# Experiment No.01

## PART A

(PART A: TO BE REFFERED BY STUDENTS)

* 1. **Aim:** Implementation of various array operations like traversal, insertion and deletion using any real life application.

###### Prerequisite: - Knowledge of any programming language

* 1. **Outcome:**

After successful completion of this experiment, students will be able to

* + 1. Explain fundamental concepts of Array.
    2. Understand and implement traversal, insertion and deletion on 1-D array

###### Theory:

An array is a linear data structure that collects elements of the same data type and stores them in contiguous and adjacent memory locations. Arrays work on an index system starting from 0 to (n-1), where n is the size of the array.

There are majorly two types of arrays, they are:

One-Dimensional Arrays:

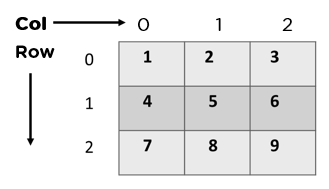


You can imagine a 1d array as a row, where elements are stored one after another.

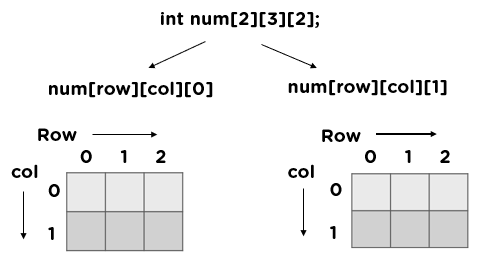
Multi-Dimensional Arrays:

These multi-dimensional arrays are again of two types. They are:

**Two-Dimensional Arrays:**



**Three-Dimensional Arrays:**



* Arrays are declared with square brackets with the size of the arrays as its argument.

Here is the syntax for arrays:

1D Arrays: int arr[n];

2D Arrays: int arr[m][n];

* Arrays are initialized as:

int arr[]= {100,200, 300, 400}; or

int arr[]= {100, 200, 300, 400};

* If we are reading the size from user, then we can declare it as int arr[n] where n is the size of the array.
  1. **Tasks to be completed**
     1. Choose a real world problem scenario
     2. Write a program in C++

1. To read data into an array from user
2. To traverse the array and print it to the user.
3. To insert element at the first, last and any specific position given by user in array
4. To delete element at the first, last and any specific position given by user in array

**PART B**

(PART B: TO BE COMPLETED BY STUDENTS)

(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case the there is no Black board access available)

|  |  |
| --- | --- |
| Roll No. C002 | Name: Kartik Sethi |
| Program : BTI | Division: B |
| Batch: B1 | Date of Experiment: |
| Date of Submission: | Grade : |

* 1. **Tasks given in PART A to be completed here**

1. #include <iostream>
2. using namespace std;
3. int main()
4. {
5. int n;
6. cout<<"Enter size of array: ";
7. cin>>n;
8. int arr[n];
9. for(int i=0;i<n;i++)
10. {
11. cout<<"Enter "<<i+1<<" Array: ";
12. cin>>arr[i];
13. }
14. cout<<endl;
15. for(int i=0;i<n;i++)
16. {
17. cout<<"["<<arr[i]<<"]"<<" ";
18. }
19. int num1,pos;
20. cout<<endl;
21. cout<<"enter the number you'll be inserting at any postion: ";
22. cin>>num1;
23. cout<<"enter the postion ";
24. cin>>pos;
25. int Arr[n+1];
26. // copying elements before the pos
27. for (int i = 0; i < pos; ++i)
28. {
29. Arr[i] = arr[i];
30. }
31. Arr[pos-1]=num1;
32. // copying elements after the pos
33. for (int i = pos; i < n; ++i)
34. {
35. Arr[i+1] = arr[i];
36. }
37. cout<<"New array: ";
38. for (int i = 0; i < n+1; ++i)
39. {
40. cout << Arr[i] << " ";
41. }
42. cout<<endl;
43. cout<<"What postion you want to delete: {index value starts from 0}";
44. cin>>pos;
45. int newArr[n];
46. // Copying elements before the position
47. for (int i = 0; i < pos; i++)
48. {
49. newArr[i] = Arr[i];
50. }
51. // Copying elements after the position
52. for (int i = pos + 1; i < n + 1; i++)
53. {
54. newArr[i - 1] = Arr[i];
55. }
56. cout << "Array after deletion: ";
57. for (int i = 0; i < n; ++i)
58. {
59. cout << newArr[i] << " ";
60. }
61. cout << endl;
62. }

***A screenshot of a computer

Description automatically generated***

* 1. **Observations and Learning:**

*(****Students must write the observations and learning based on their understanding built about the subject matter and inferences drawn)***

###### Conclusion:

*(****Students must write the conclusive statements as per the attainment of individual outcomes listed above and learning/observation noted in section B.2)***

-Was able to implement various array operations like traversal, insertion and deletion using any real life application.

###### Question of curiosity:

###### What is the difference between int array[] and int[] array?

**Ans;** In standard C++, there is no functional difference between int array[] and int[] array. Both are equivalent ways to declare an integer array.

However, there might be some subtle differences or preferences in specific coding styles or compilers, but for general C++ programming, they are interchangeable.

###### Can we declare array size as negative? Justify

Ans: An array is a data structure that stores a collection of elements of the same type. The size of an array specifies the number of elements that it can hold. Since the number of elements cannot be negative, the size of an array cannot be negative either.

###### What are the advantages and disadvantages of an array?

**Ans:**

**Advantages of Arrays:**

* **Efficient random access:** You can access any element in an array directly using its index, making it efficient for retrieving or modifying specific elements.
* **Memory efficiency:** Arrays store elements in contiguous memory locations, reducing memory overhead compared to some other data structures.
* **Simple implementation:** Arrays are relatively easy to implement and use in programming languages.
* **Cache-friendly:** Due to their contiguous memory layout, arrays often exhibit good cache locality, which can improve performance in certain scenarios.

**Disadvantages of Arrays:**

* **Fixed size:** The size of an array is determined at creation and cannot be easily changed. If you need a dynamic data structure, you might need to use other data structures like Array Lists or vectors.
* **Inefficient insertion and deletion:** Inserting or deleting elements in the middle of an array can be expensive, as it requires shifting elements to make space or fill the gap.
* **Wastage of memory:** If you don't know the exact size of the data you need to store beforehand, you might end up allocating more memory than necessary, leading to wastage.
* **Vulnerability to buffer overflows:** If you access elements outside the array's bounds, it can lead to buffer overflows, a common security vulnerability.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*