Name: Veeransh Shah

Reg No.: 221070063

EXPERIMENT No. : 1

**AIM:**

The primary objective of this experiment is to develop and implement a modular C++ program for array manipulation. The program allows users to dynamically input, display, search for elements, sort, delete, and insert elements in an integer array.

**THEORY:**

The **input** function prompts the user to enter elements into an array of a specified size, while the **output** function displays the array elements.

The **search** function enables users to input an element and checks if it exists in the array, providing the index if found.

The **sort** function utilizes the bubble sort algorithm to arrange the array elements in ascending order.

The **del** function deletes an element from the array based on the user-specified index, shifting subsequent elements to fill the gap.

Conversely, the **insert** function inserts a user-provided element at a specified index, shifting existing elements to accommodate the new entry.

The **main** function orchestrates the execution of these array manipulation functions, allowing users to interactively input, manipulate, and analyze the array's contents. This code provides a practical illustration of fundamental array operations and serves as a learning example for those exploring basic data manipulation in C++.

**ALGORITHM:**

* 1.Input Function (void input(int a[], int size))
  + Display a prompt asking the user to enter the elements of the array.
  + Use a loop to iterate through each element of the array.
  + Read each element from the user using cin.
* 2.Output Function (void output(int b[], int size))
  + Display a message indicating that the elements are going to be displayed.
  + Use a loop to iterate through each element of the array.
  + Print each element to the console.
* 3.Search Function (void search(int c[], int size))
  + Declare variables element, n, and i.
  + Display a prompt asking the user to enter the element to search.
  + Use a loop to iterate through each element of the array.
  + If the element is found, set n to 1 and break out of the loop.
  + If the element is not found, set n to 0.
  + Display the result based on the value of n.
* 4.Sort Function (void sort(int d[], int size))
  + Display a message indicating that the array will be sorted.
  + Use nested loops to iterate through each pair of elements and swap them if they are in the wrong order.
  + Display the sorted array.
* 5.Delete Function (void del(int e[], int size))
  + Declare a variable index.
  + Display a prompt asking the user to enter the index of the element to delete.
  + Use a loop to shift the elements to the left, effectively removing the element at the specified index.
  + Display the elements after deletion.
* 6.Insert Function (void insert(int f[], int size))
  + Declare variables index and ele.
  + Display a prompt asking the user to enter the element to insert and the index at which to insert.
  + Use a loop to shift elements to the right to make space for the new element.
  + Insert the new element at the specified index.
  + Display the elements after insertion.
* 7.Main Function (int main())
  + Declare a variable size.
  + Display a prompt asking the user to enter the size of the array.
  + Declare an array arr of size size.
  + Call the input function to get the array elements from the user.
  + Call the output function to display the array elements.
  + Call the search function to search for an element in the array.
  + Call the sort function to sort the array.
  + Call the del function to delete an element from the array.
  + Call the insert function to insert an element into the array.

**EXAMPLE:**

Suppose you want to manipulate an array of integers. Here's an example interaction with the program:

1. Enter the size of the array: 5
2. Enter the elements of the array

8 3 5 2 1

1. Displaying elements

8 3 5 2 1

1. Enter the element you want to search: 5

Element 5 is at index 2 of the array

1. The array sorted in ascending order is

1 2 3 5 8

1. Enter the index no. of the array you want to delete: 3

Displaying elements after deletion

1 2 3 8

1. Enter the no. you want to insert: 6

Enter the index no. where you want to insert: 2

Displaying elements after insertion

1 2 6 3 8

In this example, the user first specifies the size of the array (step 1), then inputs the elements of the array (step 2). The program displays the entered elements (step 3) and allows the user to search for a specific element (step 4).

After that, the program sorts the array in ascending order (step 5), deletes an element at a specified index (step 6), and finally inserts a new element at a specified index (step 7).

**CONCLUSION:**

Thus we have successfully implemented simple array operations in this experiment.

Screenshot:

