DHA Suffa University
Department of Computer Science
CS 2001L – Data Structures and Algorithms Lab
Spring 2021

Lab 02 – Search Algorithms

Objective:

To learn about

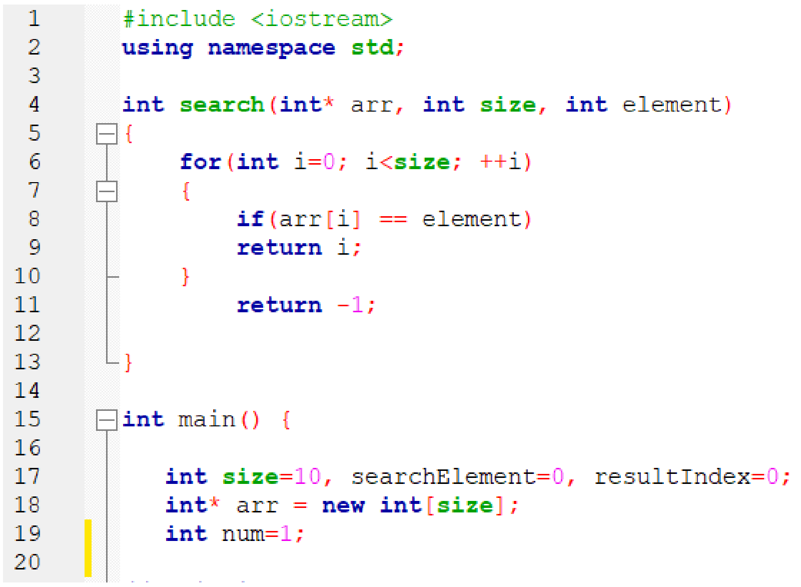
- Linear Search
- Binary Search

Search Algorithms

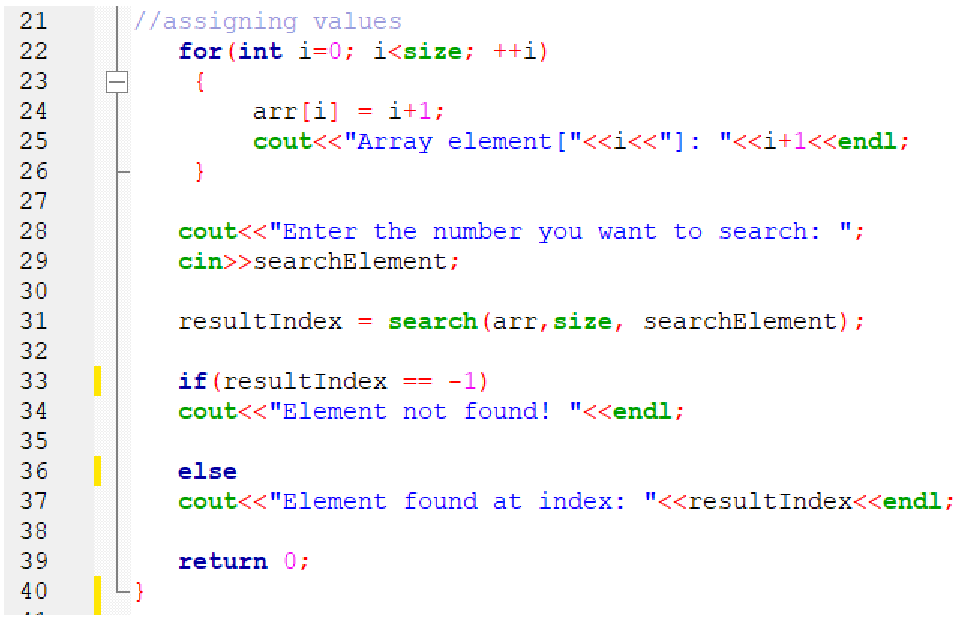
Linear Search

One of the simplest searching algorithms is linear search. To search an element in the array we compare the given element with each element starting with leftmost element in array, if element found at any index, return the index position otherwise return -1 to indicate that element was not found in Array.

Following code demonstrated the implementation of Linear Search Algorithm in C++.



```cpp
#include <iostream>
using namespace std;

int search(int* arr, int size, int element)
{
    for(int i=0; i<size; ++i)
    {
        if(arr[i] == element)
        return i;
    }
        return -1;

}

int main() {

    int size=10, searchElement=0, resultIndex=0;
    int* arr = new int[size];
    int num=1;

```

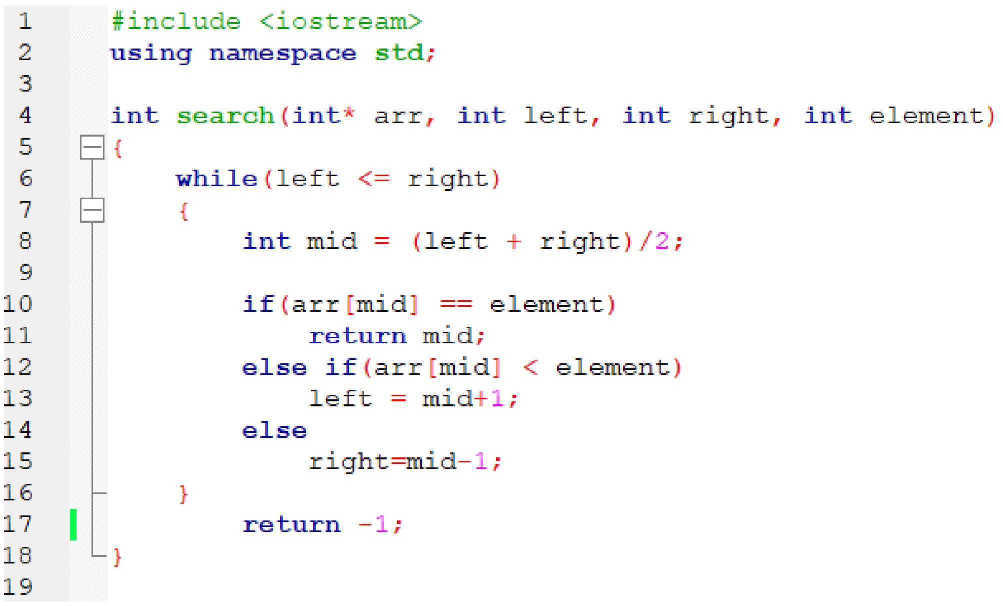


```
//assigning values
    for(int i=0; i<size; ++i)
     {
         arr[i] = i+1;
         cout<<"Array element["<<i<<"]: "<<i+1<<endl;
     }

    cout<<"Enter the number you want to search: ";
    cin>>searchElement;

    resultIndex = search(arr,size, searchElement);

    if(resultIndex == -1)
    cout<<"Element not found! "<<endl;

    else
    cout<<"Element found at index: "<<resultIndex<<endl;

    return 0;
}
```

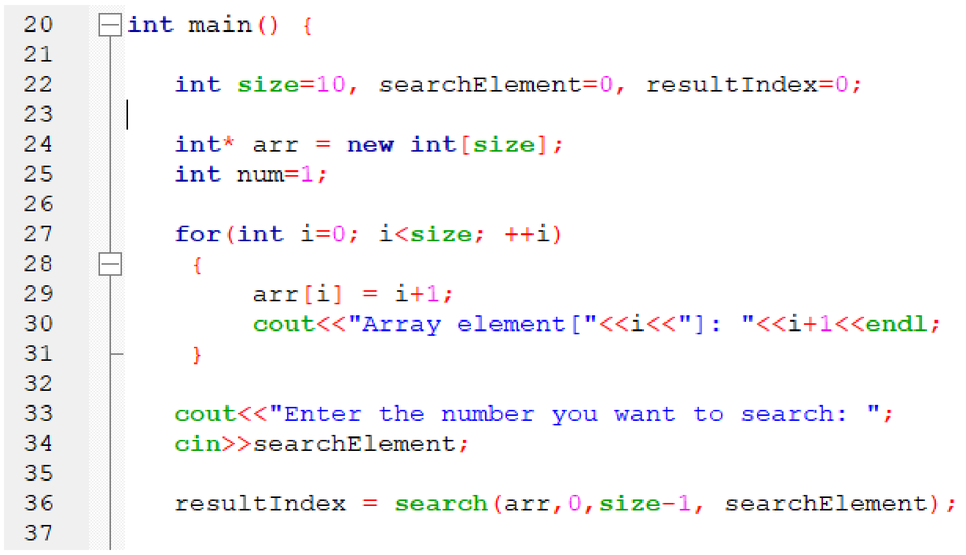
Binary Search

Binary search algorithms searches the element in a sorted array by comparing the given element with the middle element of the search space, if element is smaller than the middle element, the search space is limited to the left half of the middle element. In other case the search space is limited to right half of the middle element. This operation is carried out repeatedly until element is found or search space becomes empty.

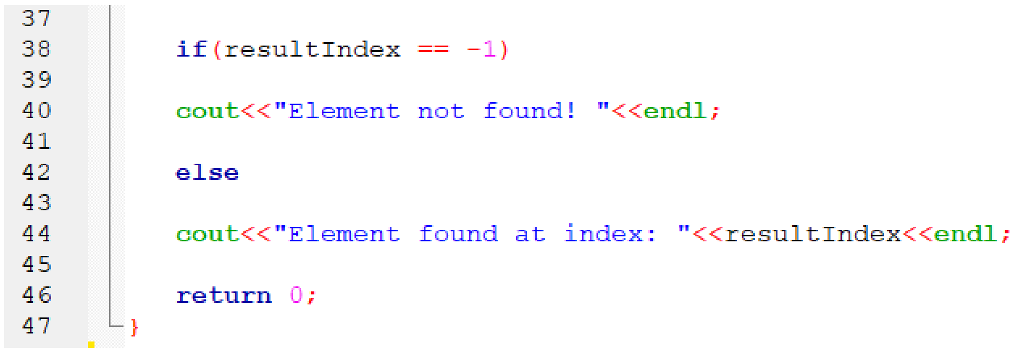
Following code demonstrated the implementation of Binary Search Algorithm in C++.



```
#include <iostream>
using namespace std;

int search(int* arr, int left, int right, int element)
{
    while(left <= right)
    {
        int mid = (left + right)/2;

        if(arr[mid] == element)
            return mid;
        else if(arr[mid] < element)
            left = mid+1;
        else
            right=mid-1;
    }
        return -1;
}

```



```cpp
int main() {

    int size=10, searchElement=0, resultIndex=0;

    int* arr = new int[size];
    int num=1;

    for(int i=0; i<size; ++i)
     {
         arr[i] = i+1;
         cout<<"Array element["<<i<<"]: "<<i+1<<endl;
     }

    cout<<"Enter the number you want to search: ";
    cin>>searchElement;

    resultIndex = search(arr,0,size-1, searchElement);

```



```cpp

    if(resultIndex == -1)

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

    else

    cout<<"Element found at index: "<<resultIndex<<endl;

    return 0;
}
```

LAB ASSIGNMENT

1. Create a program that returns indexes for all the occurrences of the search element.

2. Apply both the search algorithms on different sized arrays (n=10, 100 and 1000) and compare iterations for different cases by producing respective graphs programmatically.

SUBMISSION GUIDELINES

- Take a screenshot of each task (code and its output), labeled properly.
- Place all the screenshots and the code files (properly labeled) in a single folder labeled with Roll No and Lab No. e.g. **‘cs191xxx_Lab01’**.
- Submit the folder at LMS
- **-100%** policies for plagiarism.