DATA STRUCTURES AND ALGORITHMS LAB - 1 ASSIGNMENT

Student Details

- Name: Pratyush Deo Singh
- Registration Number: 25BCE5101
- Course / Branch: B. Tech CSE (CORE)
- Semester: 2nd
- Subject: Data Structures and Algorithms
- Faculty Name: Dr. Malini A

Program 1(a): Linear Search in 1D Array

Aim

To write a C program to search an element in an array using Linear Search.

Algorithm

1. Read number of elements
2. Read array elements
3. Read the element to be searched
4. Compare the element with each array element
5. Display the position if found

Program Code

linear_search.c > ...

```c
#include <stdio.h>

int main() {
    int n, i, key;
    printf("Enter number of elements: ");
    scanf("%d", &n);

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

    printf("Enter element to search: ");
    scanf("%d", &key);

    for (i = 0; i < n; i++) {
        if (arr[i] == key) {
            printf("Element found at position %d", i + 1);
            return 0;
        }
    }

    printf("Element not found");
    return 0;
}
```

Ln 27, Col 1 Spaces: 4 UTF-8 CRLF { } C Win32

Output

```
Enter number of elements: 5
Enter elements:
10
39
12
312
13
Enter element to search: 312
Element found at position 4
```

Program 1(b): Binary Search in 1D Array

Aim

To write a C program to search an element in an array using Binary Search.

Algorithm

1. Read number of elements
2. Read sorted array elements
3. Read the element to be searched
4. Apply binary search logic
5. Display the position if found

Program Code

binary_search.c

```c
#include <stdio.h>

int main() {
    int n, i, key;
    printf("Enter number of elements: ");
    scanf("%d", &n);

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

    printf("Enter element to search: ");
    scanf("%d", &key);

    int low = 0, high = n - 1, mid;

    while (low <= high) {
        mid = (low + high) / 2;

        if (arr[mid] == key) {
            printf("Element found at position %d", mid + 1);
            return 0;
        } else if (arr[mid] < key) {
            low = mid + 1;
        } else {
            high = mid - 1;
        }
    }

    printf("Element not found");
    return 0;
}
```

Ln 35, Col 1 Spaces: 4 UTF-8 CRLF { } C Win32

Output

```
Enter number of elements: 5
Enter sorted elements:
10
20
30
40
50
Enter element to search: 40
Element found at position 4
```

Program 2(a): Selection Sort

Aim

To write a C program to sort numbers in ascending order using Selection Sort.

Algorithm

1. Read number of elements
2. Compare each element with remaining elements
3. Place smallest element at correct position
4. Repeat until sorted

Program Code

selection_sort.c

```c
#include <stdio.h>

int main() {
    int n, i, j, min, temp;
    printf("Enter number of elements: ");
    scanf("%d", &n);

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

    for (i = 0; i < n - 1; i++) {
        min = i;
        for (j = i + 1; j < n; j++) {
            if (arr[j] < arr[min]) {
                min = j;
            }
        }
        temp = arr[i];
        arr[i] = arr[min];
        arr[min] = temp;
    }

    printf("Sorted array:\n");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }

    return 0;
}
```

Ln 33, Col 1 Spaces: 4 UTF-8 CRLF { } C Win32

Output

```
Enter number of elements: 5
Enter elements:
1234
12
3134
12
431
Sorted array:
12 12 431 1234 3134
```

Program 2(b): Insertion Sort

Aim

To write a C program to sort numbers in ascending order using Insertion Sort.

Algorithm

1. Read number of elements
2. Insert each element into its correct position
3. Repeat until array is sorted

Program Code

insertion_sort.c > ...

```c
#include <stdio.h>

int main() {
    int n, i, j, key;
    printf("Enter number of elements: ");
    scanf("%d", &n);

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

    for (i = 1; i < n; i++) {
        key = arr[i];
        j = i - 1;

        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = key;
    }

    printf("Sorted array:\n");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }

    return 0;
}
```

Ln 32, Col 1 Spaces: 4 UTF-8 CRLF { } C Win32

Output

```
Enter number of elements: 5
Enter elements:
2131
12
34143
1234
0
Sorted array:
0 12 1234 2131 34143
```

Program 3: Sum of Elements in 2D Array

Aim

To write a C program to find the sum of elements in a 2D array.

Algorithm

1. Read number of rows and columns
2. Read array elements
3. Add all elements
4. Display the sum

Program Code

sum_2d_array.c

```c
#include <stdio.h>

int main() {
    int r, c, i, j, sum = 0;
    printf("Enter rows and columns: ");
    scanf("%d %d", &r, &c);

    int arr[r][c];
    printf("Enter elements:\n");
    for (i = 0; i < r; i++) {
        for (j = 0; j < c; j++) {
            scanf("%d", &arr[i][j]);
            sum += arr[i][j];
        }
    }

    printf("Sum = %d", sum);
    return 0;
}
```

Ln 2, Col 1 Spaces: 4 UTF-8 CRLF { } C Win32

Output

```
Enter rows and columns: 3 3
Enter elements:
1
2
3
4
5
6
7
8
9
Sum = 45
```

Program 4: Search an Element in 2D Array

Aim

To write a C program to search an element in a 2D array and display its position.

Algorithm

1. Read rows and columns
2. Read array elements
3. Read the element to search
4. Compare element with each array element
5. Display position if found

Program Code

search_2d_array.c

```c
#include <stdio.h>

int main() {
    int r, c, i, j, key, found = 0;
    printf("Enter rows and columns: ");
    scanf("%d %d", &r, &c);

    int arr[r][c];
    printf("Enter elements:\n");
    for (i = 0; i < r; i++) {
        for (j = 0; j < c; j++) {
            scanf("%d", &arr[i][j]);
        }
    }

    printf("Enter element to search: ");
    scanf("%d", &key);

    for (i = 0; i < r; i++) {
        for (j = 0; j < c; j++) {
            if (arr[i][j] == key) {
                printf("Element found at position (%d, %d)", i + 1, j + 1);
                found = 1;
                break;
            }
        }
    }

    if (!found) {
        printf("Element not found");
    }

    return 0;
}
```

Ln 35, Col 1 Spaces: 4 UTF-8 CRLF { } C Win32

Output

```
Enter rows and columns: 2 2
Enter elements:
1
2
3
4
Enter element to search: 4
Element found at position (2, 2)
```

Result

Thus, all the given programs were successfully executed and the outputs were verified.