****

**Green University of Bangladesh**

**Department of Computer Science and Engineering (CSE)**

**Faculty of Sciences and Engineering**

**Semester: (Fall, Year: 2024), B.Sc. in CSE (Day)**

**Lab Report NO #02**

**Course Title: Data Structure Lab**

**Course Code: CSE 206**

**Section: D8**

**Lab Experiment Name:** Basic operations of one-dimensional and two-dimensional array

**Student Details**

|  |  |  |
| --- | --- | --- |
| **Name** | | **ID** |
| **1.** | Rukonuzzaman Topu | 232002280 |

**Lab Date : 11/09/24**

**Submission Date : 18/09/24**

**Course Teacher’s Name : Md. Parvez Hossain**

|  |
| --- |
| **Lab Report Status**  **Marks: ………………………………… Signature: .....................**  **Comments: .............................................. Date: ..............................** |

**1. INTRODUCTION**

The purpose of this lab reports is to know the concepts of array searching in the C program. Here, we will see how we can search an element from the array with the help of linear search and binary search. In this lab report our aim is to solve some real-world problems efficiently.

**2. OBJECTIVES**

The primary objectives of this lab report are as follows:

* We will know about linear search.
* We will know about binary search.
* We will see how use recursion method.

**3. IMPLEMENTATION**

Task 1: Implement linear search algorithms using recursion.

Solution:

#include <stdio.h>

int linearSearch(int ar[], int v, int i, int size) {

    if (i >= size) {

        return -1;

    }

    else if (ar[i] == v) {

        return i;

    }

    return linearSearch(ar, v, i + 1, size);

}

int main() {

    int ar[] = {4, 12, 3, 67, 4, 9, 32, 1};

    int v;

    int size = sizeof(ar) / sizeof(ar[0]);

    printf("\nEnter the character to search: ");

    scanf(" %d", &v);

    int i=0;

    int result = linearSearch(ar, v, i, size);

    if (result == -1) {

        printf("Value '%d' not found in the ar.\n", v);

    } else {

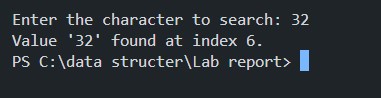
        printf("Value '%d' found at index %d.\n", v, result);

    }

    return 0;

}

Output:



Task 2: Implement Linear Search for an array with character data using recursive method.

Solution:

#include <stdio.h>

int linearSearch(char ar[], char v, int i, int size) {

    if (i >= size) {

        return -1;

    }

    else if (ar[i] == v) {

        return i;

    }

    return linearSearch(ar, v, i + 1, size);

}

int main() {

    char ar[] = {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'};

    char v;

    int size = sizeof(ar) / sizeof(ar[0]);

    printf("\nEnter the character to search: ");

    scanf(" %c", &v);

    int i=0;

    int result = linearSearch(ar, v, i, size);

    if (result == -1) {

        printf("Character '%c' not found in the ar.\n", v);

    } else {

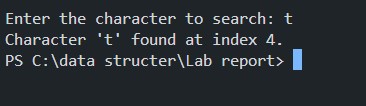
        printf("Character '%c' found at index %d.\n", v, result);

    }

    return 0;

}

Output:

****

Task 3: Implement Binary Search for an array with character data using recursive method.

Solution:

#include <stdio.h>

int binarySearch(char ar[], char v, int left, int right) {

    if (left > right) {

        return -1;

    }

    int middle = left + (right - left) / 2;

    if (ar[middle] == v) {

        return middle;

    }

    else if (ar[middle] > v) {

        return binarySearch(ar, v, left, middle - 1);

    }

    else{

    return binarySearch(ar, v, middle + 1, right);

    }

}

int main() {

    char ar[] = {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'};

    char v;

    int size = sizeof(ar) / sizeof(ar[0]);

    printf("\nEnter the character to search: ");

    scanf("%c", &v);

    int i=0;

    int result = binarySearch(ar, v, i, size - 1);

    if (result == -1) {

        printf("Character '%c' not found in the ar.\n", v);

    } else {

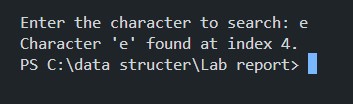
        printf("Character '%c' found at index %d.\n", v, result);

    }

    return 0;

}

Output:

****

**4. DISCUSSION**

Through this lab report, we have explored the application of recursion to array element searching. We've addressed three distinct scenarios: linear searching of numbers and strings, and binary searching of elements. For the linear search problems, we employed a recursive approach, iterating through the array or string until the target element is found or the end is reached. In contrast, binary search, which is more efficient for sorted arrays or strings, leverages a divide-and-conquer strategy. By repeatedly halving the search space, binary search significantly reduces the time required to locate an element.