ARRAY PROGRAMS IN C++

Write a C++ program to find the sum and average of one dimensional integer array.

#include<iostream>

using namespace std;

int main()

{

int Arr[100],n,i,sum=0;

cout<<"Enter number of elements you want to insert ";

cin>>n;

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

{

cout<<"Enter element "<<i+1<<":";

cin>>Arr[i];

}

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

sum+=Arr[i];

cout<<"\nThe sum of Array is :"<<sum;

cout<<"\nThe average of Array is :"<<sum/i;

return 0;

}

Write a C++ program to swap first and last element of an integer 1-d array.

#include<iostream>

using namespace std;

int main()

{

int Arr[100],n,i,temp;

cout<<"Enter number of elements you want to insert ";

cin>>n;

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

{

cout<<"Enter element "<<i+1<<":";

cin>>Arr[i];

}

temp=Arr[0];

Arr[0]=Arr[n-1];

Arr[n-1]=temp;

cout<<"\nArray after swapping"<<endl;

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

cout<<Arr[i]<<" ";

return 0;

}

Write a C++ program to reverse the element of an integer 1-D array.

include<iostream>

using namespace std;

int main()

{

int Arr[100],n,temp,i,j;

cout<<"Enter number of elements you want to insert ";

cin>>n;

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

{

cout<<"Enter element "<<i+1<<":";

cin>>Arr[i];

}

for(i=0,j=n-1;i<n/2;i++,j--)

{

temp=Arr[i];

Arr[i]=Arr[j];

Arr[j]=temp;

}

cout<<"\nReverse array"<<endl;

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

cout<<Arr[i]<<" ";

return 0;

}

Write a C++ program to find the largest and smallest element of an array.

#include<iostream>

using namespace std;

int main()

{

int Arr[100],n,i,small,large;

cout<<"Enter number of elements you want to insert ";

cin>>n;

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

{

cout<<"Enter element "<<i+1<<":";

cin>>Arr[i];

}

small=Arr[0];

large=Arr[0];

for(i=1;i<n;i++)

{

if(Arr[i]<small)

small=Arr[i];

if(Arr[i]>large)

large=Arr[i];

}

cout<<"\nLargest element is :"<<large;

cout<<"\nSmallest element is :"<<small;

return 0;

}

Write a menu driven C++ program with following option   
a. Accept elements of an array   
b. Display elements of an array   
c. Sort the array using insertion sort method   
d. Sort the array using selection sort method   
e. Sort the array using bubble sort method   
Write C++ functions for all options. The functions should have two parameters name of the array and number of elements in the array.

#include<iostream>

using namespace std;

void accept(int Arr[], int s);

void display(int Arr[], int s);

void isort(int Arr[], int s);

void ssort(int Arr[], int s);

void bsort(int Arr[],int s);

int main()

{

int Arr[100],n,choice;

cout<<"Enter Size of Array ";

cin>>n;

do

{

cout<<"\n\nMENU";

cout<<"\n1. Accept elements of array";

cout<<"\n2. Display elements of array";

cout<<"\n3. Sort the array using insertion sort method";

cout<<"\n4. Sort the array using selection sort method";

cout<<"\n5. Sort the array using bubble sort method";

cout<<"\n6. Exit";

cout<<"\n\nEnter your choice 1-6 :";

cin>>choice;

switch(choice)

{

case 1: accept(Arr,n);

break;

case 2: display(Arr,n);

break;

case 3: isort(Arr,n);

break;

case 4: ssort(Arr,n);

break;

case 5: bsort(Arr,n);

break;

case 6: break;

default:cout<<"\nInvalid choice";

}

}while(choice!=6);

return 0;

}

void accept(int Arr[], int s)

{

for(int I=0;I<s;I++)

{

cout<<"Enter element "<<I+1<<":";

cin>>Arr[I];

}

}

void display(int Arr[], int s)

{

cout<<"The elements of the array are:\n";

for(int I=0;I<s;I++)

cout<<Arr[I]<<" ";

}

void isort(int Arr[], int s)

{

int I,J,Temp;

for(I=1;I<s;I++)

{

Temp=Arr[I];

J=I-1;

while((Temp<Arr[J]) && (J>=0))

{

Arr[J+1]=Arr[J];

J--;

}

Arr[J+1]=Temp;

}

}

void ssort(int Arr[], int s)

{

int I,J,Temp,Small;

for(I=0;I<s-1;I++)

{

Small=I;

for(J=I+1;J<s;J++) *//finding the smallest element*

if(Arr[J]<Arr[Small])

Small=J;

if(Small!=I)

{

Temp=Arr[I]; *//Swapping*

Arr[I]=Arr[Small];

Arr[Small]=Temp;

}

}

}

void bsort(int Arr[],int s)

{

int I,J,Temp;

for(I=0;I<s-1;I++)

{

for(J=0;J<(s-1-I);J++)

if(Arr[J]>Arr[J+1])

{

Temp=Arr[J]; *//swapping*

Arr[J]=Arr[J+1];

Arr[J+1]=Temp;

}

}

}

P is one-dimensional array of integers. Write a C++ function to efficiently search for a data VAL from P. If VAL is present in the array then the function should return value 1 and 0 otherwise.

#include<iostream>

using namespace std;

bool lsearch(int Arr[], int s, int VAL);

int main()

{

int Arr[100],n,val;

bool found;

cout<<"Enter number of elements you want to insert ";

cin>>n;

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

{

cout<<"Enter element "<<i+1<<":";

cin>>Arr[i];

}

cout<<"Enter the number you want to search ";

cin>>val;

found=lsearch(Arr,n,val);

if(found)

cout<<"\nItem found";

else

cout<<"\nItem not found";

return 0;

}

bool lsearch(int Arr[], int s, int VAL)

{

for(int I=0; I<s; I++)

{

if(Arr[I]==VAL)

return **true**;

}

return **false**;

}

Suppose a one-dimensional array AR containing integers is arranged in ascending order. Write a user-defined function in C++ to search for an integer from AR with the help of Binary search method, returning an integer 0 to show absence of the number and integer 1 to show presence of the number in the array. Function should have three parameters : (i) array AR (ii) the number to be searched and (iii) the number of elements N in the array.

#include<iostream>

using namespace std;

bool bsearch(int AR[], int N, int VAL);

int main()

{

int AR[100],n,val;

bool found;

cout<<"Enter number of elements you want to insert ";

cin>>n;

cout<<"Enter element in ascending order\n";

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

{

cout<<"Enter element "<<i+1<<":";

cin>>AR[i];

}

cout<<"\nEnter the number you want to search ";

cin>>val;

found=bsearch(AR,n,val);

if(found)

cout<<"\nItem found";

else

cout<<"\nItem not found";

return 0;

}

bool bsearch(int AR[], int N, int VAL)

{

int Mid,Lbound=0,Ubound=N-1;

while(Lbound<=Ubound)

{

Mid=(Lbound+Ubound)/2;

if(VAL>AR[Mid])

Lbound=Mid+1;

else if(VAL<AR[Mid])

Ubound=Mid-1;

else

return **true**;

}

return **false**;

}

Suppose A, B, C are arrays of integers of size M, N, and M + N respectively. The numbers in array A appear in ascending order while the numbers in array B appear in descending order. Write a user defined function in C++ to produce third array C by merging arrays A and B in ascending order. Use A, B and C as arguments in the function.

#include<iostream>

using namespace std;

void Merge(int A[], int B[], int C[], int N, int M, int &K);

int main()

{

int A[100], B[100], C[200],i,n,m,k;

cout<<"\nEnter number of elements you want to insert in first array ";

cin>>n;

cout<<"Enter element in ascending order\n";

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

{

cout<<"Enter element "<<i+1<<":";

cin>>A[i];

}

cout<<"\nEnter number of elements you want to insert in second array ";

cin>>m;

cout<<"Enter element in descending order\n";

for(i=0;i<m;i++)

{

cout<<"Enter element "<<i+1<<":";

cin>>B[i];

}

Merge(A,B,C,n,m,k);

cout<<"\nThe Merged Array in Ascending Order"<<endl;

for(i=0;i<k;i++)

{

cout<<C[i]<<" ";

}

return 0;

}

void Merge(int A[], int B[], int C[], int N, int M, int &K)

{

int I=0, J=M-1;

K=0;

while (I<N && J>=0)

{

if (A[I]<B[J])

C[K++]=A[I++];

else if (A[I]>B[J])

C[K++]=B[J--];

else

{

C[K++]=A[I++];

J--;

}

}

for (int T=I;T<N;T++)

C[K++]=A[T];

for (int T=J;T>=0;T--)

C[K++]=B[T];

}

Suppose X. Y, Z are arrays of integers of size M, N, and M + N respectively. The numbers in array X and Y appear in descending order. Write a user-defined function in C++ to produce third array Z by merging arrays X and Y in descending order.

#include<iostream>

using namespace std;

void Merge(int A[], int B[], int C[], int N, int M, int &K);

int main()

{

int A[100], B[100], C[200],i,n,m,k;

cout<<"\nEnter number of elements you want to insert in first array ";

cin>>n;

cout<<"Enter element in descending order\n";

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

{

cout<<"Enter element "<<i+1<<":";

cin>>A[i];

}

cout<<"\nEnter number of elements you want to insert in second array ";

cin>>m;

cout<<"Enter element in descending order\n";

for(i=0;i<m;i++)

{

cout<<"Enter element "<<i+1<<":";

cin>>B[i];

}

Merge(A,B,C,n,m,k);

cout<<"\nThe Merged Array in Descending Order"<<endl;

for(i=0;i<k;i++)

{

cout<<C[i]<<" ";

}

return 0;

}

void Merge(int A[], int B[], int C[], int N, int M, int &K)

{

int I=0, J=0;

K=0;

while (I<N && J<M)

{

if (A[I]>B[J])

C[K++]=A[I++];

else if (A[I]<B[J])

C[K++]=B[J++];

else

{

C[K++]=A[I++];

J++;

}

}

for (int T=I;T<N;T++)

C[K++]=A[T];

for (int T=J;T<M;T++)

C[K++]=B[T];

}