DSA Micro Project

(4330704)

**Title: - Types of Search in DSA**

**Project by:-**

**Enrollment no.: -236250307015**

**Name: - Chandariya Dhruvil A**

**Enrollment no.: -236250307054**

**Name: - Kanzariya Raj N**

**Enrollment no.: -236250307108**

**Name: - Vadgama Shyam N**

**-------------------------------------------------------------**

**SOURCE CODE: -**

**#include <stdio.h>**

**#include <stdlib.h>**

**struct Node {**

**int data;**

**struct Node\* next;**

**};**

**int arr[5];**

**// Function for linear search**

**void linearSearch() {**

**int i, m;**

**printf("\n================= Linear Search =================\n");**

**printf("Enter 5 array elements:\n");**

**for (i = 0; i < 5; i++) {**

**printf("Element %d: ", i+1);**

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

**}**

**printf("\nEnter search key: ");**

**scanf("%d", &m);**

**for (i = 0; i < 5; i++) {**

**if (arr[i] == m) {**

**printf("\nElement found successfully at position a[%d]=%d\n", i, m);**

**return;**

**}**

**}**

**printf("\nElement not found.\n");**

**}**

**// Function for binary search**

**void binarySearch() {**

**int key, i;**

**int high = 4, low = 0, mid;**

**printf("\n================= Binary Search =================\n");**

**printf("Enter 5 organized array elements:\n");**

**for (i = 0; i < 5; i++) {**

**printf("Element %d: ", i+1);**

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

**}**

**printf("\nEnter search key: ");**

**scanf("%d", &key);**

**while (low <= high) {**

**mid = (high + low) / 2;**

**if (key == arr[mid]) {**

**printf("\nKey found at position a[%d]\n", mid);**

**return;**

**} else if (key > arr[mid]) {**

**low = mid + 1;**

**} else {**

**high = mid - 1;**

**}**

**}**

**printf("\nKey not found.\n");**

**}**

**// Function for pointer-based array search**

**void pointerSearch() {**

**int n=5, i, search, \*ptr, found = 0;**

**printf("\n============= Pointer-based Array Search =============\n");**

**printf("Enter %d elements of the array:\n", n);**

**for (i = 0; i < n; i++) {**

**printf("Element %d: ", i+1);**

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

**}**

**printf("\nEnter the element to search: ");**

**scanf("%d", &search);**

**ptr = arr;**

**for (i = 0; i < n; i++) {**

**if (\*(ptr + i) == search) {**

**printf("\nElement %d found at position %d.\n", search, i + 1);**

**found = 1;**

**break;**

**}**

**}**

**if (!found) {**

**printf("\nElement %d not found in the array.\n", search);**

**}**

**}**

**// Function for linked list search**

**void linkedListSearch() {**

**struct Node\* head = NULL;**

**int n, data, key;**

**printf("\n=============== Linked List Search ===============\n");**

**printf("Enter the total number of elements to insert: ");**

**scanf("%d", &n);**

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

**struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));**

**printf("Enter data for node %d: ", i+1);**

**scanf("%d", &data);**

**newNode->data = data;**

**newNode->next = NULL;**

**if (head == NULL) {**

**head = newNode;**

**} else {**

**struct Node\* temp = head;**

**while (temp->next != NULL) {**

**temp = temp->next;**

**}**

**temp->next = newNode;**

**}**

**}**

**printf("\nEnter data to search: ");**

**scanf("%d", &key);**

**struct Node\* temp = head;**

**int index = 0;**

**while (temp != NULL) {**

**if (temp->data == key) {**

**printf("\nElement found at index %d in the linked list.\n", index);**

**return;**

**}**

**temp = temp->next;**

**index++;**

**}**

**printf("\nElement not found in the linked list.\n");**

**}**

**int main() {**

**int choice;**

**while (1) {**

**// Display the menu with enhanced formatting**

**printf("\n==================== Search Menu ====================\n");**

**printf("|1. Linear Search|\n");**

**printf("|2. Binary Search|\n");**

**printf("|3. Pointer-based Array Search|\n");**

**printf("|4. Linked List Search|\n");**

**printf("|5. Exit|\n");**

**printf("======================================================\n");**

**printf("|Enter your choice: ");**

**scanf("%d", &choice);**

**// Switch case to handle user choice**

**switch (choice) {**

**case 1:**

**linearSearch();**

**break;**

**case 2:**

**binarySearch();**

**break;**

**case 3:**

**pointerSearch();**

**break;**

**case 4:**

**linkedListSearch();**

**break;**

**case 5:**

**printf("\nExiting the program. Goodbye!\n");**

**exit(0); // Exit the program**

**default:**

**printf("\nInvalid choice! Please select a valid option.\n");**

**break;**

**}**

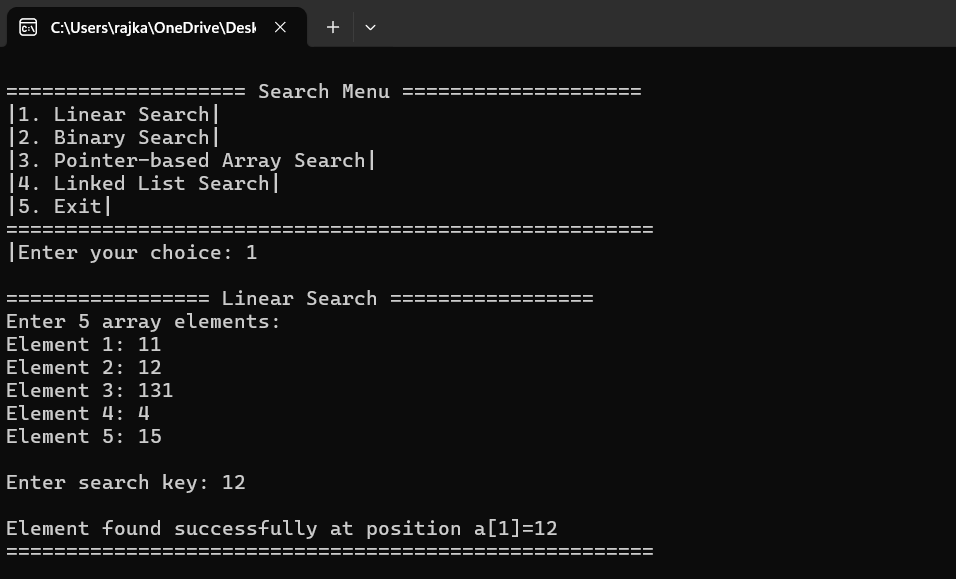
**printf("======================================================\n");**

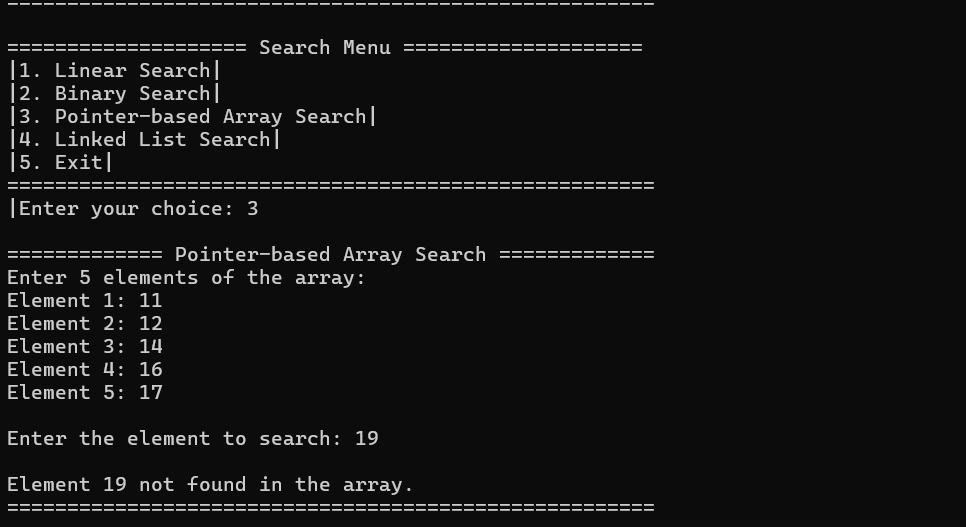
**}**

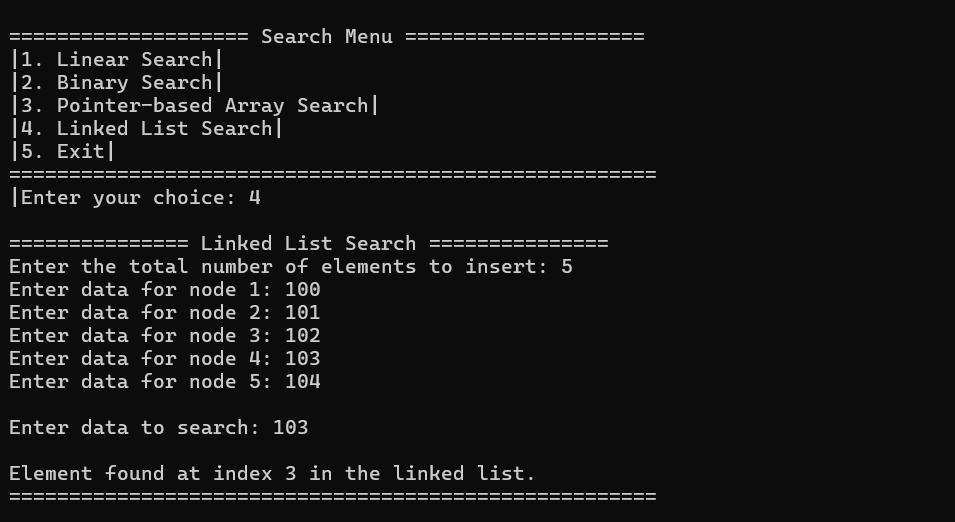
**return 0;**

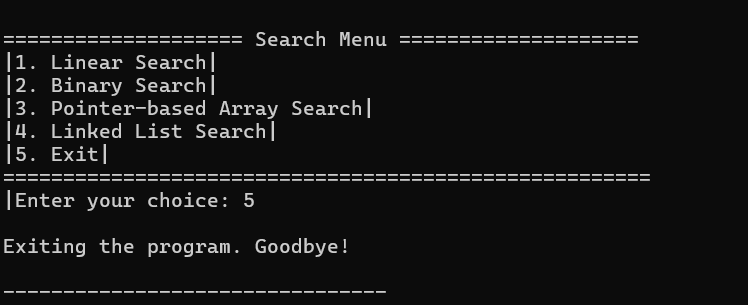
**}**

**OUTPUT**

****

****

****

****

**-------------------------------------------------------------**

**Description:-**

**This C program provides a menu-driven interface for searching elements in different data structures using various search algorithms. It allows the user to select from four different search techniques: Linear Search, Binary Search, Pointer-based Array Search, and Linked List Search. Here's a breakdown of each feature:**

**1. Linear Search**

**The user inputs 5 elements into an array.**

**A liear search is performed to find a specific element by checking each array element sequentially**

**If the element is found, its position is displayed; otherwise, a message indicates that the element was not found.**

**2. Binary Search**

**The user inputs 5 elements in a sorted array**

**Binary search is applied to find the search key by repeatedly dividing the search interval in half.**

**If the key is found, its position is shown; otherwise, a message is displayed if not found**

**3. Pointer-based Array Search:**

**Similar to linear search, but uses pointer arithmetic to traverse the array.**

**The user inputs 5 elements, and the program searches for a given element using a pointer to iterate through the array.**

**4. Linked List Search:**

**The user creates a linked list by entering a series of elements.**

**The program then searches the linked list for a specific value and indicates whether the value was found and at which index.**

**5. Exit:**

**The program offers an option to exit, which terminates the loop and ends the program.**

**The program continuously runs in a loop until the user selects the exit option. Each search method is independent and provides a clear result on whether the searched element was found.**

**The main loop in the program is responsible for continuously presenting the user with a menu of search options and executing the corresponding search function based on the user’s choice. Here's a step-by-step breakdown:**

**1. Menu Display:**

**Inside the loop, the program displays a menu with five options:**

**Linear Search**

**Binary Search**

**Pointer-based Array Search**

**Linked List Search**

**Exit**

**The menu repeats after each operation until the user chooses to exit.**

**2. User Input:**

**The user is prompted to enter their choice by typing the corresponding number (1–5). This input is stored in the variable choice.**

**3. Switch Statement:**

**The switch statement evaluates the user's choice and calls the appropriate function:**

**Case 1 calls linearSearch().**

**Case 2 calls binarySearch().**

**Case 3 calls pointerSearch().**

**Case 4 calls linkedListSearch().**

**Case 5 terminates the program by calling exit(0).**

**If the input doesn't match any valid option (1–5), the default case is triggered, which displays an error message indicating an invalid choice.**

**4. Loop Continuation:**

**After the selected search operation is completed (or if an invalid choice is made), the loop continues, displaying the menu again and waiting for the next user input.**

**The loop only breaks and ends when the user selects option 5 to exit the program.**

**This infinite loop ensures the program remains interactive, allowing the user to perform multiple searches without restarting the program.**

***Thank you***