# EXPERIMENT – 3

**AIM:** Write a program to demonstrate the functioning of the pointers.

**ALGORITHM:**

**SOURCE CODE:**

#include <iostream>

using namespace std;

int main() {

int a;

cout<<"Enter any integer value: ";

cin>>a;

int \*b;

b = &a;

int \*\*c;

c = &b;

cout<<"The value stored in the pointers are:"<<endl;

cout<<"a:"<<a<<endl;

cout<<"b: "<<b<<endl;

cout<<"c: "<<c<<endl;

cout<<"&a: "<<&a<<endl;

cout<<"&b: "<<&b<<endl;

cout<<"\*a: "<<\*b<<endl;

cout<<"\*b: "<<\*c<<endl;

cout<<"\*\*c: "<<\*\*c<<endl;

return 0;

}

**EXPERIMENT-3**

**AIM:** Write a program to demonstrate the functioning of the pointers.

**OUTPUT:**

**Text

Description automatically generatedEXPERIMENT-4**

**AIM:** Perform Linear Search and Binary Search on an array. Description of programs:

1. Read an array of type integer.
2. Input element from user for searching.
3. Search the element by passing the array to a function and then returning the position of the element from the function else return -1 if the element is not found.
4. Display the position where the element has been found.

**ALGORITHM:**

**SOURCE CODE:**

#include <iostream>

using namespace std;

int linearSearch(int arr[10],int size,int e){

for (int i=0;i<size;i++){

if (arr[i] == e){

return i;

}

}

return -1;

}

int binarySearch(int arr[10],int size,int a){

int min= 0, max= size-1;int mid;

while(min<=max){

mid = (min + max)/2;

if (a == arr[mid]){

return mid;

}

else if (a > arr[mid]){

min = mid + 1;

}

else{

max = mid - 1;

}

}

}

int main() {

int arr[] = {1,2,4,6,9,12,14,15,19,23,27,34};

int a,r,t;

int size = sizeof(arr)/sizeof(int);

cout<<"The input array is: ";

for (int i=0;i<size;i++){

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

}

cout<<endl;

cout<<"Element to search: "; cin>>a;

cout<<"USING: 1.LINEAR SEARCH 2.BINARY SEARCH "<<endl;cin>>t;

if(t==1){

r = linearSearch(arr,size,a);

cout<<"The element is found at index "<<r;

}

if(t==2){

r=binarySearch(arr,size,a);

cout<<"The element is found at index "<<r;

}

return 0;

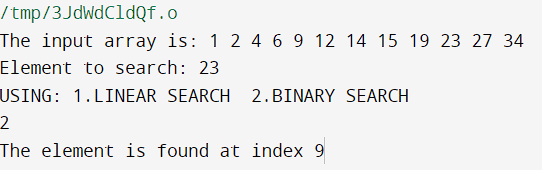
}

**EXPERIMENT-4**

**AIM:** Perform Linear Search and Binary Search on an array. Description of programs:

1. Read an array of type integer.
2. Input element from user for searching.
3. Search the element by passing the array to a function and then returning the position of the element from the function else return -1 if the element is not found.
4. Display the position where the element has been found.

**OUTPUT:**

****

**Text

Description automatically generated**

**EXPERIMENT-1**

**AIM:** Write a program to insert and delete an element from an array.

**ALGORITHM:**

**SOURCE CODE:**

#include <iostream>

using namespace std;

void insertElement(int arr[], int n, int x, int pos)

{

for (int i = n - 1; i >= pos; i--){

arr[i + 1] = arr[i];}

arr[pos] = x;

}

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

int pos;

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

if (arr[i] == key){

pos =i;

break;}

else{

pos=-1; }

}

if (pos==-1){

cout<<"NOT FOUND";

return n;}

for (int i = pos; i < n - 1; i++){

arr[i] = arr[i + 1];}

return n - 1;

}

int main() {

int arr[100],size;

int x,p;

cout<<"size of the array: "; cin>>size;

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

for (int i =0;i<size;i++){

cin>>arr[i]; }

cout<<"ARRAY: "; for (int i=0;i<size;i++){

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

}

cout<<endl;

cout<<"Enter the value to be inserted and its position: "; cin>>x>>p;

insertElement(arr, size, x, p);

size++;

cout<<"NEW ARRAY : ";

for (int i = 0; i < size; i++){

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

}

cout<<endl;

cout<<"Enter the value to be deleted "; cin>>x;

size=deleteElement(arr, size,x);

cout << "\n\nArray after deletion\n";

for (int i = 0; i < size; i++){

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

}

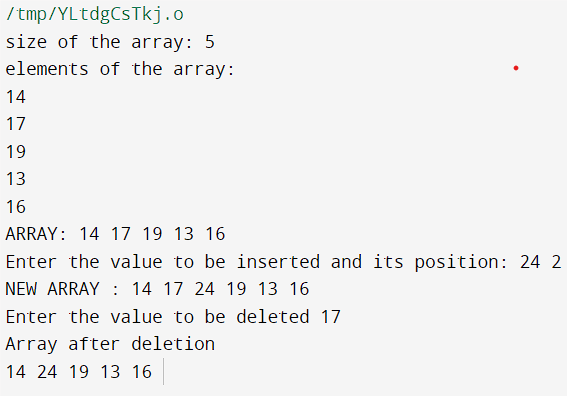
return 0;

}

**EXPERIMENT-1**

**AIM:** Write a program to insert and delete an element from an array.

**OUTPUT**:



**EXPERIMENT-2**

**AIM:** Implement sparse matrix using array. Description of program:

1. Read a 2D array from the user.
2. Store it in the sparse matrix form, use array of structures.
3. Print the final array.

**ALGORITHM:**

**SOURCE CODE:**

#include<iostream>

using namespace std;

int main()

{

int sparseMatrix[5][5] =

{

{0 , 0 , 3 , 0 , 0 },

{0 , 8 , 0 , 0 , 1 },

{0 , 0 , 0 , 7 , 0 },

{0 , 5 , 0 , 0 , 0 },

{6 , 0 , 9 , 0 , 0 }

};

for (int row = 0; row < 5; row++){

for (int column = 0; column < 5; column++){

cout<<sparseMatrix[row][column]<<" ";}

cout<<endl;}

cout<<endl;

int size = 0;

for (int row = 0; row < 5; row++){

for (int column = 0; column < 5; column++){

if (sparseMatrix[row][column] != 0){

size++;

}}}

int resultMatrix[size][3];

int k = 0;

for (int row = 0; row < 5; row++)

for (int column = 0; column < 5; column++)

if (sparseMatrix[row][column] != 0)

{

resultMatrix[k][0] = row;

resultMatrix[k][1] = column;

resultMatrix[k][2] = sparseMatrix[row][column];

k++;}

cout<<"R-Row C-Column V-Value"<<endl<<endl;

cout<<"R C V"<<endl;

for (int row=0; row<size; row++)

{

for (int column = 0; column<3; column++){

cout<<resultMatrix[row][column]<<" ";}

cout<<endl;

}

return 0;

}

**EXPERIMENT-2**

**AIM:** Implement sparse matrix using array. Description of program:

1. Read a 2D array from the user.
2. Store it in the sparse matrix form, use array of structures.
3. Print the final array.

**OUTPUT:**

**A picture containing table

Description automatically generated**