**SEARCHING ALGORITHMS**

**LAB # 04**

**Data Structures & Algorithms**

Submitted by: **Shah Raza**

Registration No: **18PWCSE1658**

Class Section: **B**

"On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work."

Student Signature: _______________

Submitted to: **Dr. Khurram Shehzad Khattak**

**Department of Computer Systems Engineering**

**University of Engineering and Technology, Peshawar**

# Lab Objectives:

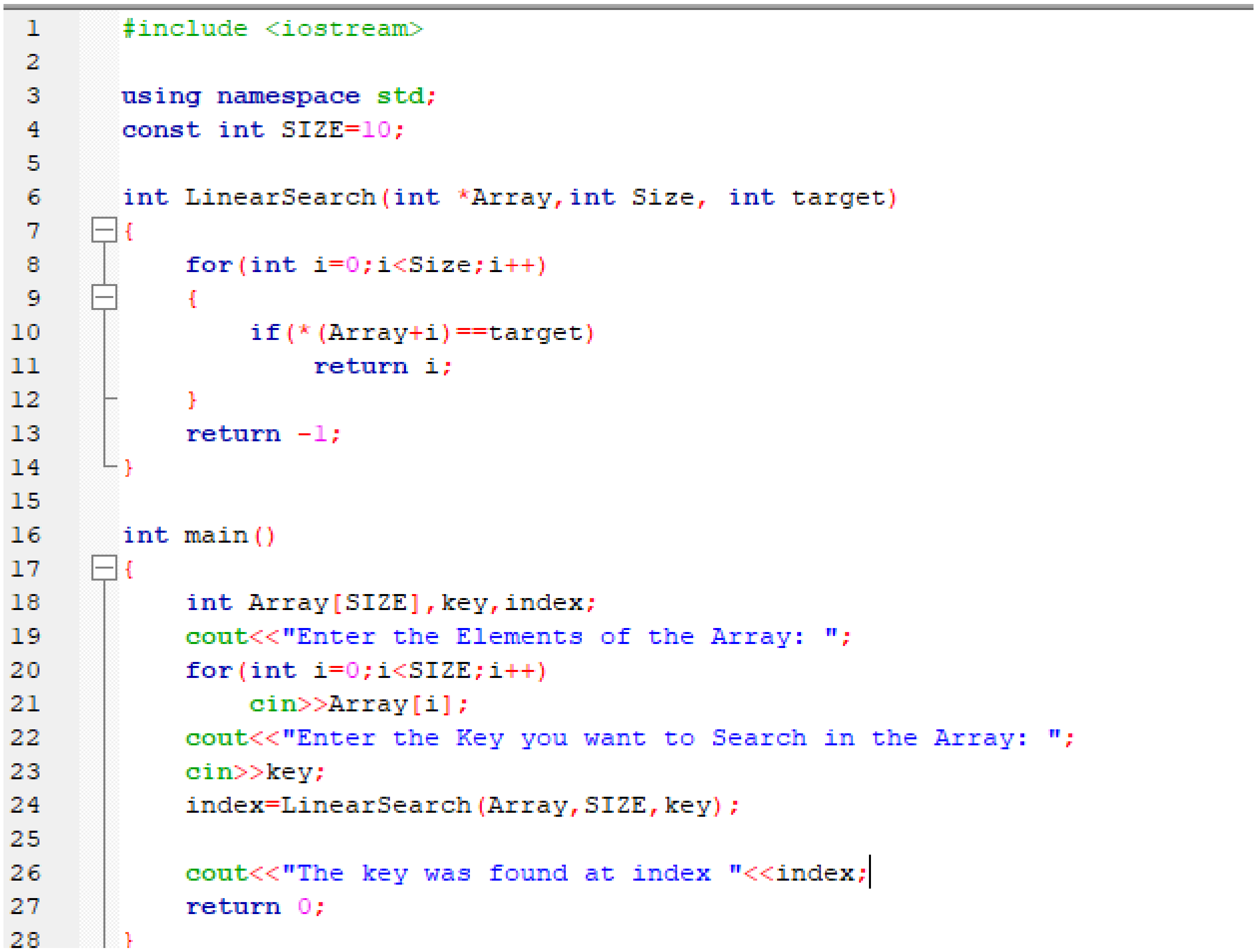
Objectives of this lab are as follows:

- Linear Search
- Binary Search

# Task # 1:

Implement Linear Search and analyze its worst, best and average case complexity.

# Code:



```
#include <iostream>

using namespace std;
const int SIZE=10;

int LinearSearch(int *Array,int Size, int target)
{
    for(int i=0;i<Size;i++)
    {
        if(*(Array+i)==target)
            return i;
    }
    return -1;
}

int main()
{
    int Array[SIZE],key,index;
    cout<<"Enter the Elements of the Array: ";
    for(int i=0;i<SIZE;i++)
        cin>>Array[i];
    cout<<"Enter the Key you want to Search in the Array: ";
    cin>>key;
    index=LinearSearch(Array,SIZE,key);

    cout<<"The key was found at index "<<index;
    return 0;
}
```

# Pseudo-Code/Explanation:

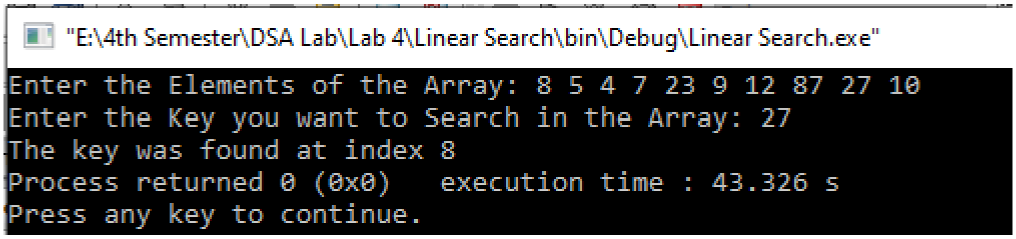
- Ask the user to enter size of the Array
- Ask the user to enter elements of the Array
- Ask the user to enter a key

- ➢ Call the linear search function and pass the array and its size to it.

  - Take a for loop from 0 to size
  - If Arr[i] is equal to key, Return the index value.

- ➢ Display the end result.

## Output:



"E:\4th Semester\DSA Lab\Lab 4\Linear Search\bin\Debug\Linear Search.exe"

```
Enter the Elements of the Array: 8 5 4 7 23 9 12 87 27 10
Enter the Key you want to Search in the Array: 27
The key was found at index 8
Process returned 0 (0x0)   execution time : 43.326 s
Press any key to continue.
```

## Complexity:

- ➢ **Best case:**

For linear search algorithm best case complexity is O [1] since in this algorithm the if the key is found at the first index of the array, the loop will only have to transverse one time.

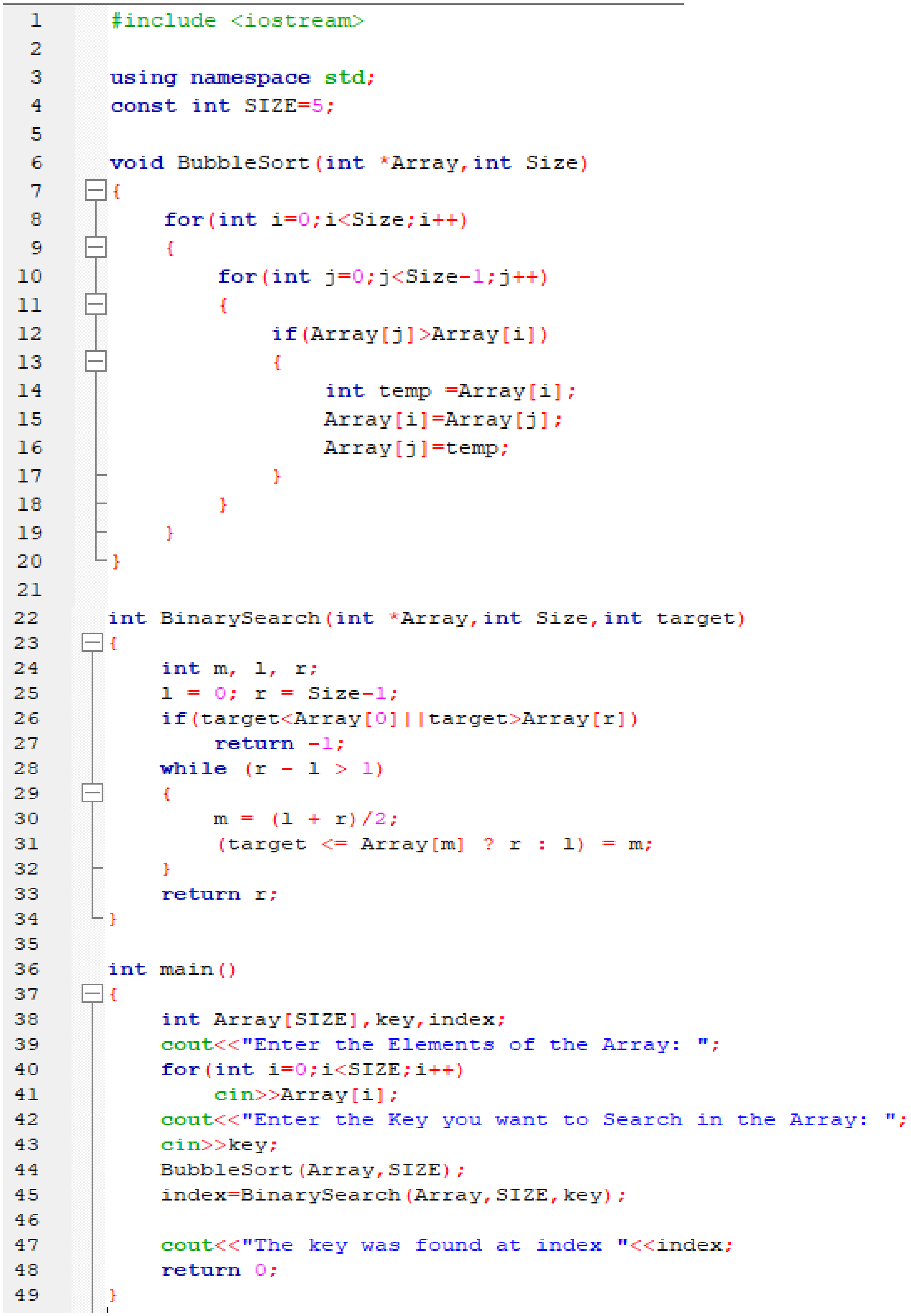
- ➢ **Worst case:**

For linear search algorithm worst case complexity is O [N] since in this algorithm if the key is at any position other than the first index, the loop will have to transverse n times.

## Task # 2:

Implement Binary Search and analyze its worst, best and average case complexity.

## Code:



```
#include <iostream>

using namespace std;
const int SIZE=5;

void BubbleSort(int *Array,int Size)
{
    for(int i=0;i<Size;i++)
    {
        for(int j=0;j<Size-1;j++)
        {
            if(Array[j]>Array[i])
            {
                int temp =Array[i];
                Array[i]=Array[j];
                Array[j]=temp;
            }
        }
    }
}

int BinarySearch(int *Array,int Size,int target)
{
    int m, l, r;
    l = 0; r = Size-1;
    if(target<Array[0]||target>Array[r])
        return -1;
    while (r - l > 1)
    {
        m = (l + r)/2;
        (target <= Array[m] ? r : l) = m;
    }
    return r;
}

int main()
{
    int Array[SIZE],key,index;
    cout<<"Enter the Elements of the Array: ";
    for(int i=0;i<SIZE;i++)
        cin>>Array[i];
    cout<<"Enter the Key you want to Search in the Array: ";
    cin>>key;
    BubbleSort(Array,SIZE);
    index=BinarySearch(Array,SIZE,key);

    cout<<"The key was found at index "<<index;
    return 0;
}
```

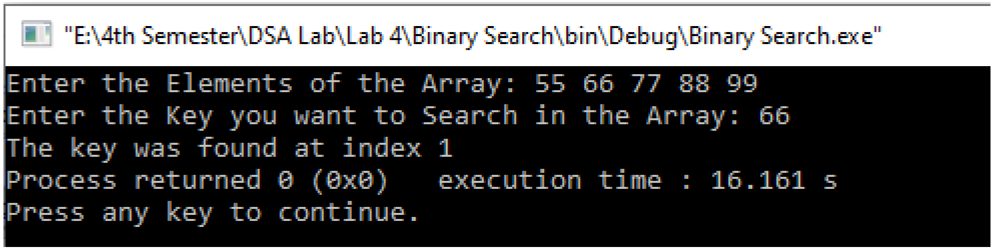
## Pseudo-Code/Explanation:

- Ask the user to enter size of the Array
- Ask the user to enter elements of the Array
- Ask the user to enter a key.
- Call the binary search function and pass the array and its size to it.

    - Calculate the midpoint of the array.
    - While l – r is greater than  1.
    - If Arr[m] is greater than or equal to the key, put r equal to m.
    - Else if Arr[m] is is less than key,  put l equal to m.
    - Return the r  value.

- Display the end result.

## Output:



"E:\4th Semester\DSA Lab\Lab 4\Binary Search\bin\Debug\Binary Search.exe"

```
Enter the Elements of the Array: 55 66 77 88 99
Enter the Key you want to Search in the Array: 66
The key was found at index 1
Process returned 0 (0x0)   execution time : 16.161 s
Press any key to continue.
```

## Complexity:

- **Best case:**

For binary search algorithm best case complexity is O [1] since in this algorithm the if the key is found at the middle index of the array, the loop will only have to transverse one time.

- **Worst case:**

For binary search algorithm worst case complexity is O [log2n] since in this algorithm the if the key is not found at the first middle index of the array, binary search begins comparing the middle element of the sub-array with the key. If the key is less than or greater than the middle element, the search continues in the lower or upper half of the array.