**Tree Traversal using Array Implementation**

*Source Code:*

#include <stdio.h>

// Define the maximum size of the array

#define MAX\_SIZE 100

// Function to perform in-order traversal

void inOrderTraversal(int arr[], int index, int size) {

    if (index < size) {

        // Traverse the left subtree (left child)

        inOrderTraversal(arr, 2 \* index + 1, size);

        // Visit the current node

        printf("%d ", arr[index]);

        // Traverse the right subtree (right child)

        inOrderTraversal(arr, 2 \* index + 2, size);

    }

}

// Function to perform pre-order traversal

void preOrderTraversal(int arr[], int index, int size) {

    if (index < size) {

        // Visit the current node

        printf("%d ", arr[index]);

        // Traverse the left subtree (left child)

        preOrderTraversal(arr, 2 \* index + 1, size);

        // Traverse the right subtree (right child)

        preOrderTraversal(arr, 2 \* index + 2, size);

    }

}

// Function to perform post-order traversal

void postOrderTraversal(int arr[], int index, int size) {

    if (index < size) {

        // Traverse the left subtree (left child)

        postOrderTraversal(arr, 2 \* index + 1, size);

        // Traverse the right subtree (right child)

        postOrderTraversal(arr, 2 \* index + 2, size);

        // Visit the current node

        printf("%d ", arr[index]);

    }

}

int main() {

    int treeArray[MAX\_SIZE];

    int size;

    int choice;

    // Input the number of elements and the array elements

    printf("Enter the number of elements: ");

    scanf("%d", &size);

    printf("Enter the array elements: ");

    for (int i = 0; i < size; i++) {

        scanf("%d", &treeArray[i]);

    }

    printf("\nMenu:\n");

    printf("1. In-Order Traversal\n");

    printf("2. Pre-Order Traversal\n");

    printf("3. Post-Order Traversal\n");

    printf("4. Exit\n");

    do {

        printf("Enter your choice: ");

        scanf("%d", &choice);

        switch (choice) {

            case 1:

                printf("In-Order Traversal: ");

                inOrderTraversal(treeArray, 0, size);

                printf("\n");

                break;

            case 2:

                printf("Pre-Order Traversal: ");

                preOrderTraversal(treeArray, 0, size);

                printf("\n");

                break;

            case 3:

                printf("Post-Order Traversal: ");

                postOrderTraversal(treeArray, 0, size);

                printf("\n");

                break;

            case 4:

                printf("Exiting the program.\n");

                break;

            default:

                printf("Invalid choice. Please enter a valid option.\n");

        }

    } while (choice != 4);

    return 0;

}

*Output:*Enter the number of elements: 7

Enter the array elements: 1 2 3 4 5 6 7

Menu:

1. In-Order Traversal

2. Pre-Order Traversal

3. Post-Order Traversal

4. Exit

Enter your choice: 1

In-Order Traversal: 4 2 5 1 6 3 7

Enter your choice: 2

Pre-Order Traversal: 1 2 4 5 3 6 7

Enter your choice: 3

Post-Order Traversal: 4 5 2 6 7 3 1

Enter your choice: 4

Exiting the program.

A screenshot of a computer

Description automatically generated

**ABISHEK S R  
RA22110030112972**