Department of
Computer Science and Engineering

Title: Basic operations of one-dimensional and two-dimensional array

Data Structure Lab

CSE 106

Green University of Bangladesh

1 Objective(s)

- To gather knowledge on one and two dimensional arrays.
- To implement different operations on one and two dimensional arrays.
- To solve problems using one and two dimensional arrays.

2 Problem analysis

An array is analogous to a basket of different colored marbles, each occupying the same space, placed next to each other in the basket, having the same shape, but identifiable by their different colors. In Java, an array is nothing but an object that holds multiple elements of a single data type under a common variable name. These multiple elements stored under one variable name are indexed to specify their individual locations in the contiguous memory allocated. However, only a fixed set of elements can be stored in a Java array. The length of an array is established at the time of its creation and is unchangeable post creation. The length property can be used to find the number of elements stored in an array. Each element in the array can be accessed using its numerical index, beginning with 0. All the above said elements are accessed on Index basis, in order to access

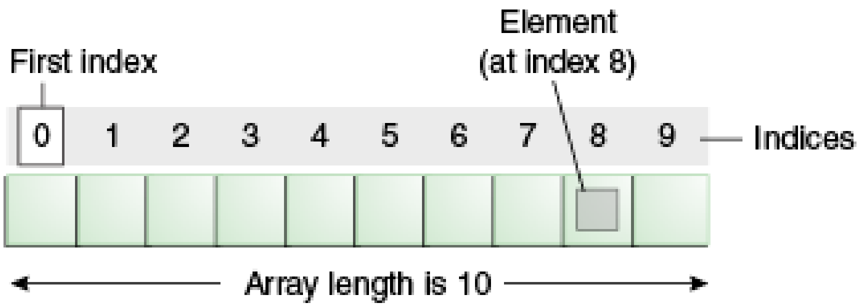

Figure 1: Basic illustration of array.

the very first element, we need to access 0th location and in order to access the last element we need to access the (n-1) location. We will be discussing following 2 types of array -

- One Dimensional Array
- Two Dimensional Array

3 One Dimensional Array

As mentioned above, a one dimensional array is a sequential collection of variables having the same datatype. We can store various elements in an array.

Now, let us try to understand how can we perform different operations on array.

3.1 Representation of One Dimensional Array

1D arrays are stored in the following manner:

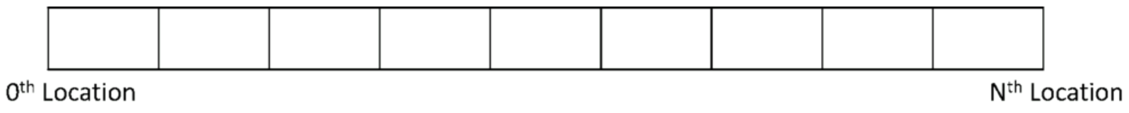

Figure 2: 1D array representation.

3.2 Declaring Array

```
datatype variableName[] ;
```

where,

datatype – The type of Data that the array will be holding, it determines the type of each element .Like an array of int, char, float , double , etc.

variableName– It's the name of the array , from which you want to identify the array.

[] (square brackets) –Thy are used to denote an array.

```
    byte byteArrayName[];
    short shortsArrayName[];
    boolean booleanArrayName[];
    long longArrayName[];
    float floatArrayName[];
    double doubleArrayName[];
    char charArrayName[];
```

3.3 Instantiating an Array

dataType variableName[] – This statement only declared an array, i.e only a reference is created.

In order to give memory to the array, we have to create or instantiate the array as below:

variableName = new dataType[size]

The above syntax, perform following 3 operations:

- Creates a new array of dataType[size]
- Assign the memory to the array with the help of the new keyword.
- Size represents the number of elements we want to store in an array.

3.4 Assigning Values to any array

We can follow either of the way to assign values to the array.

```
int[] arrayName[] = {1,2,3,4,5};
usingNewKeyword[0]=1;
usingNewKeyword[1]=1;
usingNewKeyword[2]=1;
usingNewKeyword[3]=1;
```

4 Two Dimensional Array

Till now we have seen how to work with the single dimensional arrays.

Now let's discuss about, Two dimensional array. It is arranged as an array of arrays. The representation of the elements is done in the form of rows & columns.

4.1 Representation of Two Dimensional Array

2D arrays are stored in the following format in the system:

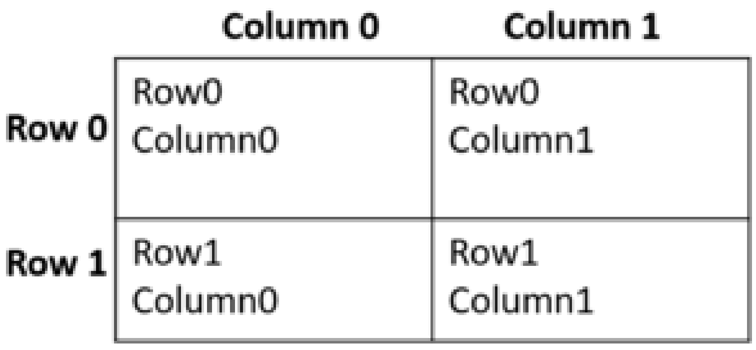

Figure 3: 2D array representation.

4.2 Declaring a Two Dimensional Array

```
datatype variableName[][]
```

where,

datatype – The type of Data that the array will be holding, it determines the type of each element. Like an array of int, char, float, double, etc.

variableName– It's the name of the array, from which you want to identify the array.

[][] – This is the symbol that shows the difference between the One Dimensional & Two Dimensional arrays.

The main difference between the One Dimensional and Two-dimensional array is [][],i.e We have two square brackets that help us in storing the data in the form of rows and columns.

4.3 Instantiating a Two Dimensional Array

With the following syntax, we can initialize the Two dimensional array.

```
variableName int[rowSize][columnSize]
```

where, [rowSize][columnSize] – It is used to define so as of how many no. of elements will be stored in a two-dimensional array, in the form of rows and columns.

4.4 Assigning Values to a Two dimensional array

```
dataType variableName[][]={
        {Row0valueColumn0Value, Row1valueColumn1Value ,},
        {Row1valueColumn0Value, Row1valueColumn1Value ,},
        .
        .
        .
           }
```

Example

```
    int variableName[][] = { { 1, 2 }, { 3, 4 } };
```

5 Implementation

Till now we have seen the ways to instantiate and assign the values to a different kind of array, but in the real environment, we need to access and manipulate these values which will help us in the multiple transactions.

Since all the elements in the array, let it be 1D or 2D, they are of the same data type. Due to this reason we usually use for loop to traverse the array and perform different operations in them.

5.1 Traversing or Accessing 1D array

With the following line of code, we could easily access or traverse through the ID array.

```
    int arrayName[5]= {10,20,30,40,50}; //Declare , initialize , Assigned values

    for(int i=0;i<5;i++) //Processing or Accessing each element of array
    {
        printf("Elemet %d is :%d", i arrayName[i]);
    }
```

5.1.1 Input/Output

```
Elemet 0 is :10
Elemet 1 is :20
Elemet 2 is :30
Elemet 3 is :40
Elemet 4 is :50
```

We are using the for loop, in order to iterate the Array that we created in the introduction section.

5.2 Traversing or Accessing 2D array

```
int arrayName[2][2] = { { 10, 20 }, { 30, 40 } };

        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++) {
                printf("Element at Row%dColumn%d is %d", i, j, arrayName[i][j]);
            }

            printf("\n");
        }
```

5.2.1 Input/Output

```
Element at Row0Column0 is 10
Element at Row0Column1 is 20

Element at Row1Column0 is 30
Element at Row1Column1 is 40
```

6 Discussion & Conclusion

Based on the focused objective(s) to understand about binary search, the additional lab exercise made me more confident towards the fulfilment of the objectives(s).

7 Lab Task (Please implement yourself and show the output to the instructor)

1. Write a program in C to copy the elements of one array into another array.

8 Lab Exercise (Submit as a report)

- Write a program in C to count the total number of duplicate elements in an array.
- Write a program in C for adding two matrices of the same size.

9 Policy

Copying from internet, classmate, seniors, or from any other source is strongly prohibited. 100% marks will be *deducted* if any such copying is detected.