendar date. Can set default format in profile. Several output formats supported. Supports arithmetic operations.

T	6 characters	6 characters	Used to store the time. Format is HHMMSS. Several output formats supported. Supports arithmetic operations.
X	1 byte	Any byte values (00 to FF)	Used to store hexadecimal values in binary format. 2 hex digits stored in 1 byte.
STRING	Variable length	Any alphanumeric characters	Used for any alphanumeric characters.
XSTRING	Variable length	Any byte values (00 to FF)	Used for any alphanumeric characters stored in hex decimal format.

Example -

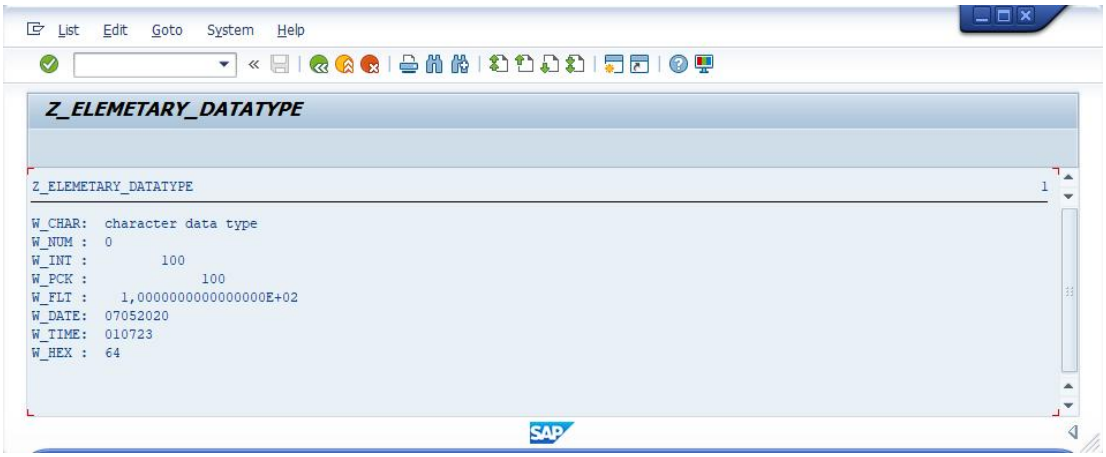
Write a program with all elementary data types.

Code -

```
*&---------------------------------------------------------------------**& Report
Z_ELEMETARY_DATATYPE*&---------------------------------------------------------------------**&
Program Written by ….*&---------------------------------------------------------------------*
REPORT  Z_ELEMETARY_DATATYPE.
* Declaring W_CHAR of character type of length 30 and *initialized with 'character data
type'.DATA W_CHAR(30) TYPE C VALUE 'character data type'.
* Declaring W_NUM of type Numeric with length of 1 byte.DATA W_NUM TYPE N VALUE
100.
* Declaring W_INT of integer type.DATA W_INT TYPE I VALUE 100.
* Declaring W_PCK of type compressed format.DATA W_PCK TYPE P VALUE 100.
* Declaring W_FLT of type Floating point.DATA W_FLT TYPE F VALUE 100.
* Declaring W_DATE of type Date.DATA W_DATE TYPE D.
* Declaring W_TIME of type Time.DATA W_TIME TYPE T.
* Declaring W_HEX of type hexadecimal binary format.DATA W_HEX TYPE X VALUE 100.
```

```
* Assinging current date from system variable.
W_DATE = SY-DATUM.
* Assinging current time from system variable.
W_TIME = SY-UZEIT.
* Displaying all variables.WRITE : 'W_CHAR: ', W_CHAR,
     / 'W_NUM : ', W_NUM,
     / 'W_INT : ', W_INT,
     / 'W_PCK : ', W_PCK,
     / 'W_FLT : ', W_FLT,
     / 'W_DATE: ', W_DATE,
     / 'W_TIME: ', W_TIME,
     / 'W_HEX : ', W_HEX.
```

Output -



Explaining Example -

W_CHAR is a character type of length 30 and left justified. So all the data gets displayed in the output.

W_NUM is numeric type, 1 character length and right justified. So the last digit of right side(0) gets displayed in the output.

W_INT is integer type, 4 bytes length and right justified. So the 100 value right justified and spaces padded to the left in remaining places.

W_PCK is packed number type, 8 bytes length and right justified. So the 100 value right justified and spaces padded to the left in remaining places.

W_FLT is floating type and 8 bytes length. So the result 100 displayed in exponential format.

W_DATE is date type and 8 characters length. So the result of run date displayed in DDMMYYYY format.

W_TIME is time type and 6 characters length. So the result of run time displayed in HHMMSS format.

W_HEX is binary type and displayed as hexa decimal value. So the result 100 displayed in hexadecimal format value 64.

SAP ABAP Complex Types -

Complex data types are created with the combination of elementary types. ABAP supports complex data types. Complex types allow us to manage and process conceptually-related data under a single name. Complex types can be processed as a whole or individually.

Complex data types are further divided into two types and those are –

- Structure data types
- Array type or Internal table data types

Structure Data Types -

Structure types are used to group the elements that logically belong together. The elements of structure can be a combination of any data type or same data type.

Syntax -

```
DATA: BEGIN OF {structure-name}
        {local variables declaration}
```

END OF {structure-name}.

Structures are classified as four types based on their definition and those are –

Structure Type	Description
Simple Structures	Contains elementary datatypes of fixed/variable length as elements and called as non-nested structures.
Nested structures	Contains one or more structures elements within a structure element.
Flat structures	Contains only elementary datatypes of fixed length.
Deep structures	Contains at least one internal table, reference type, string as component.

Example -

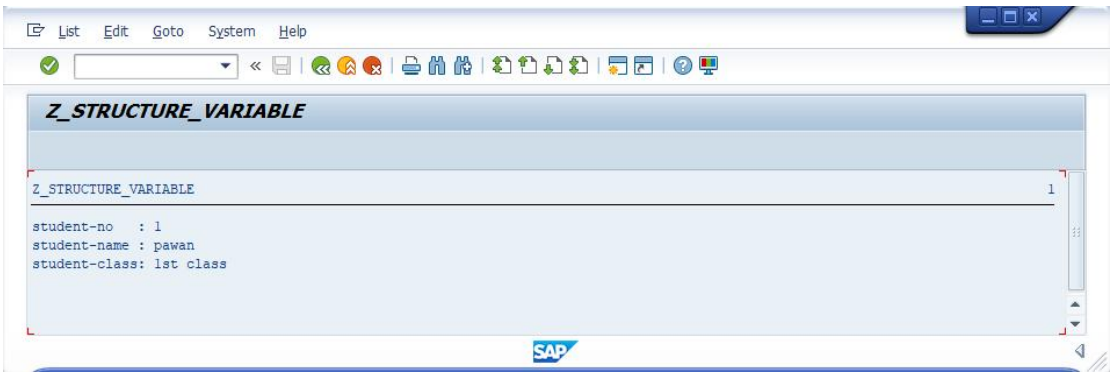
Write a simple program to get the understanding of Structure variable.

Code -

```
*&---------------------------------------------------------------------**& Report
Z_STRUCTURE_VARIABLE*&---------------------------------------------------------------------**&
Program Written by …*&---------------------------------------------------------------------*
REPORT  Z_STRUCTURE_VARIABLE.
* Declaring student structure with student no, student name and* student classDATA: BEGIN
OF student,
     no           TYPE n,
     name(25)      TYPE c,
     class(10)     TYPE c,
     END OF student.
* Assigning value to student noMOVE 1  TO student-no.
* Assigning value to student nameMOVE 'pawan' TO student-name.
* Assigning value to student classMOVE '1st class' TO student-class.
```

* Displaying student structure details by using stucture appendingWRITE : 'student-no :',
student-no,
/ 'student-name :', student-name,
/ 'student-class:', student-class.

Output -



Explaining Example -

student is a structure variable. **no**, **name** and **class** are elementary variables in **student** structure variable.

The structure variables are referring like - **no** as **student-no**, **name** as **student-name** and **class** as **student-class**.

These references **student-no**, **student-name** and **student-class** are used in programming process to manuplate the data.

Internal Table or Array Types -

Internal table contains series of lines which are repeated from single line. One line may contain single or multiple elements that are combination of same or different data types. Internal table has a line data type used to identify the table rows using unique or non-unique key. Table type determines how the table entries can be accessed. Internal table most advanced version of array and also called as array.

Syntax -

```
TYPES: BEGIN OF {table-name}
            {local variables declaration}
            END OF {table-name}.
DATA: {table-name1} TYPE {table-name} OCCURS n.
```

Example -

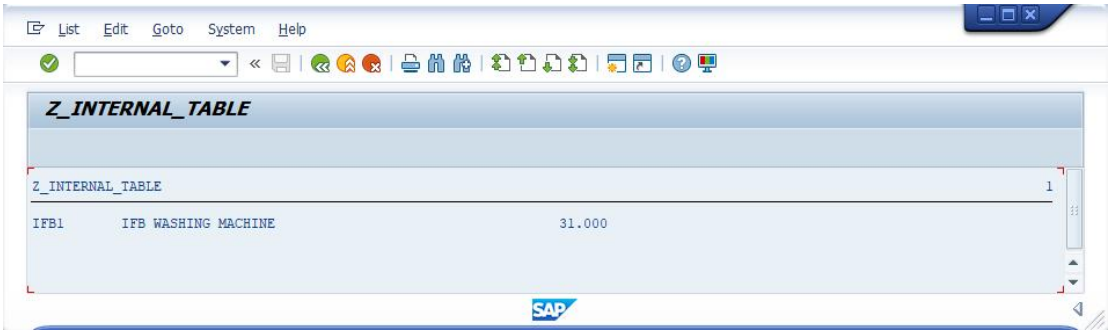
Simple example to create product information internal table with below structure.

Code -

```
REPORT  Z_INTERNAL_TABLE.
* internal table Structure creationTYPES: BEGIN OF t_product,
      pid(10)     TYPE C,
      pname(40)   TYPE C,
      pamount(10) TYPE P,
      END OF t_product.
* Data & internal table declarationDATA: wa TYPE t_product,
                wa1 TYPE t_product,
     it TYPE TABLE OF t_product.

wa-pid     = 'IFB1'.
wa-pname   = 'IFB WASHING MACHINE'.
wa-pamount = 31000.
* Appending data to the internal tableAPPEND wa TO it.
* Reading internal table of index 1READ TABLE it INTO wa1 INDEX 1.
IF sy-subrc = 0.
  WRITE: wa1-pid, wa1-pname, wa1-pamount.ELSE.
  WRITE 'No Record Found'.ENDIF.
```

Output -



Explaining Example -

TYPES: BEGIN OF t_product....	Structure declaration with pid(product id), pname(product name) and pamount(product amount)
DATA: wa TYPE t_product	Declaring Work area(wa) of type t_product
wa1 TYPE t_product	Declaring Work area1(wa1) of type t_product
it TYPE TABLE OF t_product	Internal table declaration of type t_product
wa-pid = 'IFB1'	Assigning value to pid(product id)
wa-pname = 'IFB WASHING MACHINE'	Assigning value to pname(product name)
wa-pamount = 31000	Assigning value to pamount(product name)
APPEND wa TO it	Assigning work area(wa) data to the table(it) data
READ TABLE it INTO wa1 INDEX 1	Reading internal table(it) with index 1 and assign it to work area1(wa1)
WRITE: wa1-pid, wa1-pname, wa1-pamount	Displaying the work area1 (wa1) using structure fields