Data Structures - Lab

Sir. Humayun

Practical #1 - Lab Task #1

Group Members

Abdullah Imran (398)
Laiba Butt (266)
Husnain Raza (282)

Submission Date

29th September, 2025 (Monday)

Lab Task #1

Problem Statement:

You are required to develop a menu-driven C++ program that performs basic operations on an array of integers. The program should allow the user to perform insertion, deletion, updating, traversal, **searching, and sorting on the array.**

Instructions:

1. **Create an integer** array of size 10.
2. Implement the following operations:
3. Insertion: Insert an element at a given position.
4. Deletion: Delete an element from a given position.
5. Updating: Update an element at a given index with a new value.
6. Traversal: Display all elements of the array.
7. Searching: Search for an element (using linear search).

Note: Use a menu-based system where the user can choose which operation to perform. Display the updated array after each operation.

Code

```
#include <iostream>

using namespace std;

int arr[10];
int size = 0;

void displayArray() {
    cout << "\nCurrent Array: ";
    if(size == 0) {
        cout << "Array is empty";
    } else {
        cout << "[ ";
        for(int i = 0; i < size; i++) {
            cout << arr[i] << " ";
        }
        cout << "]";
    }
    cout << " (Size: " << size << "/10)" << endl;
}

void insertElement() {
    if(size >= 10) {
        cout << "\nArray is full! Cannot insert more elements." << endl;
        return;
    }

    int element, position;
    cout << "\nEnter element to insert: ";
    cin >> element;
    cout << "Enter position (0 to " << size << "): ";
    cin >> position;

    if(position < 0 || position > size) {
        cout << "Invalid position! Position should be between 0 and " << size << endl;
        return;
    }

    for(int i = size; i > position; i--) {
        arr[i] = arr[i-1];
    }

    arr[position] = element;
    size++;

    cout << "Element " << element << " inserted at position " << position << endl;
    displayArray();
}

void deleteElement() {
    if(size == 0) {
        cout << "\nArray is empty! Nothing to delete." << endl;
        return;
    }

    int position;
    cout << "\nEnter position to delete (0 to " << (size-1) << "): ";
    cin >> position;

    if(position < 0 || position >= size) {
        cout << "Invalid position! Position should be between 0 and " << (size-1) << endl;
        return;
    }
    int deletedElement = arr[position];

    for(int i = position; i < size-1; i++) {
        arr[i] = arr[i+1];
    }

    size--;

    cout << "Element " << deletedElement << " deleted from position " << position << endl;
    displayArray();
}

void updateElement() {
    if(size == 0) {
        cout << "\nArray is empty! Nothing to update." << endl;
        return;
    }

    int index, newValue;
    cout << "\nEnter index to update (0 to " << (size-1) << "): ";
    cin >> index;

    if(index < 0 || index >= size) {
        cout << "Invalid index! Index should be between 0 and " << (size-1) << endl;
        return;
    }

    cout << "Current value at index " << index << " is: " << arr[index] << endl;
    cout << "Enter new value: ";
    cin >> newValue;

    int oldValue = arr[index];
    arr[index] = newValue;

    cout << "Element at index " << index << " updated from " << oldValue << " to " << newValue << endl;
    displayArray();
}

void searchElement() {
    if(size == 0) {
        cout << "\nArray is empty! Nothing to search." << endl;
        return;
    }

    int element;
    cout << "\nEnter element to search: ";
    cin >> element;
    bool found = false;
    cout << "Search results for element " << element << ": ";

    for(int i = 0; i < size; i++) {
        if(arr[i] == element) {
            if(!found) {
                cout << "\nElement found at position: ";
                found = true;
            }
            cout << i << " ";
        }
    }

    if(!found) {
        cout << "\nElement not found in the array.";
    }
    cout << endl;
}

void displayMenu() {
    cout << "\nARRAY OPERATIONS MENU" << endl;
    cout << "1. Insert Element" << endl;
    cout << "2. Delete Element" << endl;
    cout << "3. Update Element" << endl;
    cout << "4. Display Array (Traversal)" << endl;
    cout << "5. Search Element" << endl;
    cout << "6. Exit" << endl;
    cout << "Enter your choice (1-6): ";
}

int main() {
    int choice;

    for(int i = 0; i < 10; i++) {
        arr[i] = 0;
    }

    cout << "ARRAY OPERATIONS PROGRAM" << endl;
    cout << "Array size: 10 (initially empty)" << endl;

    do {
        displayMenu();
        cin >> choice;

        switch(choice) {
            case 1:
                insertElement();
                break;
            case 2:
                deleteElement();
                break;
            case 3:
                updateElement();
                break;
            case 4:
                cout << "\nArray Traversal";
                displayArray();
                break;
            case 5:
                searchElement();
                break;
            case 6:
                cout << "\nThank you for using Array Operations Program!" << endl;
                break;

            default:
                cout << "\nInvalid choice! Please enter a number between 1-6." << endl;
        }

        if(choice != 6) {
            cout << "\nPress Enter to continue...";
            cin.ignore();
            cin.get();
        }

    } while(choice != 6);

    return 0;
}
```

Output

Add Element

```
~/University/Data Structures/practicals

./lab-task-1

ARRAY OPERATIONS PROGRAM
Array size: 10 (initially empty)

ARRAY OPERATIONS MENU
1. Insert Element
2. Delete Element
3. Update Element
4. Display Array (Traversal)
5. Search Element
6. Exit
Enter your choice (1-6): 1

Enter element to insert: 10
Enter position (0 to 0): 0
Element 10 inserted at position 0

Current Array: [ 10 ] (Size: 1/10)

Press Enter to continue...
```

Update Element

```
~/University/Data Structures/practicals

./lab-task-1

ARRAY OPERATIONS MENU
1. Insert Element
2. Delete Element
3. Update Element
4. Display Array (Traversal)
5. Search Element
6. Exit
Enter your choice (1-6): 3

Enter index to update (0 to 0): 0
Current value at index 0 is: 10
Enter new value: 50
Element at index 0 updated from 10 to 50

Current Array: [ 50 ] (Size: 1/10)

Press Enter to continue...
```

Search Element

```
~/University/Data Structures/practicals

./lab-task-1

ARRAY OPERATIONS MENU
1. Insert Element
2. Delete Element
3. Update Element
4. Display Array (Traversal)
5. Search Element
6. Exit
Enter your choice (1-6): 5

Enter element to search: 50
Search results for element 50:
Element found at position(s): 0

Press Enter to continue...
```

Delete Element

```
~/University/Data Structures/practicals

./lab-task-1
```

```
ARRAY OPERATIONS MENU
1. Insert Element
2. Delete Element
3. Update Element
4. Display Array (Traversal)
5. Search Element
6. Exit
Enter your choice (1-6): 2

Enter position to delete (0 to 1): 0
Element 50 deleted from position 0

Current Array: [ 40 ] (Size: 1/10)

Press Enter to continue...
```

Display Array (Traversal)

```
~/University/Data Structures/practicals

./lab-task-1
```

```
ARRAY OPERATIONS MENU
1. Insert Element
2. Delete Element
3. Update Element
4. Display Array (Traversal)
5. Search Element
6. Exit
Enter your choice (1-6): 4

Array Traversal
Current Array: [ 40 ] (Size: 1/10)

Press Enter to continue...
```