KANISHK BEHL

102105058

3EIC-2

DSA ASSIGN-2

WAP a program to check whether a given number is present in array or not(linear search).

Run Debug Stop Share Save { } Beautify Language C++

main.cpp

```cpp
#include <iostream>

bool linearSearch(const int arr[], int size, int target) {
    for (int i = 0; i < size; ++i) {
        if (arr[i] == target) {
            return true; // Number found in the array
        }
    }
    return false; // Number not found in the array
}

int main() {
    const int size = 5;
    int array[size] = {10, 5, 8, 3, 7};

    int targetNumber;
    std::cout << "Enter the number to search: ";
    std::cin >> targetNumber;

    if (linearSearch(array, size, targetNumber)) {
        std::cout << targetNumber << " is present in the array." << std::endl;
    } else {
        std::cout << targetNumber << " is not present in the array." << std::endl;
    }

    return 0;
}
```

input

```
Enter the number to search: 6
6 is not present in the array.


...Program finished with exit code 0
Press ENTER to exit console.
```

WAP a program to get the second maximum and second minimum elements in an array.

(20) Lecture 1 Intro... (20) Python for Begi... (20) DBMS -

Run Debug Stop Share Save {} Beautify

main.cpp

```cpp
#include <iostream>
#include <climits>

void findSecondMaxMin(const int arr[], int size, int &secondMax, int &secondMin) {
    if (size < 2) {
        std::cerr << "Array size should be at least 2." << std::endl;
        return;
    }

    int max = INT_MIN;
    int min = INT_MAX;

    // Find the maximum and minimum elements in the array
    for (int i = 0; i < size; ++i) {
        if (arr[i] > max) {
            secondMax = max;
            max = arr[i];
        } else if (arr[i] > secondMax && arr[i] != max) {
            secondMax = arr[i];
        }

        if (arr[i] < min) {
            secondMin = min;
            min = arr[i];
        } else if (arr[i] < secondMin && arr[i] != min) {
            secondMin = arr[i];
        }
    }
}

int main() {
    const int size = 7;
    int array[size] = {10, 5, 8, 3, 7, 2, 9};

    int secondMax = INT_MIN;
    int secondMin = INT_MAX;

    findSecondMaxMin(array, size, secondMax, secondMin);

    if (secondMax != INT_MIN) {
        std::cout << "Second Maximum Element: " << secondMax << std::endl;
    } else {
        std::cout << "Array does not have a second maximum element." << std::endl;
    }

    if (secondMin != INT_MAX) {
        std::cout << "Second Minimum Element: " << secondMin << std::endl;
    } else {
        std::cout << "Array does not have a second minimum element." << std::endl;
    }

    return 0;
}
```

```
Second Maximum Element: 9
Second Minimum Element: 3


...Program finished with exit code 0
Press ENTER to exit console.
```

WAP a program to perform insertion (any location), detection (any location) and transversal in an array.

main.cpp

```cpp
// Function to insert an element at a specified location in the array
void insertElement(int arr[], int &size, int position, int value) {
    if (position < 0 || position > size) {
        std::cerr << "Invalid position for insertion." << std::endl;
        return;
    }

    // Shift elements to make space for the new element
    for (int i = size - 1; i >= position; --i) {
        arr[i + 1] = arr[i];
    }

    // Insert the new element
    arr[position] = value;

    // Increase the size of the array
    ++size;
}

// Function to delete an element from a specified location in the array
void deleteElement(int arr[], int &size, int position) {
    if (position < 0 || position >= size) {
        std::cerr << "Invalid position for deletion." << std::endl;
        return;
    }

    // Shift elements to fill the gap left by the deleted element
    for (int i = position; i < size - 1; ++i) {
        arr[i] = arr[i + 1];
    }

    // Decrease the size of the array
    --size;
}

// Function to traverse and display the elements of the array
void traverseArray(const int arr[], int size) {
    std::cout << "Array Elements: ";
    for (int i = 0; i < size; ++i) {
        std::cout << arr[i] << " ";
    }
    std::cout << std::endl;
}

int main() {
    const int maxSize = 100; // Maximum size of the array
    int arr[maxSize];
    int size = 0; // Current size of the array

    // Insert elements into the array
    insertElement(arr, size, 0, 10);
    insertElement(arr, size, 1, 20);
    insertElement(arr, size, 2, 30);
    insertElement(arr, size, 3, 40);

    // Display the array after insertion
    traverseArray(arr, size);

    // Insert an element at a specified location
    insertElement(arr, size, 2, 25);

    // Display the array after insertion
    traverseArray(arr, size);

    // Delete an element from a specified location
    deleteElement(arr, size, 1);

    // Display the array after deletion
    traverseArray(arr, size);

    return 0;
}
```

```
Array Elements: 10 20 30 40
Array Elements: 10 20 25 30 40
Array Elements: 10 25 30 40


...Program finished with exit code 0
Press ENTER to exit console.
```

Write a menu drive driven program to perform the addition, multiplication and subtraction of 2 arrays.

```cpp
#include <iostream>

// Function to perform addition of two arrays
void addArrays(const int arr1[], const int arr2[], int result[], int size) {
    for (int i = 0; i < size; ++i) {
        result[i] = arr1[i] + arr2[i];
    }
}

// Function to perform subtraction of two arrays
void subtractArrays(const int arr1[], const int arr2[], int result[], int size) {
    for (int i = 0; i < size; ++i) {
        result[i] = arr1[i] - arr2[i];
    }
}

// Function to perform multiplication of two arrays
void multiplyArrays(const int arr1[], const int arr2[], int result[], int size) {
    for (int i = 0; i < size; ++i) {
        result[i] = arr1[i] * arr2[i];
    }
}

// Function to display the elements of an array
void displayArray(const int arr[], int size) {
    std::cout << "Array Elements: ";
    for (int i = 0; i < size; ++i) {
        std::cout << arr[i] << " ";
    }
    std::cout << std::endl;
}

int main() {
    const int maxSize = 5; // Maximum size of the arrays
    int array1[maxSize], array2[maxSize], result[maxSize];
    int size;

    // Input size of the arrays
    std::cout << "Enter the size of the arrays: ";
    std::cin >> size;

    if (size <= 0 || size > maxSize) {
        std::cerr << "Invalid size of the arrays." << std::endl;
        return 1;
    }

    // Input elements of the first array
    std::cout << "Enter elements of the first array:" << std::endl;
    for (int i = 0; i < size; ++i) {
        std::cout << "Element " << i + 1 << ": ";
        std::cin >> array1[i];
    }

    // Input elements of the second array
    std::cout << "Enter elements of the second array:" << std::endl;
    for (int i = 0; i < size; ++i) {
        std::cout << "Element " << i + 1 << ": ";
        std::cin >> array2[i];
    }

    int choice;
    do {
        // Display menu
        std::cout << "\nMenu:\n";
        std::cout << "1. Addition\n";
        std::cout << "2. Subtraction\n";
        std::cout << "3. Multiplication\n";
        std::cout << "4. Exit\n";
        std::cout << "Enter your choice: ";
        std::cin >> choice;

        switch (choice) {
            case 1:
                addArrays(array1, array2, result, size);
                std::cout << "Result of Addition:" << std::endl;
                displayArray(result, size);
                break;

            case 2:
                subtractArrays(array1, array2, result, size);
                std::cout << "Result of Subtraction:" << std::endl;
                displayArray(result, size);
                break;

            case 3:
                multiplyArrays(array1, array2, result, size);
                std::cout << "Result of Multiplication:" << std::endl;
                displayArray(result, size);
                break;

            case 4:
                std::cout << "Exiting the program.\n";
                break;

            default:
                std::cout << "Invalid choice. Please enter a valid option.\n";
                break;
        }

    } while (choice != 4);

    return 0;
}
```

WAP to perform sorting while merging(merge 2 sorted arrays into one sorted array).

main.cpp

```cpp
#include <iostream>

void mergeSortedArrays(const int arr1[], int size1, const int arr2[], int size2, int result[]) {
    int i = 0, j = 0, k = 0;

    while (i < size1 && j < size2) {
        if (arr1[i] <= arr2[j]) {
            result[k++] = arr1[i++];
        } else {
            result[k++] = arr2[j++];
        }
    }

    // Copy the remaining elements of arr1, if any
    while (i < size1) {
        result[k++] = arr1[i++];
    }

    // Copy the remaining elements of arr2, if any
    while (j < size2) {
        result[k++] = arr2[j++];
    }
}

int main() {
    const int maxSize = 50; // Maximum size of the arrays
    int array1[maxSize], array2[maxSize], result[maxSize * 2]; // Assuming maximum size for the merged array

    int size1, size2;

    // Input size of the first array
    std::cout << "Enter the size of the first sorted array: ";
    std::cin >> size1;

    if (size1 <= 0 || size1 > maxSize) {
        std::cerr << "Invalid size of the first array." << std::endl;
        return 1;
    }

    // Input elements of the first sorted array
    std::cout << "Enter elements of the first sorted array:" << std::endl;
    for (int i = 0; i < size1; ++i) {
        std::cout << "Element " << i + 1 << ": ";
        std::cin >> array1[i];
    }

    // Input size of the second array
    std::cout << "Enter the size of the second sorted array: ";
    std::cin >> size2;

    if (size2 <= 0 || size2 > maxSize) {
        std::cerr << "Invalid size of the second array." << std::endl;
        return 1;
    }

    // Input elements of the second sorted array
    std::cout << "Enter elements of the second sorted array:" << std::endl;
    for (int i = 0; i < size2; ++i) {
        std::cout << "Element " << i + 1 << ": ";
        std::cin >> array2[i];
    }

    // Merge the two sorted arrays
    mergeSortedArrays(array1, size1, array2, size2, result);

    // Display the merged and sorted array
    std::cout << "Merged and Sorted Array:" << std::endl;
    for (int i = 0; i < size1 + size2; ++i) {
        std::cout << result[i] << " ";
    }
```

```
Enter the size of the first sorted array: 1 2 3 4
Enter elements of the first sorted array:
Element 1: Enter the size of the second sorted array: Enter elements of the second sorted array:
Element 1: Element 2: 5 6 7 8
Element 3: Merged and Sorted Array:
2 4 5 6


...Program finished with exit code 0
Press ENTER to exit console.
```