



Data Structure & Algorithm

Lecture 3 Array

Chhoeum Vantha, Ph.D.
Telecom & Electronic Engineering

Content

- Characteristic of Array
- Types of Arrays
- Array Operations
 - Insertion
 - Deletion
 - searching
- Disadvantages of Using Arrays

Characteristic of Array

Characteristic of Array

- Array is a **static data structure** that represents a collection of a **fixed number** of **homogeneous data** items or
- A fixed-size indexed sequence of elements, all of the **same type**.
- The individual elements are typically **stored in consecutive memory locations**.

Characteristic of Array

- The **length** of the array is determined when the array is created, and cannot be changed.
- The array is the most commonly used data storage structure;
- It's built into most programming languages.

Types of Arrays

Types of Arrays


- One-dimensional array: only one index is used
- Multi-dimensional array: array involving more than one index
- Static array: the compiler determines how memory will be allocated for the array
- Dynamic array: memory allocation takes place during execution

One Dimensional Static Array

- Syntax:
- `DataType arrayName [CAPACITY];`
- `DataType arrayName [CAPACITY] = {
initializer_list };`
- Example in C++:
 - `int b [5];`
 - `int b [5] = {19, 68, 12, 45, 72};`

Two Dimensional Static Array

- dimensional array can be seen as a table with 'x' rows and 'y' columns
- the row number ranges from 0 to (x-1)
- the column number ranges from 0 to (y-1).

	Column 0	Column 1	Column 2 
Row 0	<code>x[0][0]</code>	<code>x[0][1]</code>	<code>x[0][2]</code>
Row 1	<code>x[1][0]</code>	<code>x[1][1]</code>	<code>x[1][2]</code>
Row 2	<code>x[2][0]</code>	<code>x[2][1]</code>	<code>x[2][2]</code>

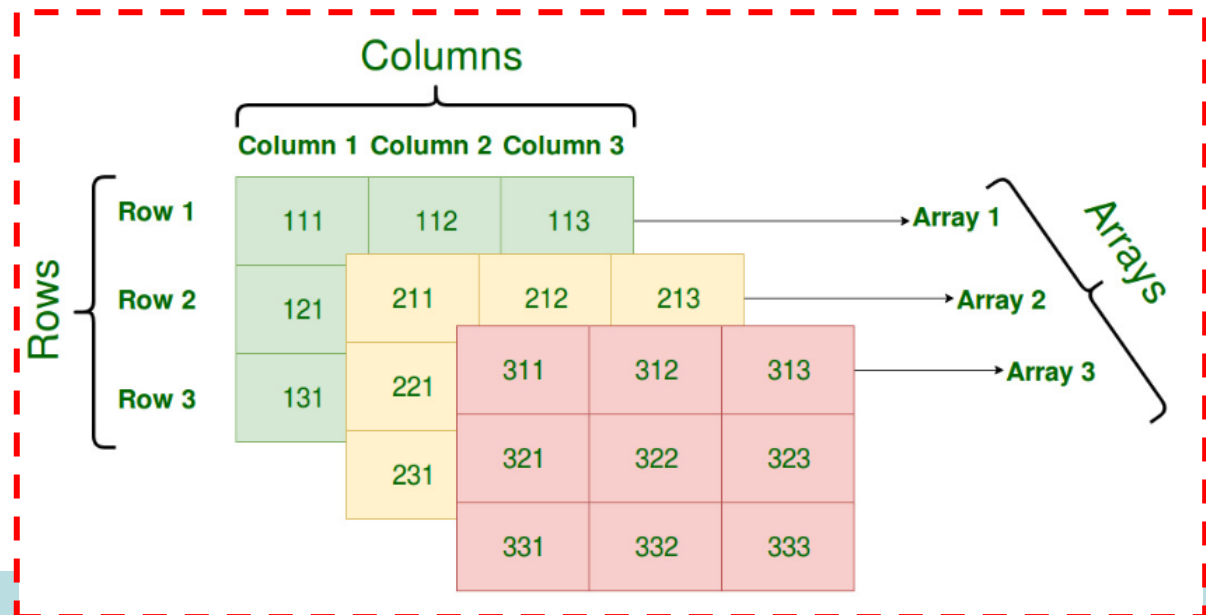
Two Dimensional Static Array

- First Method:
 - `int x[3][4] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11}`
 - 3 rows and 4 columns.
- Second Method:
 - `int x[3][4] = {{0,1,2,3}, {4,5,6,7}, {8,9,10,11}};`
- Third Method:

```
int x[3][4];
for(int i = 0; i < 3; i++){
    for(int j = 0; j < 4; j++){
        cin >> x[i][j];
    }
}
```

Three Dimensional Static Array

- Initialization in a Three-Dimensional array is the same as that of Two-dimensional arrays.
- The difference is as the number of dimensions increases so the number of nested braces will also increase.



Three Dimensional Static Array

- Method 1:

```
int x[2][3][4] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23};
```

- Method 2:

```
int x[2][3][4] =  
{  
    { {0,1,2,3}, {4,5,6,7}, {8,9,10,11} },  
    { {12,13,14,15}, {16,17,18,19}, {20,21,22,23} }  
};
```

Array Operations

Array Operations: Insertion

- Adding an element to the array

- Beginning

- Middle

- End

- Position

1	Brown		1	Brown
2	Davis		2	Davis
3	Johnson		3	Johnson
4	Smith		4	Smith
5	Wagner		5	Wagner
6			6	Ford
7			7	
8			8	

➤ Insert *Ford* at the *End* of array

Array Operations: Insertion

- Suppose, we have the following array:

Index	0	1	2	3	4	5	6	7	8	9
Value	12	10	7	43	26	83				

- Insert value **99** to position (index) 2, insertion process:

Index	0	1	2	3	4	5	6	7	8	9
Value	12	10	99	7	43	26	83			

Diagram illustrating the insertion process. The value 99 is inserted at index 2. Elements from index 3 onwards are shifted one position to the right. The elements being shifted are 7, 43, 26, and 83, which are shown in circles below the array.

Array Operations: Insertion

- How do you do it in C++?
- Write a tree step to insert **17** to array at position **1**



Microsoft Word
Document

Array Operations: Deletion

- Removing an element to the array

- Beginning
- Middle
- End
- Position

1	Brown	1	Brown
2	Davis	2	Davis
3	Ford	3	Ford
4	Johnson	4	Johnson
5	Smith	5	Smith
6	Taylor	6	Taylor
7	Wagner	7	
8		8	

➤ Insert *Wagner* at the *End* of the array


Array Operations: Deletion

- For the following array:

Index	0	1	2	3	4	5	6	7	8	9
Value	12	10	99	7	43	26	83			

- Delete an element example number 7 from an array:

Index	0	1	2	3	4	5	6	7	8	9
Value	12	10	99	43	26	83				



Array Operations: Deletion

- How do you do it in C++?
- Write a tree step to delete ... to array at position...

Array Operations: Searching

- Looking for elements which match given number:

Linear search

Array

6	3	0	5	1	2	8	-1	4
---	---	---	---	---	---	---	----	---

Element to search: 8

Array Operations: Searching

- Non-duplicate search by input value in the array
 - Check an input value with each value of elements in the array, in case an input value is equal to the value of any element of the array, the procedure search is finished (break)
- Duplicate search by input value in the array
 - Check an input value with every value (till the end element) of elements in the array, to find, how many elements of the array are equal to the input value?

Array Operations: Searching

- How do you do it in C++?

Disadvantages of Using Arrays

- Need to define a size for array
 - High overestimate (waste of space)
- insertion and deletion is very slow
 - need to move elements of the list
- redundant memory space
 - it is difficult to estimate the size of array

W3 – Lab 3

Exercise

Create an array to store data of any type, you want (int, double, string, char, float,...)

- Create a function to **show elements** of the array
- Create a function to **insert an element** to array to the position, which is input by the user
- Create a function to **delete an element** (by position, which is input by the user) from the array;

Exercise .Cont

Create an array to store data of any type, you want (int, double, string, char, float,...)

- Create a function to **delete all elements** with **value**, which is input by the user, from an array;
- Create a function to **search** (non-duplicate and duplicate) element(s) in the array by **value**, which is **input** by the user.

Thanks!