**Ex No: EXP 1 – ARRAY DATA STRUCTURE**

**Date:**

**AIM:**

To design and implement a C++ program that performs a variety of array operations, including insertion, deletion, searching, sorting, and set operations such as union and intersection.

**ALGORITHM:**

1. **Initialize the Array:**  
   Prompt the user to enter the size of the array and its elements.
2. **Display Menu:**  
   Show a menu listing the available array operations and ask the user to select an option.
3. **Insert Element:**  
   Add a new element either at the end or at a specific position in the array.
4. **Display Elements:**  
   Traverse the array using a loop and display all elements.
5. **Reverse Array:**  
   Display the elements of the array in reverse order.
6. **Search by Value:**  
   Ask the user for a value and return its index if found.
7. **Search by Position:**  
   Ask the user for a position and return the corresponding element.
8. **Delete Element:**  
   Remove an element either from the end or a specified position in the array.
9. **Find Pair Close to Zero:**  
   Identify and display the pair of elements whose sum is closest to zero.
10. **Count Ones:**  
    Count the number of occurrences of the value 1 in the array.
11. **Remove Duplicates:**  
    Eliminate duplicate values from the array, preserving unique elements only.
12. **Union Operation:**  
    Combine two arrays and return a new array containing all unique elements from both.
13. **Intersection Operation:**  
    Return a new array containing only the elements that are common to both arrays.
14. **Search Operations:**  
    Implement and perform Linear Search, Binary Search, and Fibonacci Search to find a given element.
15. **Sorting Techniques:**  
    Sort the array in ascending order using Bubble Sort, Merge Sort, and Quick Sort algorithms.

**CODING:**

#include<iostream>

#include<cmath>

#include<algorithm>

using namespace std;

int insertAtEnd(int arr[],int n,int key,int capacity)

{

    if(n>=capacity)

    {

        return n;

    }

    arr[n]=key;

    return (n+ 1);

}

int insertAtPosition(int arr[],int n,int key,int pos,int capacity)

{

    for(int i=n- 1;i>=pos;i--)

    {

        arr[i+ 1]=arr[i];

    }

    arr[pos]=key;

}

int display(int arr[],int n)

{

    for(int i=0;i<n;i++)

    {

        cout<<arr[i]<<" ";

    }

}

int reverseElements(int arr[],int n)

{

    cout<<"Reversing the array elements:"<<endl;

    for(int i=n- 1;i>=0;i--)

    {

        cout<<arr[i]<<" ";

    }

}

int findElement(int arr[],int n,int key)

{

    for(int i=0;i<n;i++)

    {

        if(arr[i]==key)

        {

            return i;

        }

    }

    return -1;

}

int indexSearch(int arr[],int n)

{

    int v;

    cout<<"enter an element to find its position :";

    cin>>v;

    for(int i=0;i<n;i++)

    {

        if(arr[i]==v)

    {

        cout<<"the element is at the position : "<<i<<endl;

    }

    }

}

int deleteAtEnd(int arr[],int n)

{

    for(int i=0;i<n-1;i++)

    {

        cout<<arr[i]<<" ";

    }

}

int deleteAtPos(int arr[],int n)

{

    int m;

    cout<<"Enter the position of element to delete it: ";

    cin>>m;

    if(m>=n)

    {

        cout<<"no such position found"<<endl;

    }

    for(int i=m;i<n;i++)

    {

        arr[i]=arr[i+1];

    }

    for(int i=0;i<n-1;i++)

    {

        cout<<arr[i]<<" ";

    }

}

int sumclosetozero(int arr[],int n){

    sort(arr,arr + 1);

    int left = 0;

    int right = n - 1;

    int closestsum = 999999;

    int n1 = 0, n2 = 0;

    while(left < right){

        int sum = arr[left] + arr[right];

        if(abs(sum) < abs(closestsum)){

            closestsum = sum;

            n1 = arr[left];

            n2 = arr[right];

        }

        if(sum < 0){

            left = left + 1;

        }

        else{

            right = right - 1;

        }

    }

    cout<<"The two numbers whose sum is close to zero are "<<n1<<" and "<<n2<<" SUM = "<<closestsum<<endl;

    return closestsum;

}

int countones(int arr[],int n)

{

    int c=0;

    for (int i=0;i<n;i++)

    {

        if(arr[i]==1)

        {

            c++;

        }

    }

    cout<<"The count of one in the array is:"<<c<<endl;

}

int removedupl(int arr[],int n)

{

    int d=0;

    for(int i=0;i<n;i++)

    {

        for(int j=i+1;j<n;j++)

        {

            if(arr[i]==arr[j])

            {

                for(int k=j;k<n-1;k++)

                {

                    arr[k]=arr[k+1];

                }

                n--;

                j--;

            }

        }

    }

    cout<<"Array after removing duplicate elements: ";

    for(int k=0;k<n;k++)

    {

        cout<<arr[k]<<" ";

    }

}

int unionDupl(int arr[],int n,int arr2[],int n2)

{

    int arr3[100];

    int n1=0;

    for(int i=0;i<n;i++)

    {

        arr3[n1]=arr[i];

        n1++;

    }

    for(int i=0;i<n2;i++)

    {

        arr3[n1]=arr2[i];

        n1++;

    }

    cout<<"Union of arrays with duplicates:"<<endl;

    for(int i=0;i<n1;i++)

    {

        cout<<arr3[i]<<" ";

    }

}

void findIntersection(int arr[], int n, int arr2[], int n2) {

    int result[100];

    int k = 0;

    for (int i = 0; i < n; i++) {

        for (int j = 0; j < n2; j++) {

            if (arr[i] == arr2[j]) {

                bool found = false;

                for (int x = 0; x < k; x++) {

                    if (result[x] == arr[i]) {

                        found = true;

                        break;

                    }

                }

                if (!found) {

                    result[k++] = arr[i];

                }

            }

        }

    }

    cout << "Intersection of arrays: ";

    for (int i = 0; i < k; i++) {

        cout << result[i] << " ";

    }

    cout << endl;

}

int linearSearch(int arr[],int n,int key)

{

    for(int i=0;i<n;i++)

    {

        if(arr[i]==key)

        {

            return i;

        }

    }

    return -1;

}

 int binarySearch(int arr[],int low,int high,int key)

 {

     while(low<=high)

     {

         int mid=low+(high-low)/2;

         if(arr[mid]==key)

         {

             return mid;

         }

         if(arr[mid]<key)

         {

             low=mid+1;

         }

         else

         {

             high=mid-1;

         }

     }

 }

 int bubbleSort(int arr[],int n)

 {

     for(int i=0;i<n;i++)

     {

         for(int j=i+1;j<n;j++)

         {

             if(arr[i]>arr[j])

             {

                 int t=arr[i];

                 arr[i]=arr[j];

                 arr[j]=t;

             }

         }

     }

     cout<<"The sorted array is:"<<endl;

     for(int i=0;i<n;i++)

     {

         cout<<arr[i]<<" ";

     }

 }

 bool exists(int arr[], int size, int val) {

    for (int i = 0; i < size; i++)

        if (arr[i] == val)

            return true;

    return false;

}

int unionWithoutDup(int arr[], int n, int arr2[], int n2) {

    int result[100];

    int size = 0;

    for (int i = 0; i < n; i++) {

        if (!exists(result, size, arr[i])) {

            result[size++] = arr[i];

        }

    }

    for (int i = 0; i < n2; i++) {

        if (!exists(result, size, arr2[i])) {

            result[size++] = arr2[i];

        }

    }

    cout << "Union two arrays without duplicate:"<<endl;

    for (int i = 0; i < size; i++) {

        cout << result[i] << " ";

    }

    cout << endl;

}

int insertionSort(int arr[],int n)

{

    for(int i=1;i<n;i++)

    {

        int key=arr[i];

        int j=i-1;

        while(j>=0&&arr[j]>key)

        {

            arr[j+1]=arr[j];

            j--;

        }

        arr[j+1]=key;

    }

    for(int i=0;i<n;i++)

    {

        cout<<arr[i]<<" ";

    }

}

int mergea(int arr[],int left,int mid,int right)

{

    int n1=mid-left+1;

    int n2=right-mid;

    int l[n1],r[n2];

    for(int i=0;i<n1;i++)

    {

        l[i]=arr[left+i];

    }

    for(int j=0;j<n2;j++)

    {

        r[j]=arr[mid+1+j];

    }

    int i=0,j=0,k=left;

    while(i<n1&&j<n2)

    {

        if(l[i]<=r[j])

        {

            arr[k++]=l[i++];

        }

        else

            {

               arr[k++]=r[j++];

            }

    }

        while(i<n1)

        {

            arr[k++]=l[i++];

        }

        while(j<n2)

        {

            arr[k++]=r[j++];

        }

}

int mergeSort(int arr[],int left,int right)

{

    if(left<right)

    {

        int mid=left+(right-left)/2;

        mergeSort(arr,left,mid);

        mergeSort(arr,mid+1,right);

        mergea(arr,left,mid,right);

    }

}

int min(int x,int y){

    return (x <= y)? x : y;

}

int fibosearch(int arr[],int key,int n){

    int fib2 = 0;

    int fib1 = 1;

    int fib = fib2 + fib1;

    while(fib < n){

        fib2 = fib1;

        fib1 = fib;

        fib = fib2 + fib1;

    }

    int offset = -1;

    while(fib > 1){

        int i = min(offset + fib2 , n - 1);

        if(arr[i] < key){

            fib = fib1;

            fib1 = fib2;

            fib2 = fib - fib1;

            offset = i;

        }

        else if(arr[i] > key){

            fib = fib2;

            fib1 = fib1 - fib2;

            fib2 = fib - fib1;

        }

        else{

            return i;

        }

    }

    if(fib1 && arr[offset + 1] == key){

        return offset + 1;

    }

    return -1;

}

void swap(int &a,int &b){

    int temp = a;

    a = b;

    b = temp;

}

int partition(int arr[],int low,int high){

    int pivot = arr[high];

    int i = low - 1;

    for(int j = low;j < high;j++){

        if(arr[j] < pivot){

            i++;

            swap(arr[i], arr[j]);

        }

    }

    swap(arr[i + 1], arr[high]);

    return i + 1;

}

void Quicksort(int arr[],int low,int high){

    if(low < high){

        int pi = partition(arr,low,high);

        Quicksort(arr,low,pi - 1);

        Quicksort(arr,pi + 1,high);

    }

}

int main()

{

    cout<<"NAME : PRAVIN."<<"ROLL NO: 24104043\n\n\n";

    int n,capacity,key,choice;

    int arr[100] = {1,2,3,4,5,65,5,43,2,34,5,56};

    n = 11;

    capacity=sizeof(arr)/sizeof(arr[0]);

    cout<<"1.Insert at the end."<<endl;

    cout<<"2.Insert at given position."<<endl;

    cout<<"3.Display the array elements."<<endl;

    cout<<"4.Display in reverse order."<<endl;

    cout<<"5.Finding an element."<<endl;

    cout<<"6.Find the element in given position."<<endl;

    cout<<"7.Deleting at specific position."<<endl;

    cout<<"8.Delete at end."<<endl;

    cout<<"9.Sum close to zero."<<endl;

    cout<<"10.Count the ones in array."<<endl;

    cout<<"11.Remove the duplicates."<<endl;

    cout<<"12.Union of array with duplicates."<<endl;

    cout<<"13.Intersection of two arrays."<<endl;

    cout<<"14.Linear search."<<endl;

    cout<<"15.Binary search."<<endl;

    cout<<"16.Bubble sort."<<endl;

    cout<<"17.Union without duplicates."<<endl;

    cout<<"18.Insertion sort"<<endl;

    cout<<"19.Merge sort."<<endl;

    cout<<"20.Fibonacci Search."<<endl;

    cout<<"21.Quick Sort."<<endl;

    cout<<"22.EXIT."<<endl;

    do{

        cout<<"\nEnter your choice(1-22) : ";

        cin>>choice;

        switch(choice)

        {

        case 1:

            {

                cout<<"----INSERT ELEMENT AT END----"<<endl;

            cout<<"Enter the element to be inserted :";

        cin>>key;

        n=insertAtEnd(arr,n,key,capacity);

        for(int i=0;i<n;i++)

        {

            cout<<arr[i]<<" ";

        }

        break;

            }

        case 2:

            {

                int pos;

                cout<<"----INSERT ELEMENT AT SPECIFIC POSITION----"<<endl;

                cout<<"Enter the elements position :";

                cin>>pos;

                cout<<"Enter the element to be inserted :";

                cin>>key;

                insertAtPosition(arr,n,key,pos,capacity);

                n++;

                for(int i=0;i<n;i++)

                    {

                        cout<<arr[i]<<" ";

                    }

                    break;

            }

        case 3:

            {

                cout<<"----DISPLAY ARRAY ELEMENTS----"<<endl;

               cout<<"The array is :";

               display(arr,n);

               break;

            }

        case 4:

                {

                    cout<<"----REVERSE ELEMENTS IN ARRAY----"<<endl;

                    n=reverseElements(arr,n);

                    break;

                }

        case 5:

            {

                cout<<"----FINDING A ELEMENT----"<<endl;

                cout<<"Enter the element to be searched :";

                cin>>key;

                n=findElement(arr,n,key);

                if(n==-1)

                {

                    cout<<"Element not found";

                }

                else

                {

                    cout<<"Element is present at the index :"<<n<<endl;

                }

                break;

            }

        case 6:

            {

                cout<<"----FIND ELEMENT AT GIVEN POSITION----"<<endl;

                n=indexSearch(arr,n);

                break;

            }

        case 7:

            {

                cout<<"----DELETE AT SPECIFIC POSITION----"<<endl;

                n=deleteAtPos(arr,n);

                break;

            }

        case 8:

            {

                cout<<"----DELETE AT END----"<<endl;

                n=deleteAtEnd(arr,n);

                break;

            }

        case 9:

            {

                cout<<"----SUM CLOSES TO ZERO----"<<endl;

               sumclosetozero(arr,n);

               break;

            }

        case 10:

            {

                cout<<"----COUNT 1S IN ARRAY----"<<endl;

                countones(arr,n);

                break;

            }

        case 11:

            {

                cout<<"----REMOVE DUPLICATES----"<<endl;

                removedupl(arr,n);

                break;

            }

        case 12:

            {

                cout<<"----UNION OF ARRAY WITH DUPLICATES----"<<endl;

            int n2= 6;

            int arr2[20]={1,14,10,16,27,79};

            unionDupl(arr,n,arr2,n2);

            break;

            }

        case 13:

            {

                cout<<"----INTERSECTION OF TWO ARRAY----"<<endl;

                int arr2[100]={1,44,3,5,4};

                int arr3[100];

                int n2=sizeof(arr2)/sizeof(arr2[0]);

                findIntersection(arr,n,arr2,n2);

                break;

            }

        case 14:

            {

                cout<<"----LINEAR SEARCH----"<<endl;

                cout<<"Enter the element to be searched:";

                cin>>key;

                int res =linearSearch(arr,n,key);

                if(res==-1)

                {

                    cout<<"Element not found";

                }

                else

                {

                    cout<<"Element is present at the index: "<<res<<endl;

                }

                break;

            }

        case 15:

            {

                cout<<"----BINARY SEARCH----"<<endl;

                int key;

                cout<<"Enter the element to be searched";

                cin>>key;

                int r=binarySearch(arr,0,n-1,key);

                if(r==-1)

                {

                    cout<<"Element is not present in the array "<<endl;

                }

                else{

                    cout<<"Element is present at the index :"<<r<<endl;

                }

                break;

            }

        case 16:

            {

                cout<<"----BUBBLESORT----"<<endl;

                bubbleSort(arr,n);

                break;

            }

        case 17:

            {

                cout<<"----UNION WITHOUT DUPLICATES----"<<endl;

            int arr2[20]={1,14,10,16,27,79};

            int n2=6;

            unionWithoutDup(arr,n,arr2,n2);

          break ;

            }

        case 18:

            {

                cout<<"----INSERTION SORT----"<<endl;

                insertionSort(arr,n);

                break;

            }

        case 19:

            {

                cout<<"----MERGE SORT----"<<endl;

                mergeSort(arr,0,n-1);

                cout<<"The sorted array is:"<<endl;

                for(int i=0;i<n;i++)

                    {

                        cout<<arr[i]<<" ";

                    }

                    break;

            }

        case 20:

            {

                cout<<"----FIBONACCI SEARCH----"<<endl;

            cout<<"Enter the element to be searched :";

            cin>>key;

            int ind = fibosearch(arr,key,n);

            if(ind >= 0){

                cout<<"Element is found at index "<<ind<<endl;

            }

            else{

                cout<<"Element is not found "<<endl;

            }

            break;

            }

        case 21:{

            cout<<"----QUICK SORT----"<<endl;

             Quicksort(arr,0,n - 1);

            cout<<"Array after sorting"<<endl;

            for(int i=0;i<n;i++){

                cout<<arr[i]<<" ";

            }

            cout<<endl;

            break;

        }

        default:{

            cout<<"\n\nCOME AGAIN!\n\n"<<endl;

            break;

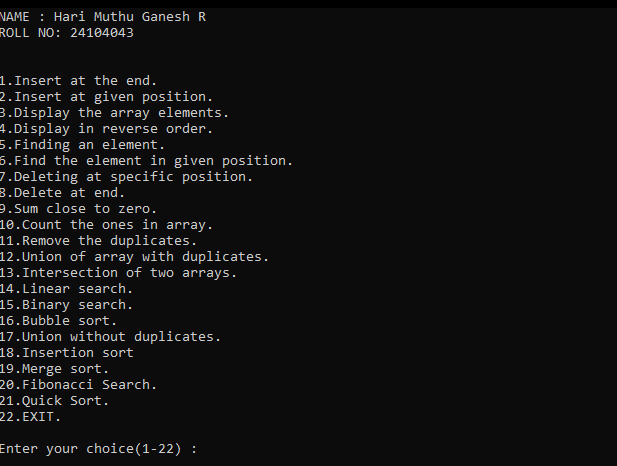
        }

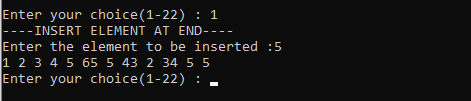
        }

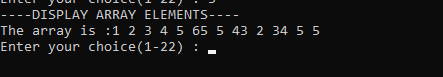
    }while(choice!=22);

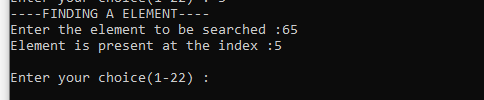
    return 0;

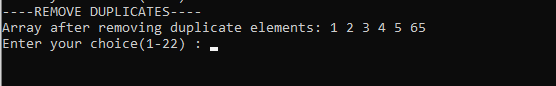
}

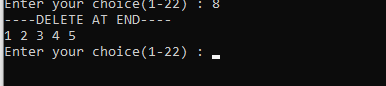


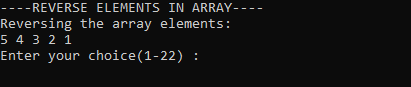


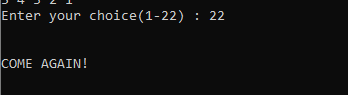












|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PROBLEM  IDENTIFICATION  (2) | IMPLEMENTATION/ EXECUTION (3) | TIME MANAGEMENT (2) | VIVA (3) | TOTAL  (10) |
|  |  |  |  |  |

**RUBRICS :**

**RESULT :**

Thus the program were executed successfully on array operation and the output were verified

**s**