UNIVERSITY OF MILA

Faculty of Science and Technology

Department of Process Engineering

Level: 1st year ST - ENG & LMD

Course : Introduction to Programming

Lecture 06:

Operations on One-Dimensional Arrays

---

by:

Dr. Farouk KECITA

Academic Year 2025/2026

1 Operations on One-Dimensional Arrays in C

This lecture covers various operations that can be performed on one-dimensional arrays in C, including searching, sorting, copying, modifying, deleting elements, and performing arithmetic operations and statistical analysis.

1.1 Searching an Element in a 1-D Array (Linear Search)

The linear search method, also known as sequential search, is a straightforward method for searching an array or list for a specific element. It operates by comparing each array member to the target value to determine if there is a match or to traverse the entire array iteratively.

1.1.1 Method 01: Finding a Specific Element

```c
#include <stdio.h>
int main() {
    int arr[] = {5, 10, 9, 12};
    int i, index = -1, target;

    printf("The target is: ");
    scanf("%d", &target);

    // Determine the index if the target is found
    for (i = 0; i < 4; i++) {
        if (arr[i] == target) {
            index = i;
        }
    }

    // if the target is not found, index = -1
    if (index == -1)
        printf("Element not found\n");
    // if the target is found, index != -1
    else {
        printf("Element found at index: %d\n", index);
    }
    return 0;
}
```

Listing 1: Linear Search - Method 01

```
Run 1:
The target is: 9
Element found at index: 2

Run 2:
The target is: 16
Element not found
```

1.1.2 Method 02: Finding a Specific Element (Alternative)

```c
#include <stdio.h>
int main() {
    int arr[] = {5, 10, 9, 12};
    int i, target;

    printf("The target is: ");
    scanf("%d", &target);

    // Determine the index if the target is found
    for (i = 0; i < 4; i++) {
        if (arr[i] == target) {
            printf("Element found at index: %d\n", i);
            return 0;
        }
    }

    // if the target is not found
    printf("Element not found\n");
    return 0;
}
```

Listing 2: Linear Search - Method 02

```
Run 1:
The target is: 9
Element found at index: 2

Run 2:
The target is: 16
Element not found
```

1.2 Sorting of 1-D Array

The sorting process involves arranging elements in an array in a particular order, usually ascending or descending. In computer science, sorting is a basic issue with several applications, including data analysis, data compression, and searching.

The most popular and used method is the "Selection Sort" algorithm. This method allows us to sort an array by repeatedly finding the lowest element from the unsorted part of the array and putting it at the beginning.

```c
#include <stdio.h>
int main() {
    int arr[] = {5, 10, 9, 12};
    int temp;
    int i, j;

    printf("Array elements Before Sorting\n");
    for(i = 0; i < 4; i++) {
        printf("arr[%d] = %d\n", i, arr[i]);
    }

    for(i = 0; i < 4; i++) {
        for(j = i + 1; j < 4; j++) {
            if(arr[i] > arr[j]) {
                temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
    }

    printf("\nArray elements After Sorting\n");
    for(i = 0; i < 4; i++) {
        printf("arr[%d] = %d\n", i, arr[i]);
    }
    return 0;
}
```

Listing 3: C Program to Sort a 1-D Array in Ascending Order

```
Array elements Before Sorting
arr[0] = 5
arr[1] = 10
arr[2] = 9
arr[3] = 12

Array elements After Sorting
arr[0] = 5
arr[1] = 9
arr[2] = 10
arr[3] = 12
```

1.3 Copying 1-D Array

If we have two arrays, array1 and array2, one is initialized and the other is just declared, and we need to copy array1 to array2, then we need to copy all the elements of array1 into array2 within a loop. This can be accomplished by copying index-by-index.

Important Note: The statement `array2 = array1;` is wrong and will produce an error. Arrays cannot be assigned directly; elements must be copied individually.

```c
#include <stdio.h>
#define MAX_SIZE 50

int main() {
    int arr1[MAX_SIZE], arr2[MAX_SIZE];
    int i, size;

    /* Input size of the array */
    printf("Enter the size of the arr1 : ");
    scanf("%d", &size);

    /* Input array elements */
    printf("Enter elements of arr1: \n");
    for(i = 0; i < size; i++) {
        scanf("%d", &arr1[i]);
    }

    /* Copy all elements from arr1 to arr2 */
    for(i = 0; i < size; i++) {
        arr2[i] = arr1[i];
    }

    /* Print all elements of arr1 */
    printf("\nElements of arr1 are : ");
    for(i = 0; i < size; i++) {
        printf("%d\t", arr1[i]);
    }

    /* Print all elements of arr2 */
    printf("\nElements of arr2 are : ");
    for(i = 0; i < size; i++) {
        printf("%d\t", arr2[i]);
    }

    return 0;
}
```

Listing 4: C Program to Copy Elements of a 1-D Array

```
Enter the size of the arr1 : 4
Enter elements of arr1:
15
18
9
10

Elements of arr1 are : 15   18   9   10
Elements of arr2 are : 15   18   9   10
```

1.4 Modifying Elements in 1-D Array

We can modify the value of an array element at the given index i in a similar way to accessing an element by using the subscript operator [ ] with the element index i.

Syntax: `array_name[i] = new_value;`

```c
#include <stdio.h>

int main() {
    int arr[] = {15, 18, 9, 10};

    printf("The third element of the array 'arr' is: %d\n",
       arr[2]);

    arr[2] = 20;  // update or change the third element

    printf("The third element of the array 'arr' is now: %d\n",
       arr[2]);

    return 0;
}
```

Listing 5: C Program to Modify an Element in 1-D Array

```
The third element of the array 'arr' is: 9
The third element of the array 'arr' is now: 20
```

1.5 Delete an Element from a 1-D Array

Deletion from an array is a costly operation. There are several ways to delete an element from an array. One of the approved methods is the ”Delete an Element from an Array by Index” approach.

Steps to delete a specific element from a 1-D array:

1. Input the position of the element to delete.
2. Compare the position (index) with the total number of elements (size).
3. If the position is greater than size, deletion is not possible.
4. Otherwise, copy the next element to the current element: `arr[i] = arr[i + 1]`.
5. Decrement the size of the array by one.
6. Display the resultant array.

```c
#include <stdio.h>

int main() {
    int i, size, index, arr[10];

    printf("Enter the size of the array: ");
    scanf("%d", &size);

    printf("Enter the elements of the array: \n");
    for (i = 0; i < size; i++) {
        printf("arr[%d] = ", i);
        scanf("%d", &arr[i]);
    }

    printf("Enter the index of the element to be deleted: ");
    scanf("%d", &index);

    if (index >= size) {
        printf("\nDeletion is not possible in the array.\n");
    }
    else {
        for (i = index; i < size - 1; i++) {
            arr[i] = arr[i + 1];
        }

        printf("The array after deleting the element is:\n");
        for (i = 0; i < size - 1; i++) {
            printf("arr[%d] = %d\n", i, arr[i]);
        }
    }
    return 0;
}
```

Listing 6: C Program to Delete an Element from 1-D Array

```
Run 1:
Enter the size of the array: 4
Enter the elements of the array:
arr[0] = 3
arr[1] = 5
arr[2] = 7
arr[3] = 10
Enter the index of the element to be deleted: 2
The array after deleting the element is:
arr[0] = 3
arr[1] = 5
arr[2] = 10

Run 2:
Enter the size of the array: 4
Enter the elements of the array:
arr[0] = 1
arr[1] = 2
arr[2] = 5
arr[3] = 7
Enter the index of the element to be deleted: 4
Deletion is not possible in the array.
```

1.6 Sum of 1-D Array Elements

The standard method used to calculate the sum of all the elements of an array is as follows:

1. Declare a variable to store the sum and initialize it to 0: `sum = 0`
2. Add each element of the array to the sum using a for loop: `sum = sum + a[i]`
3. Print the sum.

```c
#include <stdio.h>

int main() {
    int arr[100], size, i, sum = 0;

    // get the array size input from user
    printf("Enter array size: \n");
    scanf("%d", &size);

    // get all elements of array
    printf("Enter array elements: \n");
    for(i = 0; i < size; i++) {
        scanf("%d", &arr[i]);
    }

    // add all elements to the variable sum
```

```c
    for(i = 0; i < size; i++) {
        sum = sum + arr[i];  // same as sum += arr[i];
    }

    // print the result
    printf("Sum of the array = %d\n", sum);

    return 0;
}
```

Listing 7: C Program to Calculate the Sum of Array Elements

```
Enter array size:
4
Enter array elements:
12
8
10
20
Sum of the array = 50
```

1.7 Arithmetic Operations on 1-D Arrays

The basic mathematical calculations such as sum, average, or product can all be performed on the elements in one-dimensional arrays and act on the arrays element-by-element.

General Syntax: If `arr1` and `arr2` are two 1-D arrays:

- Addition: `result[i] = arr1[i] + arr2[i]`
- Subtraction: `result[i] = arr1[i] - arr2[i]`
- Multiplication: `result[i] = arr1[i] * arr2[i]`
- Division: `result[i] = arr1[i] / arr2[i]`

```c
#include <stdio.h>
#define SIZE 5

int main() {
    int i, arr1[SIZE], arr2[SIZE], sum[SIZE];

    printf("Enter the first array:\n");
    for(i = 0; i < SIZE; i++) {
        printf("arr1[%d] = ", i);
        scanf("%d", &arr1[i]);
    }

    printf("Enter the second array:\n");
```

```c
    for(i = 0; i < SIZE; i++) {
        printf("arr2[%d] = ", i);
        scanf("%d", &arr2[i]);
    }

    for(i = 0; i < SIZE; i++) {
        sum[i] = arr1[i] + arr2[i];
    }

    printf("Sum of two arrays is:\n");
    for(i = 0; i < SIZE; i++) {
        printf("sum[%d] = arr1[%d] + arr2[%d] = %d\n", i, i, i,
            sum[i]);
    }

    return 0;
}
```

Listing 8: C Program to Add Two Arrays

```
Enter the first array:
arr1[0] = 12
arr1[1] = 13
arr1[2] = 4
arr1[3] = 20
arr1[4] = 18

Enter the second array:
arr2[0] = 20
arr2[1] = 12
arr2[2] = 16
arr2[3] = 18
arr2[4] = 9

Sum of two arrays is:
sum[0] = arr1[0] + arr2[0] = 32
sum[1] = arr1[1] + arr2[1] = 25
sum[2] = arr1[2] + arr2[2] = 20
sum[3] = arr1[3] + arr2[3] = 38
sum[4] = arr1[4] + arr2[4] = 27
```

1.8 Statistical Analysis on Arrays

Statistical analysis on arrays involves calculating measures like mean, variance, and standard deviation to understand the distribution and characteristics of the data.

Formulas:

$$\text{Mean: } \bar{X} = \frac{\sum_{i=1}^{N} X_i}{N}$$

$$\text{Variance: } \sigma^2 = \frac{\sum_{i=1}^{N} (X_i - \bar{X})^2}{N}$$

$$\text{Standard Deviation: } \sigma = \sqrt{\text{Variance}}$$

Algorithm for Statistical Analysis:

1. Read N items (scores of students).
2. Calculate the sum and mean of the items.
3. Calculate the variance.
4. Calculate the standard deviation.

```c
#include <stdio.h>
#include <math.h>

int main() {
    int scores[100], N, i;
    float sum = 0, mean, variance = 0, std_deviation;

    // Read number of students
    printf("Enter the number of students: ");
    scanf("%d", &N);

    // Read scores
    printf("Enter the scores of %d students:\n", N);
    for(i = 0; i < N; i++) {
        printf("Score %d: ", i + 1);
        scanf("%d", &scores[i]);
        sum += scores[i];
    }

    // Calculate mean
    mean = sum / N;

    // Calculate variance
    for(i = 0; i < N; i++) {
        variance += pow((scores[i] - mean), 2);
    }
    variance = variance / N;

    // Calculate standard deviation
    std_deviation = sqrt(variance);

    // Display results
    printf("\n=== Statistical Analysis Results ===\n");
```

```c
    printf("Number of students: %d\n", N);
    printf("Sum of scores: %.2f\n", sum);
    printf("Mean (Average) score: %.2f\n", mean);
    printf("Variance: %.2f\n", variance);
    printf("Standard Deviation: %.2f\n", std_deviation);

    return 0;
}
```

Listing 9: C Program for Statistical Analysis of Student Scores

```
Enter the number of students: 5
Enter the scores of 5 students:
Score 1: 85
Score 2: 90
Score 3: 78
Score 4: 92
Score 5: 88

=== Statistical Analysis Results ===
Number of students: 5
Sum of scores: 433.00
Mean (Average) score: 86.60
Variance: 22.24
Standard Deviation: 4.72
```

Summary of Array Operations

Operation	Description	Example
Searching	Find if an element exists in array	Linear search
Sorting	Arrange elements in order	Selection sort
Copying	Duplicate array contents	Element-wise copy
Modifying	Change element value	arr[i] = new_value
Deleting	Remove an element	Shift elements left
Summation	Calculate total of all elements	sum += arr[i]
Arithmetic	Perform math operations	result[i] = a[i] + b[i]
Statistical	Calculate statistical measures	Mean, Variance, Std Dev

Table 1: Summary of One-Dimensional Array Operations

Key Takeaways:

- Arrays are fundamental data structures for storing collections of similar data
- Most array operations require iterating through elements using loops
- Understanding array operations is essential for efficient programming
- Statistical analysis on arrays helps in data interpretation and decision making