មហាវិទ្យាល័យវិស្វកម្ម
FACULTY OF ENGINEERING

Data Structure & Algorithm I

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

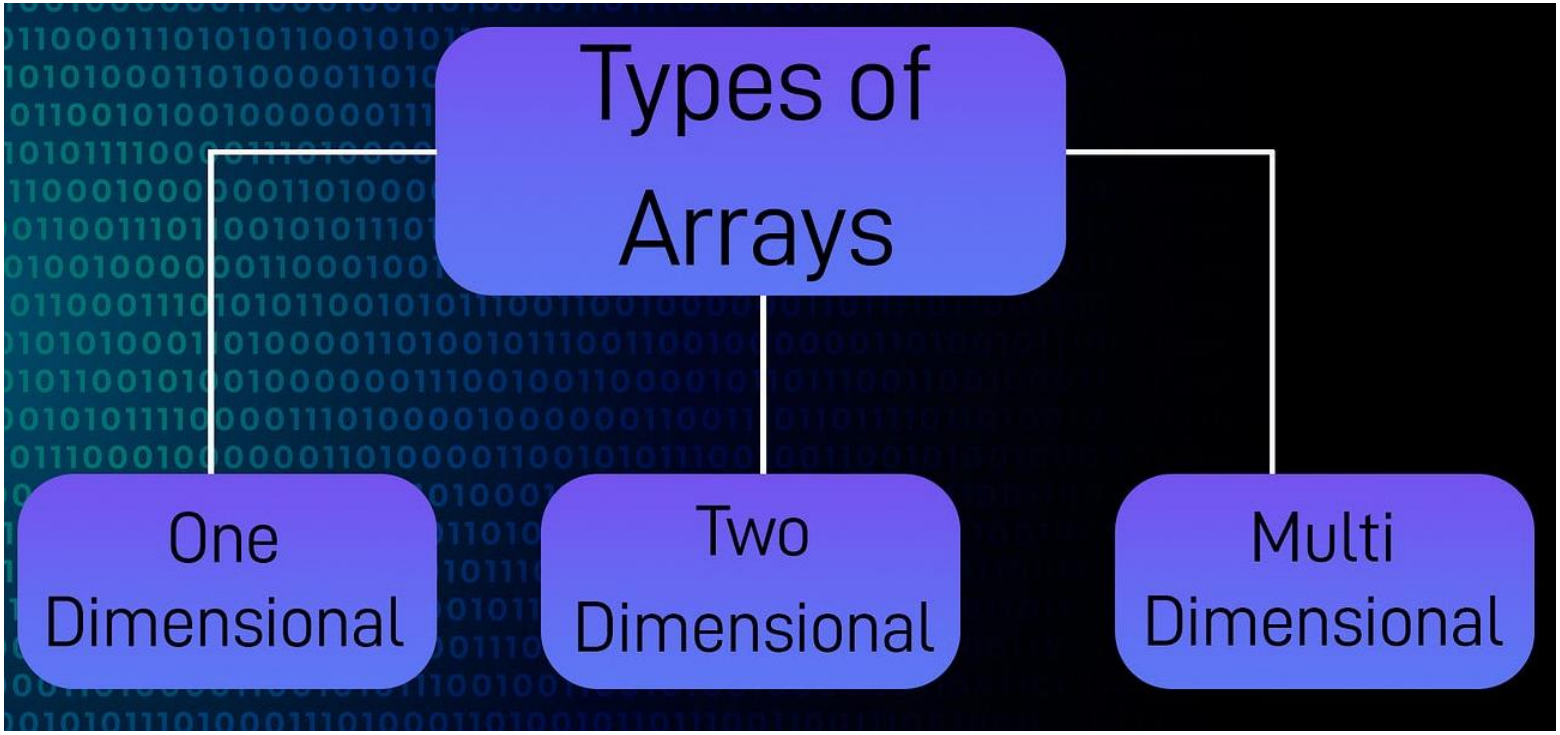
- Array is a static data structure that represents a collection of a fixed number of homogeneous data items or
- A fixed-size indexed sequence of elements, all of the same type.
- The individual elements are typically stored in consecutive memory locations.

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

- One-dimensional array: only one index is used
- Multi-dimensional array: array involving more than one index



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	x[0][0]	x[0][1]	x[0][2]
Row 1	x[1][0]	x[1][1]	x[1][2]
Row 2	x[2][0]	x[2][1]	x[2][2]

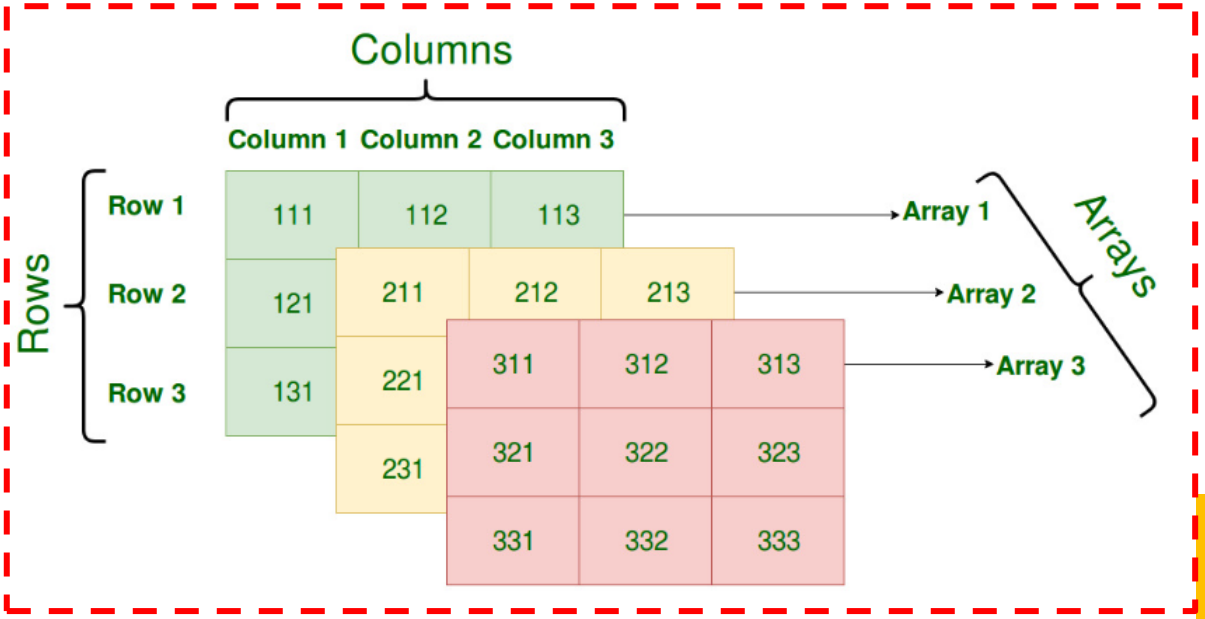
Two Dimensional Static Array

- 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:
  - iint 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:

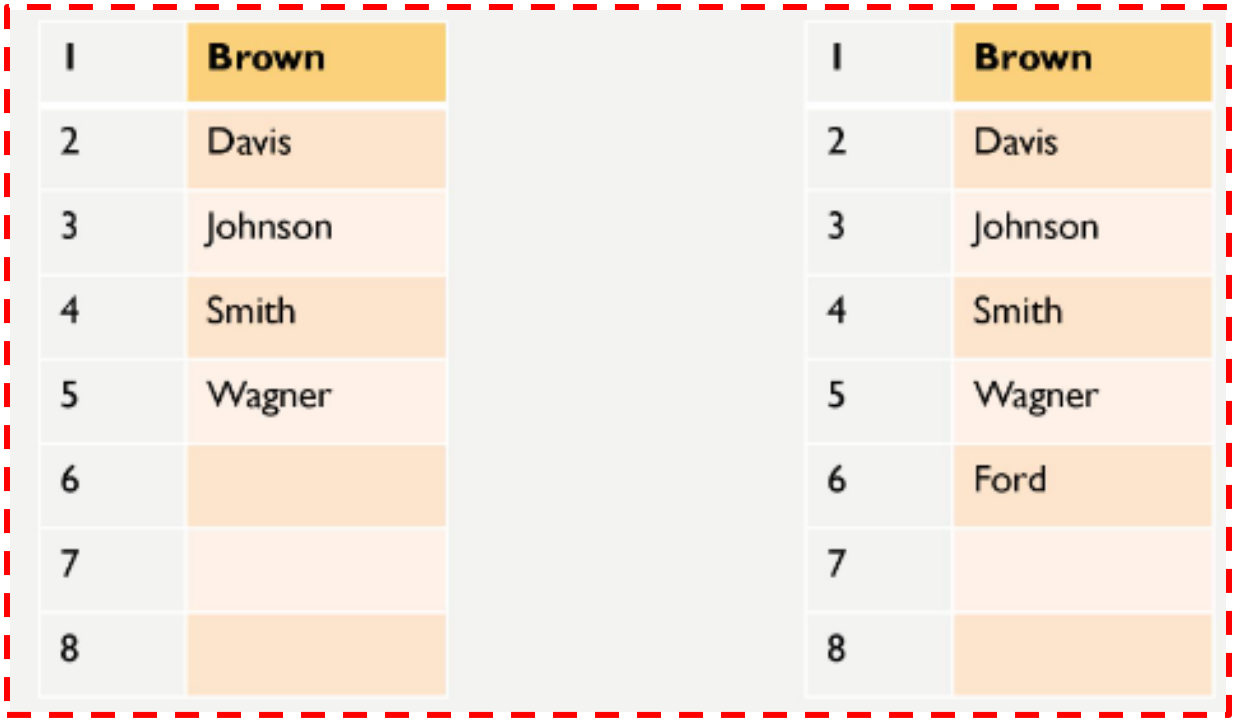
```
int x[2][3][4] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
                  11, 12, 13, 14, 15, 16, 17, 18, 19,
                  20, 21, 22, 23};
```

- 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: Insertion

- Adding an element to the array
  - Beginning
  - Middle
  - End
  - Position



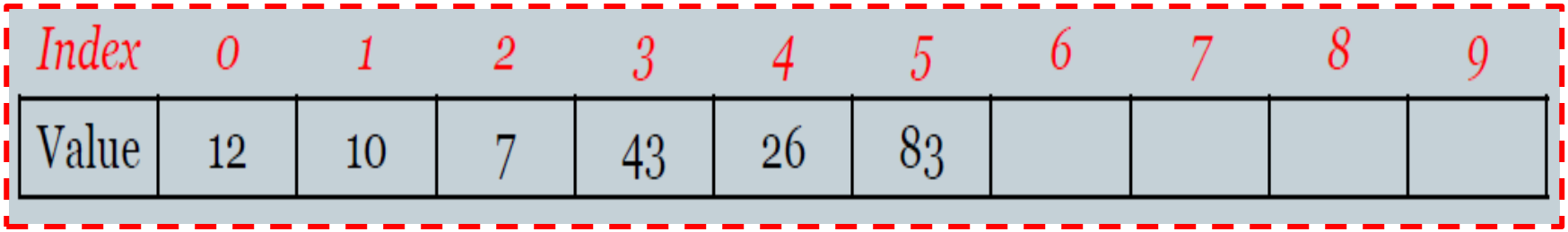
1	Brown
2	Davis
3	Johnson
4	Smith
5	Wagner
6	
7	
8	

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

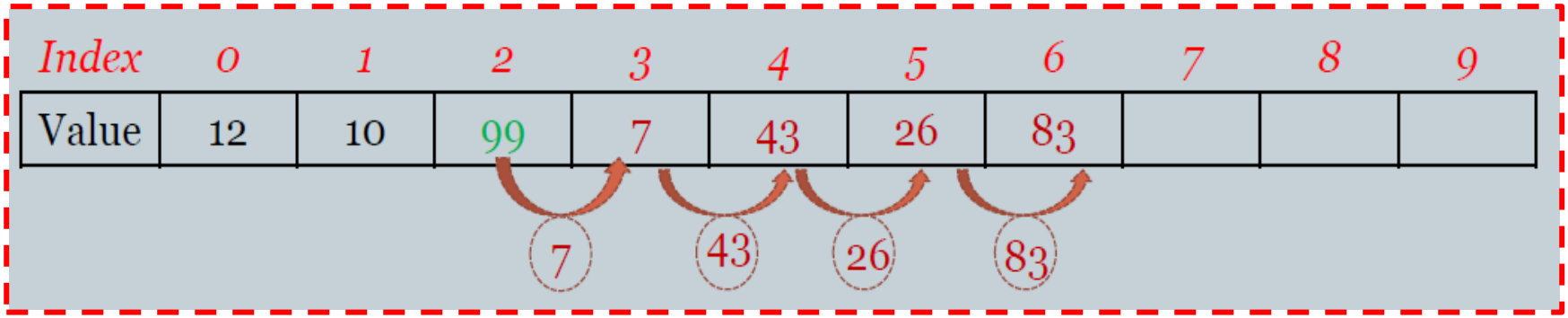
➢ Insert *Ford* at the *End* of array

Array Operations: Insertion

- Suppose, we have the following array:



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



Array Operations: Insertion

```
1   #include<iostream>
2   using namespace std;
3   int main()
4   {
5       int arr[10], i, elem, pos, arr_size;
6       cout<<"Enter the Size for Array: ";
7       cin>>arr_size;
8       cout<<"Enter "<<arr_size<<" Array Elements: ";
9       // insert element to array
10      for(i=0; i<arr_size; i++)
11      {
12          cin>>arr[i];
13      }
```

Array Operations: Insertion

```
14      // Ask what element and position of array want to insert
15      cout<<"\nEnter Element to Insert: ";
16      cin>>elem;
17      cout<<"At What Position ? ";
18      cin>>pos;
19      // adding new element regarding to position
20      for(i=arr_size; i>pos; i--)
21      {
22          arr[i] = arr[i-1];
23      }
24      arr[i] = elem;
25      arr_size++;
```

Array Operations: Insertion

```
26      // display array element
27      cout<<"\nThe New Array is:\n";
28      for(i=0; i<arr_size; i++)
29      {
30          cout<<arr[i]<<"   ";
31      }
32      cout<<endl;
33      return 0;
34  }
```

Array Operations: Insertion

```
for(i=arr_size; i>pos; i--)
{
    arr[i] = arr[i-1];
}
arr[i] = elem;
arr_size++;
```

Example:

arr_size = 5

arr

0	1	2	3	4	index
8	7	2	4	3	value

elem = 17

Pos = 1

i = 5; 5>1; i = 4

arr[5] = arr [4]

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

Tree-step for insertion in an array:

Array Operations: Insertion

```
for(i=arr_size; i>pos; i--)
{
    arr[i] = arr[i-1];
}
arr[i] = elem;
arr_size++;
```

Example:

arr_size = 5

arr

0	1	2	3	4
8	7	2	4	3

elem = 17

Pos = 1

i = 4; 4>1; i = 3

arr[4] = arr [3]

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

Tree-step for insertion in an array:

Array Operations: Insertion

```
for(i=arr_size; i>pos; i--)
{
    arr[i] = arr[i-1];
}
arr[i] = elem;
arr_size++;
```

Example:

arr_size = 5

arr

0	1	2	3	4
8	7	2	4	3

elem = 17

Pos = 1

i = 3; 3>1; i = 2

arr[3] = arr [2]

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

Tree-step for insertion in an array:

Array Operations: Insertion

```
for(i=arr_size; i>pos; i--)
{
    arr[i] = arr[i-1];
}
arr[i] = elem;
arr_size++;
```

Example:

arr_size = 5

arr

0	1	2	3	4
8	7	2	4	3

elem = 17

Pos = 1

i = 2; 2>1; i = 1

arr[2] = arr [1]

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

Tree-step for insertion in an array:

Array Operations: Insertion

```
for(i=arr_size; i>pos; i--)
{
    arr[i] = arr[i-1];
}
arr[i] = elem;
arr_size++;
```

Example:

arr_size = 5

arr

0	1	2	3	4
8	7	2	4	3

elem = 17

Pos = 1

i = 1; 1>1; i = 0=> F

Arr[1] = 17

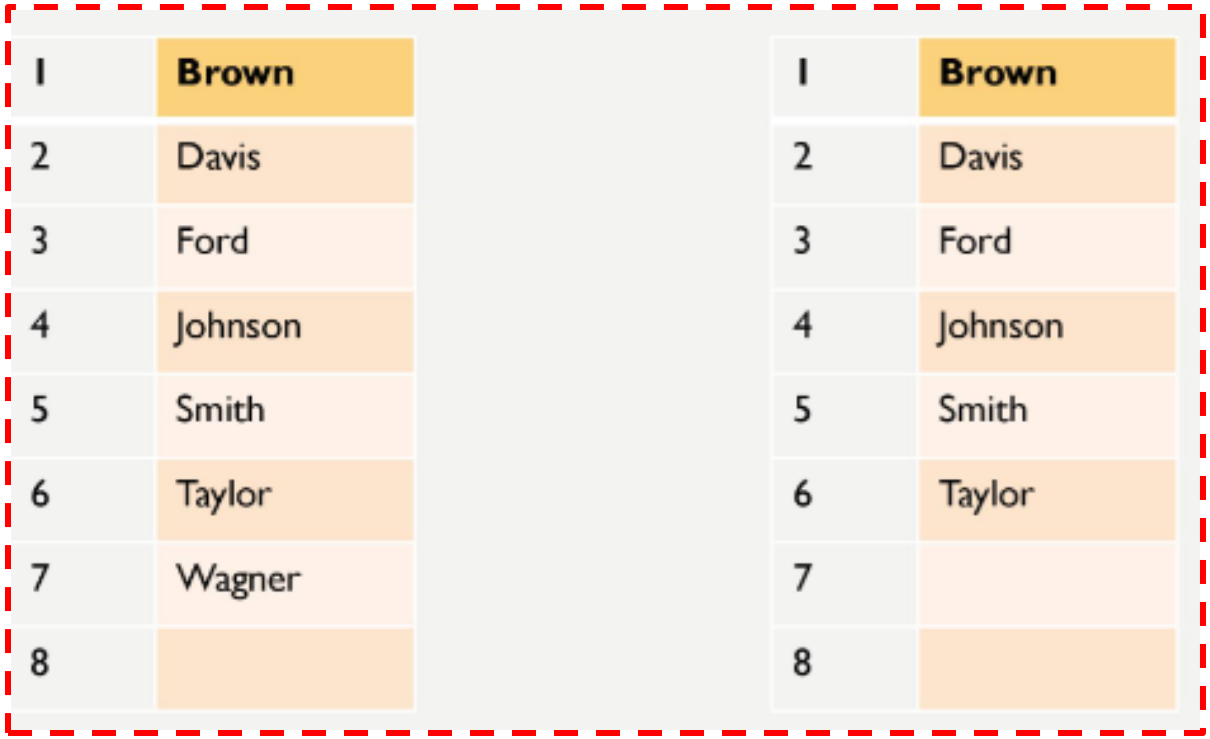
Arr_size = 6

0	1	2	3	4	5
8	17	7	2	4	3

Tree-step for insertion in an array:

Array Operations: Deletion

- Removing an element to the array
  - Beginning
  - Middle
  - End
  - Position



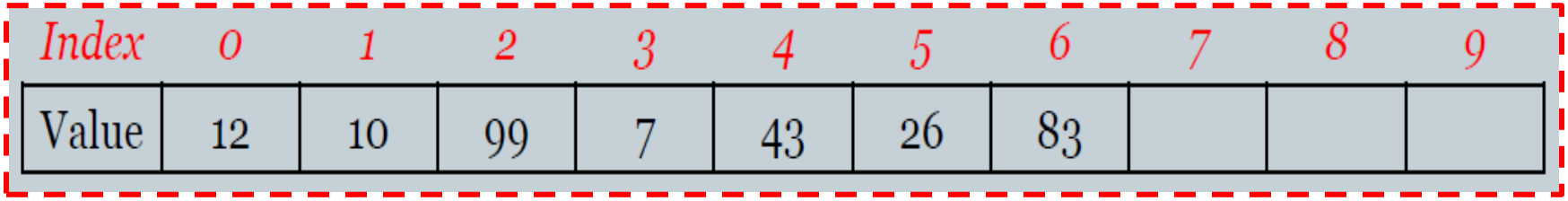
1	Brown
2	Davis
3	Ford
4	Johnson
5	Smith
6	Taylor
7	Wagner
8	

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

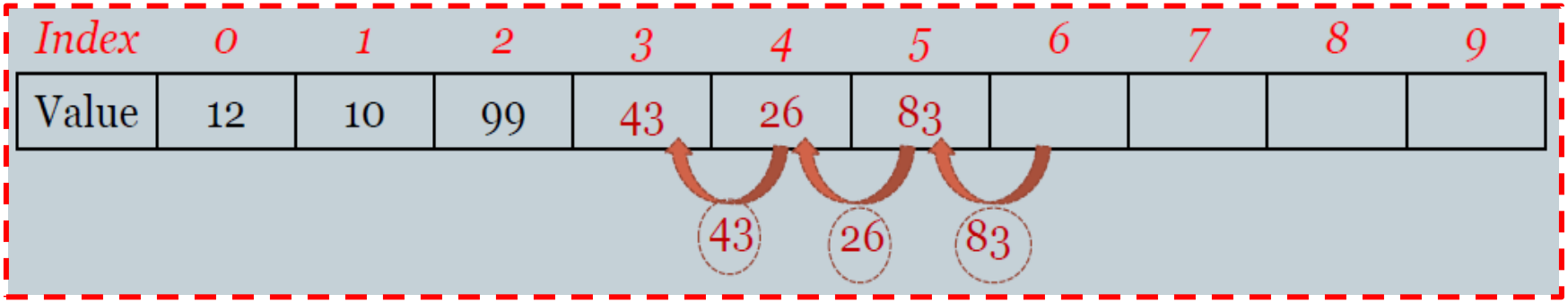
➢ Insert *Wagner* at the *End* of the array

Array Operations: Deletion

- For the following array:

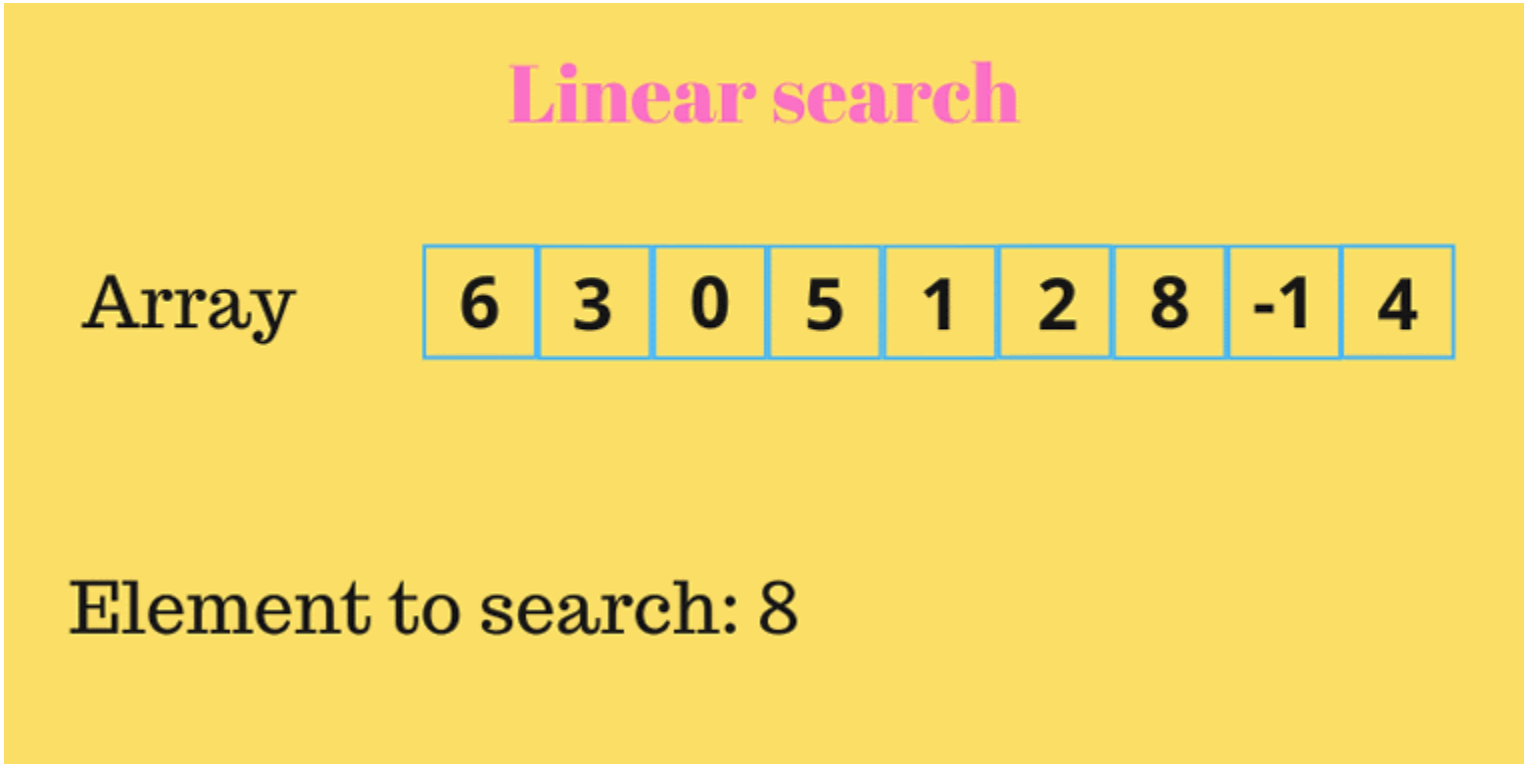


- Delete an element example number 7 from an array:



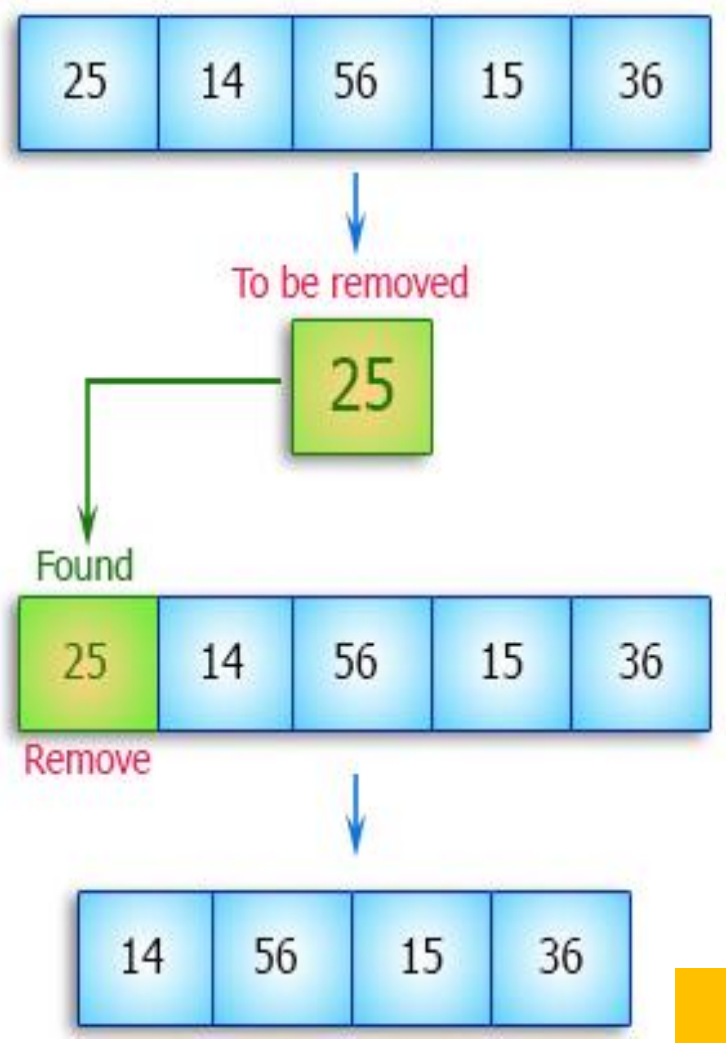
Array Operations: Searching

- Looking for elements which match given number:



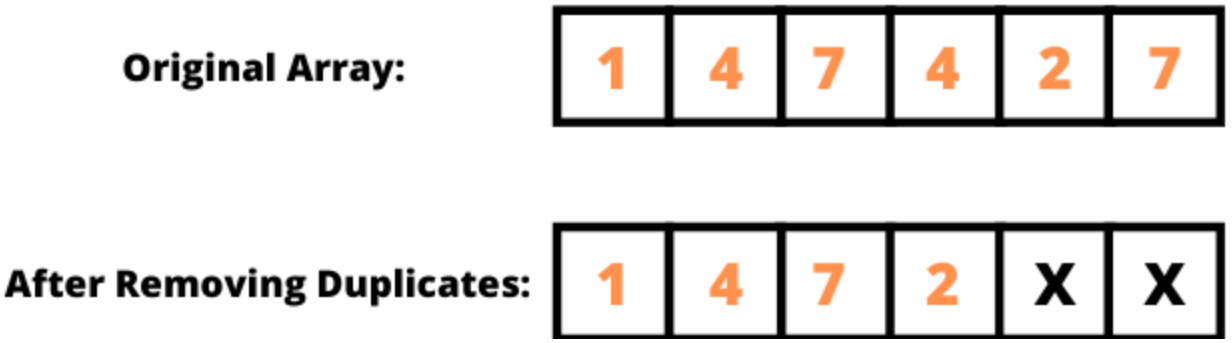
Array Operations: Searching

- **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)/..
  - **Remove** element when found from an Array



Array Operations: Searching

- **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.
  - **Remove duplicated from an Array**

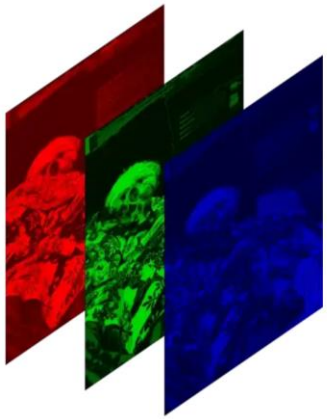


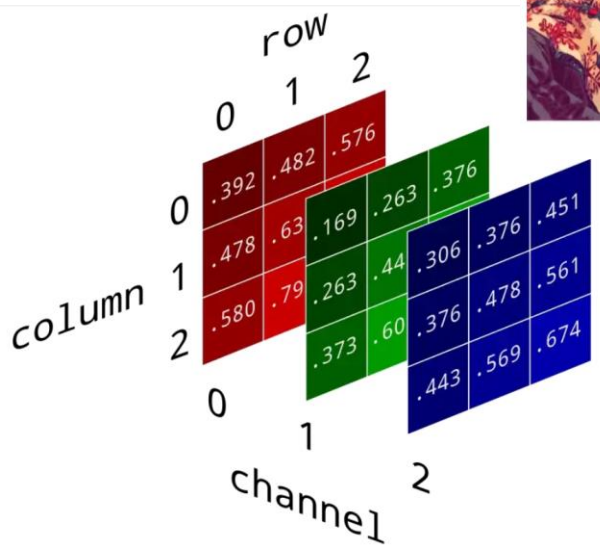
Disadvantages of Using Arrays

- 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 isn’t easy to estimate the size of the array

Applications of Arrays

- Implementation of Stacks and Queues
- Implementation of other data structures: lists, heaps, hash tables, strings, and VLists.
- CPU Scheduling
- Processing an Image





W3 – Lab

Exercise

Create a class of array to store data of any type, you want (int, double, string, char, float,…), containing a few functions:

1. **Insert** an element to the array
2. **Insert a new element**: beginning/ending/any position
3. **Display** array elements of the array
4. **Delete** an element: beginning/ending/any position
5. **Search**: Non-duplicate and Duplicate Array

Thanks!