Batch Script - Arrays

Arrays are not specifically defined as a type in Batch Script but can be implemented. The following things need to be noted when arrays are implemented in Batch Script.

- Each element of the array needs to be defined with the set command.
- The 'for' loop would be required to iterate through the values of the array.

Creating an Array

An array is created by using the following set command.

```
set a[0]=1
```

Where 0 is the index of the array and 1 is the value assigned to the first element of the array.

Another way to implement arrays is to define a list of values and iterate through the list of values. The following example show how this can be implemented.

Example

```
@echo off
set list = 1 2 3 4
(for %%a in (%list%) do (
   echo %%a
))
```

Output

The above command produces the following output.

```
1
2
3
4
```

Accessing Arrays

You can retrieve a value from the array by using subscript syntax, passing the index of the value

you want to retrieve within square brackets immediately after the name of the array.

Example

```
@echo off
set a[0]=1
echo %a[0]%
```

In this example, the index starts from 0 which means the first element can be accessed using index as 0, the second element can be accessed using index as 1 and so on. Let's check the following example to create, initialize and access arrays −

```
@echo off
set a[0] = 1
set a[1] = 2
set a[2] = 3
echo The first element of the array is %a[0]%
echo The second element of the array is %a[1]%
echo The third element of the array is %a[2]%
```

The above command produces the following output.

```
The first element of the array is 1
The second element of the array is 2
The third element of the array is 3
```

Modifying an Array

To add an element to the end of the array, you can use the set element along with the last index of the array element.

Example

```
@echo off
set a[0] = 1
set a[1] = 2
set a[2] = 3
Rem Adding an element at the end of an array
Set a[3] = 4
echo The last element of the array is %a[3]%
```

The above command produces the following output.

```
The last element of the array is 4
```

You can modify an existing element of an Array by assigning a new value at a given index as shown in the following example −

```
@echo off
set a[0] = 1
set a[1] = 2
set a[2] = 3
Rem Setting the new value for the second element of the array
Set a[1] = 5
echo The new value of the second element of the array is %a[1]%
```

The above command produces the following output.

```
The new value of the second element of the array is 5
```

Iterating Over an Array

Iterating over an array is achieved by using the 'for' loop and going through each element of the array. The following example shows a simple way that an array can be implemented.

```
@echo off
setlocal enabledelayedexpansion
set topic[0] = comments
set topic[1] = variables
set topic[2] = Arrays
set topic[3] = Decision making
set topic[4] = Time and date
set topic[5] = Operators

for /l %%n in (0,1,5) do (
   echo !topic[%%n]!
)
```

Following things need to be noted about the above program −

- Each element of the array needs to be specifically defined using the set command.
- The 'for' loop with the /L parameter for moving through ranges is used to iterate through the array.

Output

The above command produces the following output.

```
Comments
variables
Arrays
Decision making
Time and date
Operators
```

Length of an Array

The length of an array is done by iterating over the list of values in the array since there is no direct function to determine the number of elements in an array.

```
@echo off
set Arr[0] = 1
set Arr[1] = 2
set Arr[2] = 3
set Arr[3] = 4
set "x = 0"
:SymLoop

if defined Arr[%x%] (
   call echo %%Arr[%x%]%%
   set /a "x+=1"
   GOTO :SymLoop
)
echo "The length of the array is" %x%
```

Output

Output The above command produces the following output.

```
The length of the array is 4
```

Creating Structures in Arrays

Structures can also be implemented in batch files using a little bit of an extra coding for implementation. The following example shows how this can be achieved.

Example

```
@echo off
set len = 3
set obj[0].Name = Joe
set obj[0].ID = 1
set obj[1].Name = Mark
set obj[1].ID = 2
set obj[2].Name = Mohan
set obj[2].ID = 3
set i = 0
:loop

if %i% equ %len% goto :eof
set cur.Name=
set cur.ID=

for /f "usebackq delims==.tokens=1-3" %%j in (`set obj[%i%]`) do (
   set cur.%%k=%%l
)
echo Name = %cur.Name%
echo Value = %cur.ID%
set /a i = %i%+1
goto loop
```

The following key things need to be noted about the above code.

- Each variable defined using the set command has 2 values associated with each index of the array.
- The variable **i** is set to 0 so that we can loop through the structure will the length of the array which is 3.
- We always check for the condition on whether the value of i is equal to the value of **len** and if not, we loop through the code.
- We are able to access each element of the structure using the obj[%i%] notation.

Output

The above command produces the following output.

```
Name = Joe
Value = 1
```

```
Name = Mark
Value = 2
Name = Mohan
Value = 3
```