|  |  |
| --- | --- |
| **Name** | Adwait S Purao |
| **UID no.** | 2021300101 |
| **Experiment No.** | 5 |

|  |  |
| --- | --- |
| **AIM:** | Demonstrate the use of one-dimensional arrays to solve a given problem. |
| **Program 1** | |
| **PROBLEM STATEMENT :** | The selection sort algorithm sorts an array by repeatedly finding the  minimum element (considering ascending order) from the unsorted part and  putting it at the beginning. The algorithm maintains two sub-arrays in a given  array.  1) The sub-array which is already sorted.  2) Remaining sub-array which is unsorted.  Every iteration of selection sort, the minimum element (considering  ascending order) from the unsorted sub-array is picked and moved to the  sorted sub-array.  Following example explains the above steps:  arr[] = 64 25 12 22 11  // Find the minimum element in arr[0...4]  // and place it at beginning  11 25 12 22 64  // Find the minimum element in arr[1...4]  // and place it at beginning of arr[1...4]  11 12 25 22 64  // Find the minimum element in arr[2...4]  // and place it at beginning of arr[2...4]  11 12 22 25 64  // Find the minimum element in arr[3...4]  // and place it at beginning of arr[3...4]  11 12 22 25 64 |
| **ALGORITHM:** | 1.START  2.Define function sort with integer array and integer parameter n  3.Define integer variables min,ind  4. for(i=0; i<n; i++)  Min=arr[i]  for(j=i+1; j<n; j++)  5. if(min>arr[j])  min=arr[j];  ind = j;  arr[ind]=arr[i];  arr[i]=min  6.Define function main  7.Enter number of elements  8.Input all elements in arr[n]  9.Call function sort(arr, n)  10.Print sorted array  11.STOP |
| **FLOWCHART:** |  |
| **PROGRAM:** | #include<stdio.h>  void sort(int arr[],int n)  {  int min,ind;  for(int i=0;i<n;i++)  {  min = arr[i];  for(int j=i+1;j<n;j++)  {  if(min>arr[j])  {  min=arr[j];  ind = j;  }  }  arr[ind]=arr[i];  arr[i]=min;  }  }  int main()  {  int n;  printf("Enter no of elements in the array: ");  scanf("%d",&n);  int arr[n];  printf("Enter the value of elements in the array: ");  for(int i=0;i<n;i++)  {  scanf("%d",&arr[i]);  }  sort(arr,n);  printf("Sorted array:\n ");  for(int i=0;i<n;i++)  {  printf("%d ",arr[i]);  }  return 0;  } |
| **RESULT:** | |
| **Program 2** | |
| **PROBLEM STATEMENT :** | Perform search of a particular element on the above array using binary  search. Binary Search will search a sorted array by repeatedly dividing the  search interval in half. Begin with an interval covering the whole array. If the  value of the search key is less than the item in the middle of the interval,  narrow the interval to the lower half. Otherwise narrow it to the upper half.  Repeatedly check until the value is found or the interval is empty. We  basically ignore half of the elements just after one comparison.  1. Compare x with the middle element.  2. If x matches with middle element, we return the mid index.  3. Else If x is greater than the mid element, then x can only lie in right half  subarray after the  mid element. So we recur for right half.  4. Else (x is smaller) recur for the left half |
| **ALGORITHM:** | 1.START  2.Define function void bin\_srch with integer array and integer parameters size and num  3.Define integer variables hi,lo=0,i  4.for(i=(size/2); arr[i]!=num; i=((hi+lo)/2))  5.if(arr[i]<num)  Then lo = i+1  6.else if(arr[i]>num)  Then hi = i-1  7.if(i==0 || i==size-1)  Then break  8.if(arr[i]==num)  Print arr[i]=i  9.else print element does not exist in array  10.Define function main  11.Define integer parameters numb and n  12.Input total number of elements of array  13.Input the elements of the array in an array in ascending order  14.Input the element to be searched  15.Call function bin\_srch  16.STOP |
| **PROGRAM:** | #include <stdio.h>  void bin\_srch(int arr[80], int size, int num)  {  int hi=size-1, lo=0, i;  for(i=(size/2); arr[i]!=num; i=((hi+lo)/2))  {  if(arr[i]<num){lo = i+1;}  else if(arr[i]>num){hi = i-1;}  if(i==0 || i==size-1){break;}  }  if(arr[i]==num){printf("arr[%d] = %d", i, arr[i]);}  else{printf("Element doesnt exist in the array");}  }  int main()  {  int numb,n;  printf("Enter the total number of elements of the array: ");  scanf("%d",&n);  int arr[n];  printf("Enter the elements of the array in ascending order:\n");  for(int i=0;i<n;i++)  {  scanf("%d",&arr[i]);  }  printf("Enter the element to be searched: \n");  scanf("%d",&numb);  bin\_srch(arr,n,numb);  return 0;} |
| **RESULT:** | |
| **Program 3** | |
| **PROBLEM STATEMENT:** | Circular Array Rotation means rotating the elements in the array where one  rotation operation moves the first element of the array to the last position  and shifts all remaining elements to the left. Here, we are given an unsorted  array and our task is to perform the circular rotation by n number of rotations  where n is a natural number.  Initial Array: [ 1 2 3 4 5 ]  After one rotation : [ 2 3 4 5 1]  After two rotation : [ 3 4 5 1 2 ] |
| **ALGORITHM:** | 1.START  2. Define function void circ() with integer array arr[] and two integer  parameter m and n  3. for(i=1;i<=m;i++)  Store the value of arr[0] in temp  4. for (int j = 0; j < n - 1; j++)  arr[j] = arr[j + 1];  arr[n-1]=temp  5. Define function main()  6. Input no of elements in array n  7. Input the no. of rotations m  8. Input integer array arr[n]  9. Call function circ(arr,n,m)  10.STOP |
| **PROGRAM:** | #include<stdio.h>  void circ(int arr[],int n,int m)  {  int temp,i,j;  for(i=1;i<=m;i++)  {  temp = arr[0];  for (int j = 0; j < n - 1; j++)  {  arr[j] = arr[j + 1];  }  arr[n-1]=temp;  }  for(int i=0;i<n;i++)  {  printf("%d ",arr[i]);  }  }  int main()  {  int m,n;  printf("Enter the number of elements in the array: ");  scanf("%d",&n);  int arr[n];  printf("Enter the elements of the array: ");  for(int i=0;i<n;i++)  {  scanf("%d",&arr[i]);  }  printf("Enter the number of rotations: ");  scanf("%d",&m);  circ(arr,n,m);  return 0;  } |
| **RESULT:** | |
| **CONCLUSION:** | In the above experiment we learnt about 1 dimensional arrays. We learnt about their declaration and how to call it in a function as a parameter. We also learnt about bubble sort, binary search algorithms and how to rotate the elements of the array. |