Data Structures

IT 205

Dr. Manish Khare

Lecture – 4
11-Jan-2018

Notice

- Friday (12-Jan-2018)
  - EL114(Digital Logic Design) – (10 AM – 10:55 AM)
  - IT205(Data Structures) – (11 AM – 11:55 AM)
  - Extra Class for IT 205 (Data Structures)
    - 12 PM – 12:55 PM

Arrays – Basic Operations

- Following are the basic operations supported by an array.
  - **Traverse** − print all the array elements one by one.
  - **Insertion** − Adds an element at the given index.
  - **Deletion** − Deletes an element at the given index.
  - **Search** − Searches an element using the given index or by the value.
  - **Update** − Updates an element at the given index.

- **Traverse Operation**

  In traversing operation of an array, each element of an array is accessed exactly for once for processing. This is also called visiting of an array.

- **Insertion Operation**

  Insert operation is to insert one or more data elements into an array. Based on the requirement, new element can be added at the beginning, end or any given index of array.

- **Deletion Operation**

  Deletion refers to removing an existing element from the array and re-organizing all elements of an array.

- **Search Operation**

  You can perform a search for array element based on its value or its index.

- **Update Operation**

  Update operation refers to updating an existing element from the array at a given index.

Declaring Arrays

- To declare an array in C/C++, a programmer specifies the type of the elements and the number of elements required by an array as follows –
  - type arrayName [arraySize];

- This is called a *single-dimensional* array. The **arraySize** must be an integer constant greater than zero and **type** can be any valid C data type. For example, to declare a 10-element array called **balance** of type double, use this statement –
  - double balance[10];

- Here *balance* is a variable array which is sufficient to hold up to 10 double numbers.

Initializing Arrays

- You can initialize an array in C either one by one or using a single statement as follows –
  - double balance[5] = {1000.0, 2.0, 3.4, 7.0, 50.0};

- The number of values between braces { } cannot be larger than the number of elements that we declare for the array between square brackets [ ].
- If you omit the size of the array, an array just big enough to hold the initialization is created. Therefore, if you write –
  - double balance[] = {1000.0, 2.0, 3.4, 7.0, 50.0};

Accessing Array Elements

- An element is accessed by indexing the array name. This is done by placing the index of the element within square brackets after the name of the array. For example –
  - double salary = balance[9];

Initializing Two-Dimensional Arrays

- Multidimensional arrays may be initialized by specifying bracketed values for each row. Following is an array with 3 rows and each row has 4 columns.

  -
    ```
    int a[3][4] = { {0, 1, 2, 3} , /* initializers for row indexed by 0 */
                    {4, 5, 6, 7} , /* initializers for row indexed by 1 */
                    {8, 9,10, 11} /* initializers for row indexed by 2 */ };
    ```

- The nested braces, which indicate the intended row, are optional. The following initialization is equivalent to the previous example –

  - `int a[3][4] = {0,1,2,3,4,5,6,7,8,9,10,11};`

Accessing Two-Dimensional Array Elements

- An element in a two-dimensional array is accessed by using the subscripts, i.e., row index and column index of the array. For example −

```
int val = a[2][3];
```

- The above statement will take the 4th element from the 3rd row of the array.

Example

- 1. Program for declaration, assignment and accessing arrays.

- 2. Program for handle a two-dimensional array.

1-Dimensional Array

Insertion Operation in Array

- Insert operation is to insert one or more data elements into an array. Based on the requirement, a new element can be added at the beginning, end, or any given index of array.
- Algorithm
  - Let **LA** be a Linear Array (unordered) with **N** elements and **K** is a positive integer such that **K<=N**. Following is the algorithm where ITEM is inserted into the Kth position of LA –
    - 1. Start
    - 2. Set J = N
    - 3. Set N = N+1
    - 4. Repeat steps 5 and 6 while J >= K
    - 5. Set LA[J+1] = LA[J]
    - 6. Set J = J-1
    - 7. Set LA[K] = ITEM
    - 8. Stop

Example

- 3. Program for Insertion operation in Arrays

- We have learnt how the insertion operation works. It is not always necessary that an element is inserted at the end of an array. Following can be a situation with array insertion −
  - Insertion at the beginning of an array
  - Insertion at the given index of an array
  - Insertion after the given index of an array
  - Insertion before the given index of an array

Delete Operation in Array

- Deletion refers to removing an existing element from the array and re-organizing all elements of an array.
- Algorithm
  - Consider **LA** is a linear array with **N** elements and **K** is a positive integer such that **K<=N**. Following is the algorithm to delete an element available at the Kth position of LA.

```
1. Start
2. Set J = K
3. Repeat steps 4 and 5 while J < N
4. Set LA[J] = LA[J + 1]
5. Set J = J+1
6. Set N = N-1
7. Stop
```

Example

- 4. Program for Deletion operation in Arrays

Search Operation in Array

- You can perform a search for an array element based on its value or its index.
- Algorithm
  - Consider **LA** is a linear array with **N** elements and **K** is a positive integer such that **K<=N**. Following is the algorithm to find an element with a value of ITEM using sequential search.

```
1. Start
2. Set J = 0
3. Repeat steps 4 and 5 while J < N
4. IF LA[J] is equal ITEM THEN GOTO STEP 6
5. Set J = J +1
6. PRINT J, ITEM
7. Stop
```

Example

- 5. Program for Search operation in Arrays

Update Operation in Array

- Update operation refers to updating an existing element from the array at a given index.
- Algorithm
  - Consider **LA** is a linear array with **N** elements and **K** is a positive integer such that **K<=N**. Following is the algorithm to update an element available at the Kth position of LA.

```
1. Start
2. Set LA[K-1] = ITEM
3. Stop
```

Example

- 6. Program for Update operation in Arrays

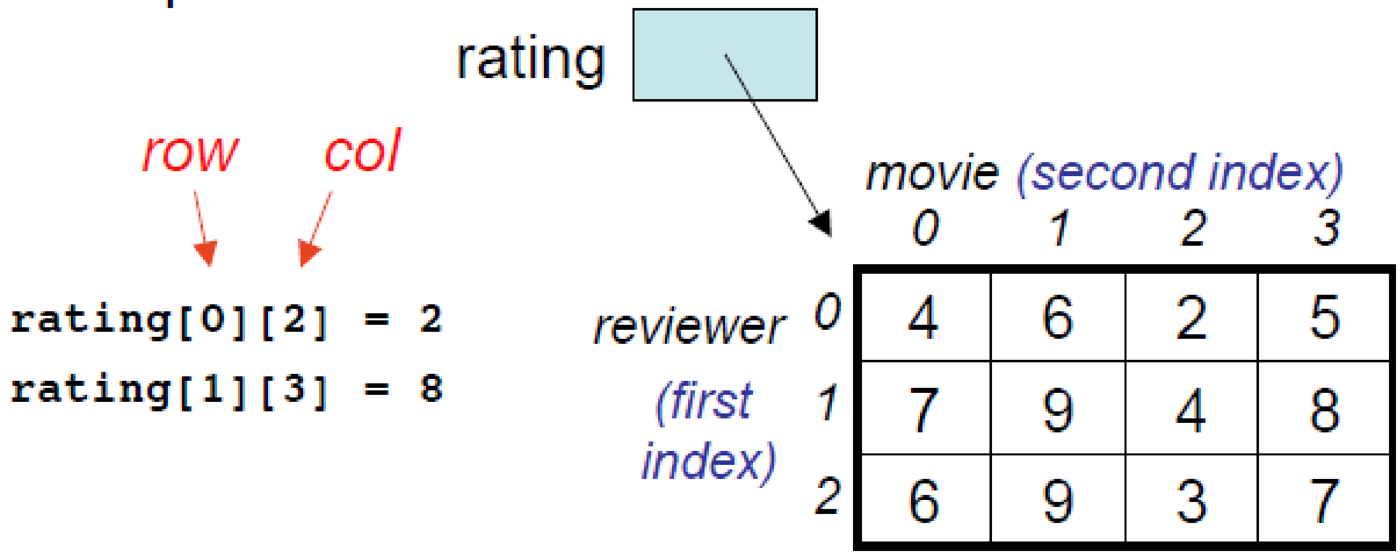
- Find Largest Element of an Array

- Calculate Average Using Arrays

2-Dimensional Array

2-Dimensional Arrays

- Two-dimensional (2D) arrays are indexed by two subscripts, one for the row and one for the column.
- Example:



```
rating[0][2] = 2
rating[1][3] = 8
```

	0	1	2	3
0	4	6	2	5
1	7	9	4	8
2	6	9	3	7

Similarity with 1-D Arrays

- Each element in the 2D array must by the same type,
  - either a primitive type or object type.
- Subscripted variables can be use just like a variable:

```
rating[0][3] = 10;
```

- Array indices must be of type `int` and can be a literal, variable, or expression.

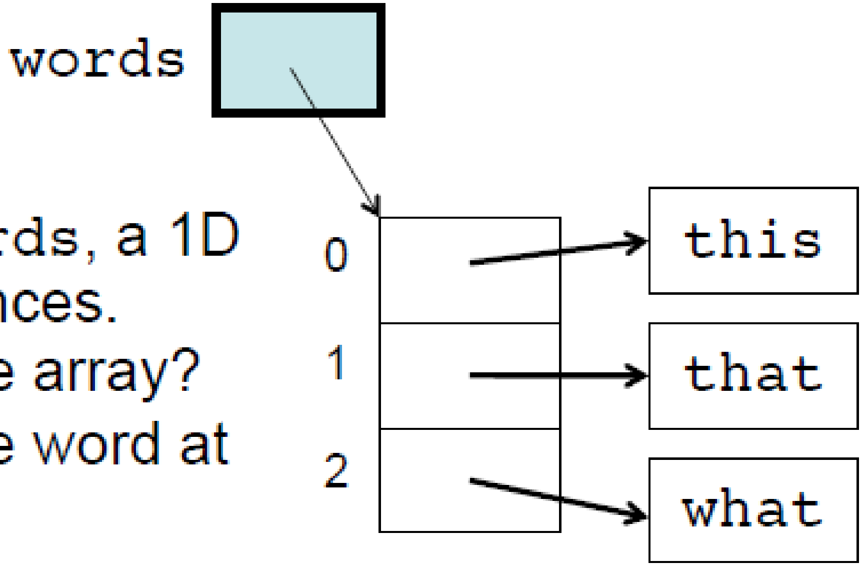
```
rating[3][j] = j;
```

- If an array element does not exists, the Java runtime system will give you an

```
ArrayIndexOutOfBoundsException
```

Size of 2-D Arrays

- When you write a method that has a 2D array as a parameter, how do you determine the size of the array?



Hint:

- Consider a variable `words`, a 1D array of `String` references.
- What is the length of the array?
- What is the length of the word at index 2?

- Addition of two 2-D Array
- Multiplication of two 2-D Array
- Traverse a 2-D Array
- Transpose a 2-D Array

Addition of two 2-D Array

- Add ( ):
  - Description: Here A is a two – dimensional array with M rows and N columns and B is a two – dimensional array with X rows and Y columns. This algorithm adds these two arrays.

```
1. If $(M \neq X)$ or $(N \neq Y)$ Then
    2. Print: Addition is not possible.
    3. Exit
[End of If]
4. Repeat For $I = 1$ to $M$
    5. Repeat For $J = 1$ to $N$
        6. Set $C[I][J] = A[I][J] + B[I][J]$
    [End of Step 5 For Loop]
[End of Step 6 For Loop]
7. Exit
```

- First, we have to check whether the rows of array A are equal to the rows of array B or the columns of array A are equal to the columns of array B. if they are not equal, then addition is not possible and the algorithm exits. But if they are equal, then first for loop iterates to the total number of rows i.e. M and the second for loop iterates to the total number of columns i.e. N. In step 6, the element A[I][J] is added to the element B[I][J] and is stored in C[I][J] by the statement:
    - C[I][J] = A[I][J] + B[I][J]

- 7. Program for Addition of two matrices

Multiplication of two 2-D Array

- Multiply ( ):
  - Description: Here A is a two – dimensional array with M rows and N columns and B is a two – dimensional array with X rows and Y columns. This algorithm multiplies these two arrays.

```
1. If (M ≠ Y) or (N ≠ X) Then
2. Print: Multiplication is not possible.
3. Else
4. Repeat For I = 1 to N
5. Repeat For J = 1 to X
6. Set C[I][J] = 0
7. Repeat For K = 1 to Y
8. Set C[I][J] = C[I][J] + A[I][K] * B[K][J]
[End of Step 7 For Loop]
[End of Step 5 For Loop]
[End of Step 4 For Loop]
[End of If]
9. Exit
```

- First we check whether the rows of A are equal to columns of B or the columns of A are equal to rows of B. If they are not equal, then multiplication is not possible. But, if they are equal, the first for loop iterates to total number of columns of A i.e. N and the second for loop iterates to the total number of rows of B i.e. X. In step 6, all the elements of C are set to zero. Then the third for loop iterates to total number of columns of B i.e. Y. In step 8, the element A[I][K] is multiplied with B[K][J] and added to C[I][J] and the result is assigned to C[I][J] by the statement:

    - C[I][J] = C[I][J] + A[I][K] * B[K][J]

- 8. Program for Multiplication of two matrices

Traverse a 2-D Array

- Traverse ( ):
- Description: Here A is a two – dimensional array with M rows and N columns. This algorithm traverses array A and applies the operation PROCESS to each element of the array.

```
1. Repeat For I = 1 to M
2. Repeat For J = 1 to N
3. Apply PROCESS to A[I][J]
[End of Step 2 For Loop]
[End of Step 1 For Loop]
4. Exit
```

- The first for loop iterates from 1 to M i.e. to the total number of rows and the second for loop iterates from 1 to N i.e. to the total number of columns. In step 3, it applies the operation PROCESS to the elements of the array A.

- 9. Program for Traverse of matrices

Transpose a 2-D Array

- Transpose ( ):
- Description: Here A is a two – dimensional array with M rows and N columns. This algorithm transposes the array.

```
1. Repeat For I = 1 to M
2. Repeat For J = 1 to N
3. Set B[J][I] = A[I][J]
[End of Step 2 For Loop]
[End of Step 1 For Loop]
4. Exit
```

- The first for loop iterates from 1 to M i.e. total number of rows and second for loop iterates from 1 to N i.e. total number of columns. In step 3, the element at location A[I][J] is assigned to B[J][I] by the statement B[J][I] = A[I][J].

- 10. Program for Transpose of matrices

Linked List