DSA Sorting & Searching Practicals in lab

**AIM:-Sequential Search:-**

#include<stdio.h>

int sequentialsearch(int arr[], int size, int valuetofind){

for (int i = 0; i < size; i++) {

if (arr[i] == valuetofind) {

return i;

}

}

return -1;

}

int main(){

int size;

int valuetofind;

printf("Enter the number of elements you want to insert");

scanf("%d",&size);

int arr[size];

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

for (int i = 0; i < size; i++) {

scanf("%d", &arr[i]);

}

printf("Enter the element to search for: ");

scanf("%d", &valuetofind);

int result=sequentialsearch(arr,size,valuetofind);

if (result != -1) {

printf("Element found at index %d.\n", result);

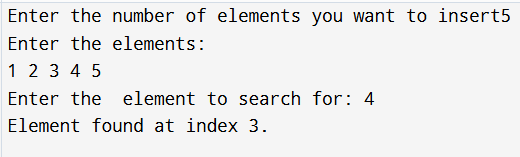
} else {

printf("Element not found in the array.\n");

}

}

**Output:-**



**AIM:-Binary Search, Bubble Sort, Selection Sort, Insertion Sort.**

#include <stdio.h>

int main() {

int arraySize;

printf("Enter size: ");

if (scanf("%d", &arraySize) != 1 || arraySize <= 0) {

printf("Invalid input. Array size must be a positive integer.\n");

return 1;

}

int array[arraySize];

printf("Enter %d elements: ", arraySize);

for (int i = 0; i < arraySize; i++) {

if (scanf("%d", &array[i]) != 1) {

printf("Error reading input.\n");

return 1;

}

}

int sortChoice;

printf("\nChoose a sorting method:\n");

printf("1. Bubble Sort\n");

printf("2. Insertion Sort\n");

printf("3. Selection Sort\n");

scanf("%d", &sortChoice);

switch (sortChoice) {

case 1:

bubbleSort(array, arraySize);

printf("Array after Bubble Sort: ");

printArray(array, arraySize);

break;

case 2:

insertionSort(array, arraySize);

printf("Array after Insertion Sort: ");

printArray(array, arraySize);

break;

case 3:

selectionSort(array, arraySize);

printf("Array after Selection Sort: ");

printArray(array, arraySize);

break;

default:

printf("Invalid choice. Exiting program.\n");

return 1;

}

int choice;

while (1) {

printf("\nEnter your choice:\n");

printf("1. Search for an element\n");

printf("2. Print the array\n");

printf("3. Exit\n");

scanf("%d", &choice);

switch (choice) {

case 1: {

int key;

printf("Enter key to search: ");

if (scanf("%d", &key) != 1) {

printf("Error reading input.\n");

return 1;

}

int index = binarySearch(array, 0, arraySize - 1, key);

if (index == -1) {

printf("Key not found in the array.\n");

} else {

printf("Key found at index %d.\n", index);

}

break;

}

case 2:

printf("Array: ");

printArray(array, arraySize);

break;

case 3:

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

return 0;

default:

printf("Invalid choice. Please try again.\n");

break;

}

}

return 0;

}

// Bubble Sort

void bubbleSort(int array[], int size) {

for (int pass = 0; pass < size - 1; pass++) {

int swapped = 0;

for (int i = 0; i < size - 1 - pass; i++) {

if (array[i] > array[i + 1]) {

int temp = array[i];

array[i] = array[i + 1];

array[i + 1] = temp;

swapped = 1;

}

}

if (!swapped) {

break;

}

}

}

// Insertion Sort

void insertionSort(int array[], int size) {

for (int i = 1; i < size; i++) {

int temp = array[i];

int j = i - 1;

while (j >= 0 && array[j] > temp) {

array[j + 1] = array[j];

j--;

}

array[j + 1] = temp;

}

}

// Selection Sort

void selectionSort(int array[], int size) {

for (int i = 0; i < size - 1; i++) {

int minIndex = i;

for (int j = i + 1; j < size; j++) {

if (array[j] < array[minIndex]) {

minIndex = j;

}

}

if (minIndex != i) {

int temp = array[i];

array[i] = array[minIndex];

array[minIndex] = temp;

}

}

}

// Binary Search

int binarySearch(int array[], int start, int end, int target) {

if (start > end) {

return -1; // not found

}

int middle = start + (end - start) / 2;

if (array[middle] == target) {

return middle;

}

if (array[middle] < target) {

return binarySearch(array, middle + 1, end, target);

} else {

return binarySearch(array, start, middle - 1, target);

}

}

// Print Array

void printArray(int array[], int size) {

for (int i = 0; i < size; i++) {

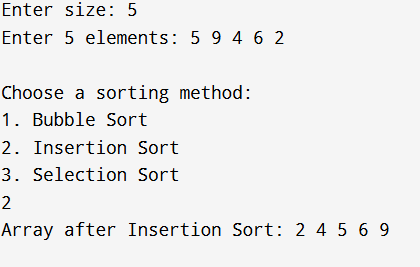
printf("%d ", array[i]);

}

printf("\n");

}

**Output:-**

****