Problem Solving by Computer – Logical Thinking

Tech Capsule 3 – Array based problems

Arrays are used in programming to store and manipulate on large data.

Let us look at the below 2 examples to understand the need or use of arrays.

Example 1–
If we are required to read and find the average of 10 integer values, it can be very easily done with the use of loops (which we discussed in the previous capsule). In each iteration of the loop we add the new value to the sum, and at the end of 10 iterations we divide the sum by 10 to give the average. The important point to note in this algorithm is that we do not need to store the value of all the 10 integers entered by the user at the same time, but we must store only the most recently read value which is added to the sum.

Example 2–
Read 10 integers from the user to find their average and also find out how many integers are less than the average. Now to design the algorithm we must first find the average (like in Example 1) and then compare each number with the average to check if it less than the average or not. So we are not supposed to forget or loose the values entered by the user till we compare them with the average. Unlike Example 1 where we had to store only the most recently read value here we have to store all the 10 values.

Now, without using arrays we can solve this problem by having 10 integer variables to hold the values entered by the user. But then there will be a difficulty in writing the iterative loop as the variable names will have to change for every iteration of the loop. Arrays help us solve this problem as seen in our subsequent discussion.

What are Arrays?

1) Arrays are a mechanism that allow us to store and systematically manipulate large number of values.
2) An array is composed of a number (n) of elements of the same type.

Array Declarations:

Declaring arrays is done in the same way in all programming languages. Here let us look at how we can represent an array in algorithms.

<base_type> array_name[number of elements]

base_type refers to the type of elements the array will hold. It can be an integer type (int) or character type (char) etc. It is important to note that all elements of any given array will be of the same type.

array_name refers to the name given to the array. Any relevant name (based on the nature of data it holds or the operation it does) can be given.

number of elements is any non-negative integer value which is equivalent to the number of elements the array will hold.

Examples –

- *int numbers[10]*
- *char name[5]*
- *short data[2]*

Accessing and Manipulating Array Elements:

If an array A is declared to have 10 elements, these elements can be accessed as A[0], A[1], …, A[9].

In almost all programming languages the first element is the 0^{th} element of the array.

Each element of the array is accessed using a number called the "INDEX" and if an array has N elements the index value will range from 0 to N-1.

Problem-

Read 10 integers from the user to find their average and also find out how many integers are less than the average.

Solution –

Here our first step will be to declare required variables and arrays.

Step 1 –

```
int sum=0, numbers[10], i=0, count=0
float average=0          /*to hold the calculated average */
```

Step 2 –

Now we read the 10 elements from the user and also calculate the sum using an iterative loop "for"

```
for  i=0; i<10; i++
do
        read numbers[i]
        sum = sum + numbers[i]
done
```

Step 3 –

Average calculation

average = sum / 10

Step 4 –

To check how many elements are less than average value.

```
for  i=0; i<10; i++
do
        if numbers[i] < average
                count = count+1
done
```

Step 5 –

Print the final count value –

print count

Observe in Step 2 and Step 4 we use a variable called "i" as an index to access each element of the array starting from 0^{th} to (N-1)th element. We must take care to initialize the index value to 0 before starting to use it in the iterative loop.

Try by yourself

Activity 1:
Design an algorithm to accept 10 integer elements for an array and print the sum of all the elements.

Activity 2:
Design an algorithm to accept 20 integer elements for an array and print the maximum 3 and minimum 3 elements from the array.
Note: Sorting of array elements are not expected as part of this activity.

Activity 3:
Design an algorithm to accept 10 integer elements for an array and then rearrange the elements in the array in reverse order. The reversed array must be displayed as output.

Activity 4:
Design an algorithm which accepts a set of N (consider N to be 30) student's examination marks (in the range of 0 to 100). Then make a count of the number of students that obtain each possible mark (i.e.;

count of how many students scored 0, count of how many students scored 1, …. till count of how many students scored 100)

Activity 5:

Modify the algorithm written in **Activity 4** in order to get count of students in a specific range of marks as defined below.

Range of marks:

0 to 10%

11% to 20%

;

;

;

91% to 100%