1 Introduction

Arrays are a fundamental data structure in the C programming language, providing a way to store multiple values of the same type under a single variable name. C supports both one-dimensional and multidimensional arrays, allowing programmers to efficiently organize and access data. One-dimensional arrays are useful for storing linear collections of data, while multidimensional arrays enable the creation of tables, matrices, or grids, making them ideal for more complex data storage. This lab focuses on understanding the implementation, usage, and manipulation of both one-dimensional and multidimensional arrays in C, with an emphasis on the practical applications of these array types in solving real-world problems.

2 C Programs and Flowcharts and Algorithms

2.1 Task: Read n Number of Values in an Array and Display It in Reverse Order

Algorithm:

1. START
2. DECLARE `n`, `arr[n]`.
3. FOR `i` = 0 to `n-1`,
    - INPUT `arr[i]`.
4. FOR `i` = `n-1` to 0,
    - DISPLAY `arr[i]`.
5. END.

Code:

```c
#include<stdio.h>
int main()

{

   int n;
   printf("Enter how many number you want to print: ");
   scanf("%d" ,&n);

   int arr[n];

   printf("Enter the elements:");
```

```
    for (int i = 0; i <n; i++)
    {
        scanf("%d" ,&arr[i]);
    }

    printf("Number in reverse order : ");

    for (int i = n-1; i >=0; i--)
    {
        printf("%d " ,arr[i]);
    }

return 0;
}
```

Input: 5
1 2 3 4 5
Output:

```
Enter how many number you want to print: 5
Enter the elements: 1 2 3 4 5
Number in reverse order : 5 4 3 2 1
```

2.2 Task: Count Duplicate Elements in an Array

Algorithm:

1. START
2. DECLARE `n`, `arr[n]`, `count = 0`.
3. FOR `i` = 0 to `n-1`,
   - INPUT `arr[i]`.
4. FOR `i` = 0 to `n-1`,
   - FOR `j` = `i+1` to `n-1`,
     - IF `arr[i] == arr[j]` AND `i != j`, THEN
       - `count = count + 1`,
       - BREAK inner loop.
5. DISPLAY `count`.
6. END.

Code:

```c
#include<stdio.h>
int main()
{
    int n;
    printf("Enter the size of an array: ");
    scanf("%d", &n);

    int arr[n];
    int count = 0;

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

    for (int i = 0; i < n; i++)
    {
        for (int j = i + 1; j < n; j++)
        {
            if (arr[i] == arr[j] && i != j)
            {
                count++;
                break;
            }
        }
    }

    printf("Total duplicate elements is: %d", count);
    return 0;
}
```

Input: 5
1 2 3 4 5
Output:

```
Enter the size of an array : 5
Enter the elements : 1 2 3 4 4
Total dublicate elements is : 1
```

2.3 Task: Find the Maximum and Minimum Elements in an Array

Algorithm:

1. START
2. DECLARE `n`, `arr[n]`, `min`, `max`.
3. FOR `i` = 0 to `n-1`,
   - READ `arr[i]`.
4. INITIALIZE `min = arr[0]`, `max = arr[0]`.
5. FOR `i` = 0 to `n-1`,
   - IF `arr[i] < min`, THEN
     - `min = arr[i]`.
   - IF `arr[i] > max`, THEN
     - `max = arr[i]`.
6. DISPLAY `max`, `min`.
7. END.

Code:

```
#include<stdio.h>

int main()
{
    int n;

    printf("Enter the size of an array: ");
    scanf("%d", &n);

    int arr[n];
    printf("Enter the elements of the array: ");

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

    int min = arr[0], max = arr[0];

    for (int i = 0; i < n; i++)
    {
```

```
        if (min > arr[i])
        {
            min = arr[i];
        }

        if (max < arr[i])
        {
            max = arr[i];
        }
    }
    printf("Maximum element is %d\n", max);
    printf("Minimum element is %d\n", min);

    return 0;
}
```

Input: 5
1 2 3 4 5
Output:

```
Enter the size of an array : 5
Enter the elements of the array : 1 2 3 4 5
Maximum element is 5
Minimum element is 1
```

2.4 Task: Separate Odd and Even Integers into Separate Arrays

Algorithm:

1. START
2. DECLARE `n`, `arr[n]`, `odd[n]`, `even[n]`, `o_count = 0`, `e_count = 0`.
3. PRINT "Enter the size of the array:"
4. INPUT `n`
5. PRINT "Enter the elements:"
6. FOR `i` = 0 to `n-1`
   - INPUT `arr[i]`
   - IF `arr[i] % 2 == 0`, THEN
     - `even[e_count++] = arr[i]`
   - ELSE

- `odd[o_count++] = arr[i]`

7. PRINT "Printing odds:"
8. FOR `i` = 0 to `o_count-1`
   - PRINT `odd[i]`
9. PRINT "Printing evens:"
10. FOR `i` = 0 to `e_count-1`
    - PRINT `even[i]`
11. END

Code:

```c
#include <stdio.h>

int main()
{
    int n;
    printf("Enter the size of the array: ");
    scanf("%d", &n);

    int arr[n];
    int odd[n];
    int even[n];
    int o_count = 0, e_count = 0;

    printf("Enter the elements: \n");

    for (int i = 0; i < n; i++)
    {
        scanf("%d", &arr[i]);
        if (arr[i] % 2 == 0)
        {
            even[e_count++] = arr[i];
        }
        else
        {
            odd[o_count++] = arr[i];
        }
    }

    printf("Printing odds:\n");
    for (int i = 0; i < o_count; i++)
    {
```

```
32          printf("%d ", odd[i]);
33      }
34
35      printf("\nPrinting evens:\n");
36      for (int i = 0; i < e_count; i++)
37      {
38          printf("%d ", even[i]);
39      }
40      printf("\n");
41
42      return 0;
43  }
```

Input: 5
1 2 3 4 5
Output:

```
Enter the size of the array: 5
Enter the elements:
1 2 3 4 5
Printing odds:
1 3 5
Printing Evens:
2 4
```

2.5 Task: Count Frequency of Each Element in an Array

Algorithm:

1. START
2. DECLARE `n`, `arr[n]`, `freq[10] = 0`
3. PRINT "Enter the size of the array:"
4. INPUT `n`
5. PRINT "Enter the elements:"
6. FOR `i` = 0 to `n-1`
   - INPUT `arr[i]`
7. FOR `i` = 0 to `n-1`
   - `freq[arr[i]]++`
8. FOR `i` = 0 to 9

- IF `freq[i] != 0`, THEN
  - PRINT `i` is appeared `freq[i]` times.

9. END

Code:

```c
#include <stdio.h>

int main()
{
    int n;
    printf("Enter the size of a array: ");
    scanf("%d", &n);

    int arr[n], freq[10] = {0};  // Array for frequencies

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

    // Count the frequency of each element
    for (int i = 0; i < n; i++)
    {
        freq[arr[i]]++;
    }

    // Print the frequency of each element
    for (int i = 0; i < 10; i++)
    {
        if (freq[i] != 0)
        {
            printf("%d is appeared %d times.\n", i, freq[i]);
        }
    }

    return 0;
}
```

Input: 5
1 2 2 2 5
Output:

```
Enter the size of a array: 5
Enter the elements : 1 2  2 2 5
1 is appeared 1 times.
2 is appeared 3 times.
5 is appeared 1 times.
```

3 Discussion

In this lab report, we worked on solving problems using one-dimensional and two-dimensional arrays. One of the challenges I faced was finding duplicate elements. Initially, I forgot to start checking from the second element, which caused the elements to be counted incorrectly. Another challenge was separating numbers into two arrays based on even and odd values. Running two loops simultaneously to separate them was a bit difficult at first, but I managed to implement it correctly after some trial and error. Additionally, I made a small mistake in the frequency-finding problem by mistakenly setting the size of the freq[] array to 5. As a result, the program only displayed frequencies for values up to 5. It took me some time to identify and fix this issue, but resolving it helped me better understand array indexing and memory allocation.

Overall, this lab improved my problem-solving skills and deepened my understanding of array operations.