Neh Samir Joshi

19BEC1112

25/02/2022

**CSE2003**

**Data Structures and Algorithms**

**[LAB]**

**LAB – 6**

**Searching and Sorting Algorithms**

**Aim:** To learn about and implement searching and sorting algorithms in C.

**Software Required:** Code editor (e.g. VS Code, Dev C++), GCC/G++ compiler

**Task 1: Implement binary search using C.**

**Code:**

#include <stdio.h>

int iterativeBinarySearch(int array[], int start\_index, int end\_index, int element){

   while (start\_index <= end\_index){

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

      if (array[middle] == element)

         return middle;

      if (array[middle] < element)

         start\_index = middle + 1;

      else

         end\_index = middle - 1;

   }

   return -1;

}

int main(void){

   printf("Enter size of array\n");

    int n;

    scanf("%d", &n);

    int array[n];

    printf("Enter %d integers\n", n);

    for(int i = 0; i < n; i++){

        scanf("%d", &array[i]);

    }

   printf("Enter key to search\n");

    int element;

    scanf("%d", &element);

   int found\_index = iterativeBinarySearch(array, 0, n-1, element);

   if(found\_index == -1 ) {

      printf("Element not found in the array ");

   }

   else {

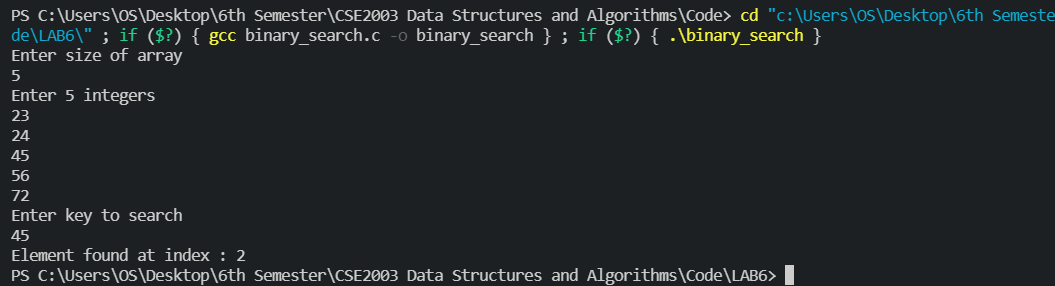
      printf("Element found at index : %d",found\_index);

   }

   return 0;

}

**Output:**



**Task 2:** Implementing Insertion sort using C.

#include <stdio.h>

void Insertion\_Sort(int arr[], int n)

{

    int temp;

    for (int i = 1; i <= n - 1; i++)

    {

        int j = i;

        while (j > 0 && arr[j - 1] > arr[j])

        {

            temp = arr[j];

            arr[j] = arr[j - 1];

            arr[j - 1] = temp;

            j--;

        }

    }

    printf("Sorted list using Insertion Sort:\n");

    for (int i = 0; i <= n - 1; i++)

    {

        printf("%d\n", arr[i]);

    }

}

int main()

{

    int n, i, j, temp;

    printf("Enter number of elements\n");

    scanf("%d", &n);

    int arr[n];

    printf("Enter %d integers\n", n);

    for (i = 0; i < n; i++)

    {

        scanf("%d", &arr[i]);

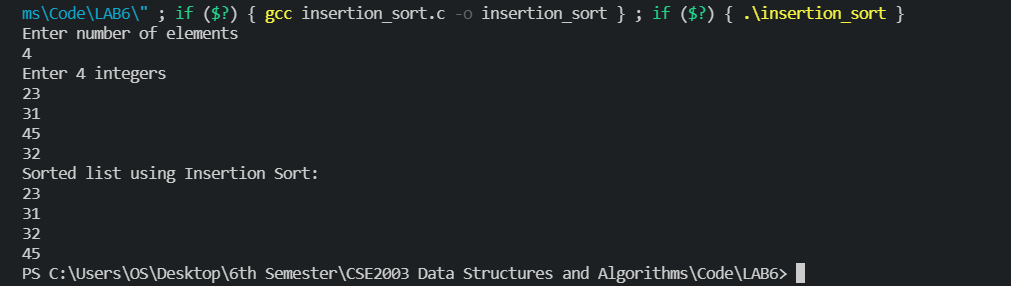
    }

    Insertion\_Sort(arr, n);

    return 0;

}

**Output:**



**Task 3: Implementing quick sort using C.**

#include <stdio.h>

void quicksort(int array[25], int low, int high)

{

    int x, y, p, temp;

    if (low < high)

    {

        p = low;

        x = low;

        y = high;

        while (x < y)

        {

            while (array[x] <= array[p] && x < high)

                x++;

            while (array[y] > array[p])

                y--;

            if (x < y)

            {

                temp = array[x];

                array[x] = array[y];

                array[y] = temp;

            }

        }

        temp = array[p];

        array[p] = array[y];

        array[y] = temp;

        quicksort(array, low, y - 1);

        quicksort(array, y + 1, high);

    }

}

int main()

{

    int x, count, array[25];

    printf("How many elements should the array have? (Max. - 25): ");

    scanf("%d", &count);

    printf("Enter %d elements for the array: ", count);

    for (x = 0; x < count; x++)

        scanf("%d", &array[x]);

    quicksort(array, 0, count - 1);

    printf("After implementing quicksort the sorted order is: ");

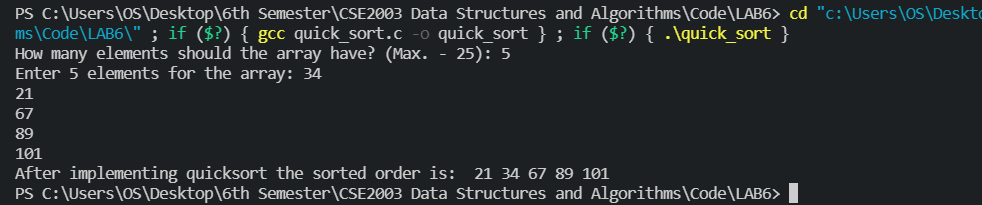
    for (x = 0; x < count; x++)

        printf(" %d", array[x]);

    return 0;

}

**Output:**



**Task 4: Implementing linear search using C.**

**Code:**

#include <stdio.h>

int linear\_search(int arr[], int size, int key){

    for(int i = 0; i < size; i++){

        if(arr[i] == key){

            return i;

        }

    }

    return -1;

}

int main(){

    printf("Enter size of array\n");

    int size;

    scanf("%d", &size);

    int arr[size];

    printf("Enter %d integers\n", size);

    for(int i = 0; i < size; i++){

        scanf("%d", &arr[i]);

    }

    printf("Enter key to search\n");

    int key;

    scanf("%d", &key);

    int index = linear\_search(arr, size, key);

    if(index == -1){

        printf("Key not found");

    }

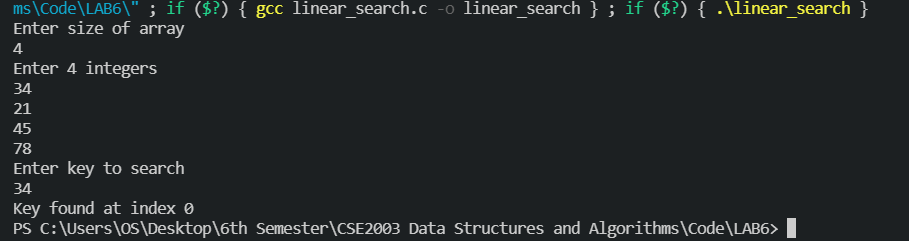
    else{

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

    }

}

**Output:**



**Task 5: Implementing Merge Sort using C.**

**Code:**

*// Merge sort in C*

#include <stdio.h>

*// Merge two subarrays L and M into arr*

void merge(int arr[], int p, int q, int r) {

*// Create L ← A[p..q] and M ← A[q+1..r]*

  int n1 = q - p + 1;

  int n2 = r - q;

  int L[n1], M[n2];

  for (int i = 0; i < n1; i++)

    L[i] = arr[p + i];

  for (int j = 0; j < n2; j++)

    M[j] = arr[q + 1 + j];

*// Maintain current index of sub-arrays and main array*

  int i, j, k;

  i = 0;

  j = 0;

  k = p;

*// Until we reach either end of either L or M, pick larger among*

*// elements L and M and place them in the correct position at A[p..r]*

  while (i < n1 && j < n2) {

    if (L[i] <= M[j]) {

      arr[k] = L[i];

      i++;

    } else {

      arr[k] = M[j];

      j++;

    }

    k++;

  }

*// When we run out of elements in either L or M,*

*// pick up the remaining elements and put in A[p..r]*

  while (i < n1) {

    arr[k] = L[i];

    i++;

    k++;

  }

  while (j < n2) {

    arr[k] = M[j];

    j++;

    k++;

  }

}

*// Divide the array into two subarrays, sort them and merge them*

void mergeSort(int arr[], int l, int r) {

  if (l < r) {

*// m is the point where the array is divided into two subarrays*

    int m = l + (r - l) / 2;

    mergeSort(arr, l, m);

    mergeSort(arr, m + 1, r);

*// Merge the sorted subarrays*

    merge(arr, l, m, r);

  }

}

*// Print the array*

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

  for (int i = 0; i < size; i++)

    printf("%d ", arr[i]);

  printf("\n");

}

*// Driver program*

int main() {

  int arr[] = {6, 5, 12, 10, 9, 1};

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

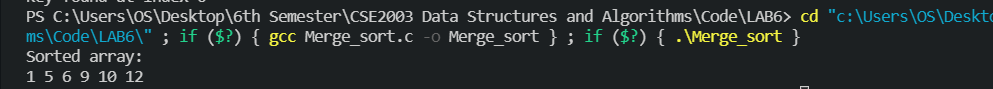
  mergeSort(arr, 0, size - 1);

  printf("Sorted array: \n");

  printArray(arr, size);

}

**Output:**



**Conclusion**

Hence, we have learnt how to implement some of the most popular searching and sorting algorithms. Some of these were using brute force and the others were using a more sophisticated divide and conquer strategy. Thus, the experiment is complete.

*Neh Samir Joshi*

*19BEC1112*