Designed

By

B.P.N.V.S.Sudheer

|  |
| --- |
| 1.Explain Linear Search Using Arrays |
| Description |
| A linear search is the simplest method of searching a data set.  Starting at the beginning of the data set, each item of data is checked until a match is made. Once the item is found, the search ends. If there is no match, the algorithm must deal with this.  A written description algorithm for a linear search might be:  1.Find out the length of the data set.  2.Set counter to 0.  3.Examine value held in the list at the counter position.  4.Check to see if the value at that position matches the value searched for.  5.If it matches, the value is found. Send a message and end the search.  6.If not, increment the counter by 1 and go back to step 3 until there are no more items to search.  7.If all the items have been checked and no match is found, send a message.  Consider this list of unordered numbers:  Table with a list of unordered numbers  Suppose we were to search for the value 2 in this list. The search would start at position 0 and check the value held there, in this case 3. 3 does not match 2, so we move on to the next position.  The value at position 1 is 5. 5 does not match 2, so we move on to the next position.  The value at position 2 is 2 - a match. The search ends. |
|  |
|  |
|  |
| Code : |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace linear\_search  {  internal class Program  {  static void Main(string[] args)  {  int[] array = new int[4] { 10, 20, 30, 40, };    Console.WriteLine("input stu id");  int stuid =Convert.ToInt32(Console.ReadLine());  int index = linear\_search(array, stuid);  if (index < 0)  {  Console.WriteLine($" stud id is not found");  Console.ReadLine();  }  else  {  Console.WriteLine($" stud id is found");  Console.ReadLine();  }  int linear\_search(int[] arr,int x )  {  for (int i = 0; i < array.Length; i++)  {  if (array[i] == x)  {  return i;  }  }  return -1;    }    }  }  } |
| Output : |

|  |
| --- |
| 2.Explain Bubble sort program using arrays |
|  |
| Description |
| Bubble Sort is a simple algorithm which is used to sort a given set of n elements provided in form of an array with n number of elements. Bubble Sort compares all the element one by one and sort them based on their values.  If the given array has to be sorted in ascending order, then bubble sort will start by comparing the first element of the array with the second element, if the first element is greater than the second element, it will swap both the elements, and then move on to compare the second and the third element, and so on.  If we have total n elements, then we need to repeat this process for n-1 times.  It is known as bubble sort, because with every complete iteration the largest element in the given array, bubbles up towards the last place or the highest index, just like a water bubble rises up to the water surface.  Sorting takes place by stepping through all the elements one-by-one and comparing it with the adjacent element and swapping them if required.  Following are the steps involved in bubble sort(for sorting a given array in ascending order):  1.Starting with the first element(index = 0), compare the current element with the next element of the array.  2 .If the current element is greater than the next element of the array, swap them.  3.If the current element is less than the next element, move to the next element. Repeat Step 1. |
|  |
| Code : |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace bubble\_sort  {  internal class Program  {  static void Main(string[] args)  {  int[] arr = new int[5] { 20, 40, 30, 60, 50, };  int temp = 0;  for (int i = 0; i < arr.Length; i++)  {  for (int j = 0; j < arr.Length; j++)  {  if (arr[i] < arr[j])  {  temp = arr[i];  arr[i] = arr[j];  arr[j] = temp;  }  }  }  for (int i = 0; i <= arr.Length; i++)  {  Console.WriteLine("{0}", arr[i]);  }  }  }  } |
| Output : |

|  |
| --- |
| 3.Explain binary search using arrays |
|  |
| It is a divide and conquer technique in binary search the search process always focussed on middle element of the list in order to perform binary search the element of the list should be arranged in a sorted order in binary search we calculate the location of the middle element then that element is compared with compared with key element  1.If the key element is equal to the middle element then we conclude that the element was found and will stop the searching process  2.If the key element is less than the middle element then we divide the array from the first element to middle element and repeat the same process by calculating middle element  3.If the key element is greater than the middle element,then we divide the array from middle element to last element and repeat the same process |