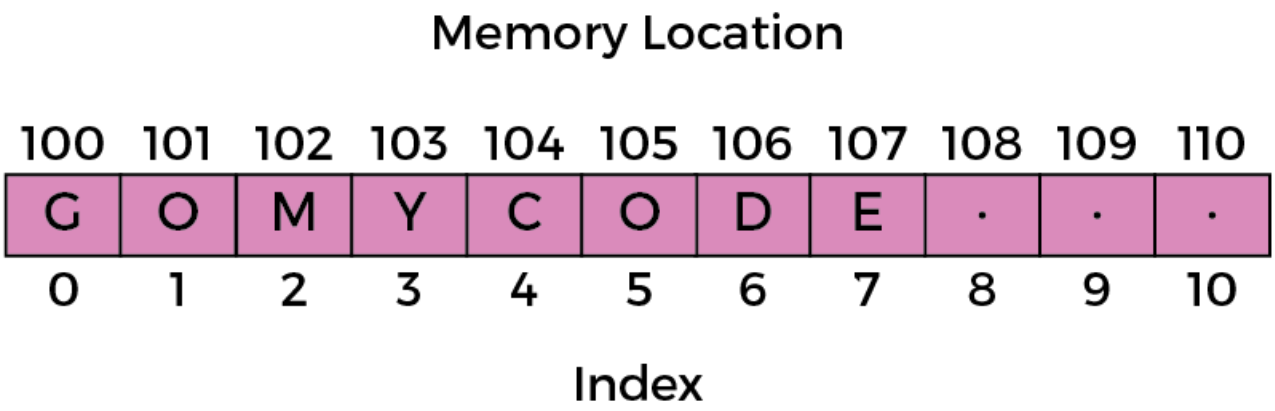
Current Skill Array.

Array.

An array is a collection of items stored at contiguous memory locations. The idea is to store multiple items of the same type together. This makes it easier to calculate the position of each element by simply adding an offset to a base value, i.e., the memory location of the first element of the array (generally denoted by the name of the array).



The above image can be looked as a top-level view of a staircase where you are at the base of the staircase. Each element can be uniquely identified by their index in the array

Why do we need arrays?

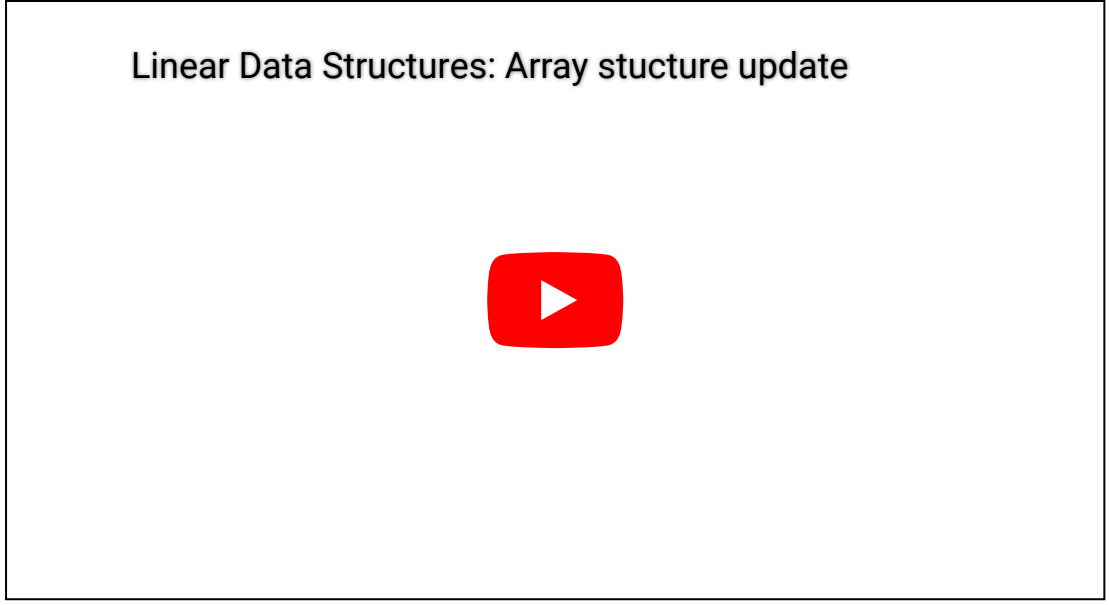
1. Here, are some reasons for using arrays:

- Arrays are best for storing multiple values in a single variable
- Arrays are better at processing many values easily and quickly
- Sorting and searching the values is easier in arrays

  Array in Algorithm :

  To declare the array, you need to specify a name and the type of data it will contain. To create it, you need to specify its length (the number of values), as array is a static data structure, which means you can not change its size once you declared.

```
array_name : ARRAY_OF type[length];
```



Practicing with arrays

After getting to know what are arrays, and what's the use of them, let's have some practice.

The first thing to start with any array is to know how to define it, and how to browse it. Let take the following code:

An array data structure, or simply an array is one of the most used data structure in programming.

Here, we will start by learning how to declare an array.

Let take the following code:

```
ALGORITHM traverse_array
VAR
   tab : ARRAY_OF INTEGER[5]; // declaring the array and define its length.
   i : INTEGER; // declaring the index for each element in the array.
BEGIN
  tab := {1,2,3,5,7,8}; // inserting the elements directly in the array.
  Write('Print Array :')
  FOR i FROM 0 TO 4 STEP 1  DO   // the index of arrays start from 0;
     Write(tab[i]) // to access the element, we use the name_array[indexation]
  END_FOR

// another version
   FOR i FROM 0 TO tab.length-1 STEP 1  DO
   // here we can replace the static
   //number in the max field using the `.length` property
```

```
        Write(tab[i])
    END_FOR
END
```

The second step is to see how we insert an element inside an array.

Remember that, when inserting elements in an array, we have to always make sure that we do not shrink the size of the array.

In the example below, we are going to ask the user to enter a new element, also its position in this array.

Let's take a look at the following code:

```
ALGORITHM insertion_array
VAR
   tab : ARRAY_OF INTEGER[5];
   i,j, pos, elt : INTEGER;
BEGIN
   FOR i FROM 0 TO 3 DO
       Read(tab[i]);// insert from user
   END_FOR

   Write("give the element to insert");
   Read(elt);
   Write("give the position to insert in array");
   Read(pos);
   j := tab.length;
   WHILE (j >= pos) DO
       tab[j+1] := tab[j]; // translation from left to right
       j := j-1; // update index
   END_WHILE

   tab[pos] := elt;
   // remember the tab.length is increased by 1.
END
```

The third step is to see how we can perform a search in an array.

There are multiple algorithms used to search an element in an array.

In this example, we are going to use a simple one.

Here are the steps that we should follow:

Start

Set J = 0

Repeat steps 4 and 5 while J < N

IF tab[J] is equal ITEM THEN GOTO STEP 6

Set J = J +1

PRINT J, ITEM

Stop

```
ALGORITHM search_array
VAR
   tab : ARRAY_OF INTEGER[5];
   i,j, pos, elt : INTEGER;
BEGIN
   FOR i FROM 0 TO 4 DO
       Read(tab[i]);// insert from user
   END_FOR

   Write("give the element to search of");
   Read(elt);

   j := 0;
   WHILE (j < tab.length) DO
       IF (tab[j] = elt) THEN
           BREAK; // element is found let break the loop
       END_IF
       j := j+1; // update index
   END_WHILE
```

```
    IF (j = tab.length) THEN  // we reached the end of array without finding the eleme
        pos := -1; // -1 means we don't find the element.
    ELSE
        pos := j;
    END_IF

    Write("The position of the element is ", pos);
END
```

The last step is to see how to delete an element from an array.

Here is the steps to follow when we want to delete an element from an array:

Start

Set J = K

Repeat steps 4 and 5 while J < N

Set LA[J] = LA[J + 1]

Set J = J+1

Stop

And here it is the structured algorithm:

```
ALGORITHM deletion_array
VAR
    tab : ARRAY_OF INTEGER[5];
    i,j, pos : INTEGER;
BEGIN
    FOR i FROM 0 TO 4 DO
        Read(tab[i]);// insert from user
    END_FOR

    Write("give the position to delete");
    Read(pos);

    j := pos;
    WHILE (j < tab.length) DO
```

```
        tab[j] := tab[j+1]; // translation from right to left
        j := j+1; // update index
    END_WHILE

    // remember the tab.length is decreased by 1.

END
```

Previous next