School of Computing Science and Engineering

Course Code : BCS01T1003 Name: Computer Programming for Problem Solving-C

UNIT III

ARRAYS AND FUNCTIONS

ARRAYS (Lecture 4)

Name of the Faculty: Dr. Saumya Chaturvedi Program Name:B.Tech(CSE)

Recapitulations

- Types of Arrays
- One- dimensional Array
- Multi- dimensional Array
- Examples of multidimensional Arrays

Objective

- Searching algorithms
- Types of searching algorithm
- Searching in one dimensional array
- Searching in two dimensional array

Searching Algorithms

Searching Algorithms are designed to check for an element or retrieve an element from any data structure where it is stored.

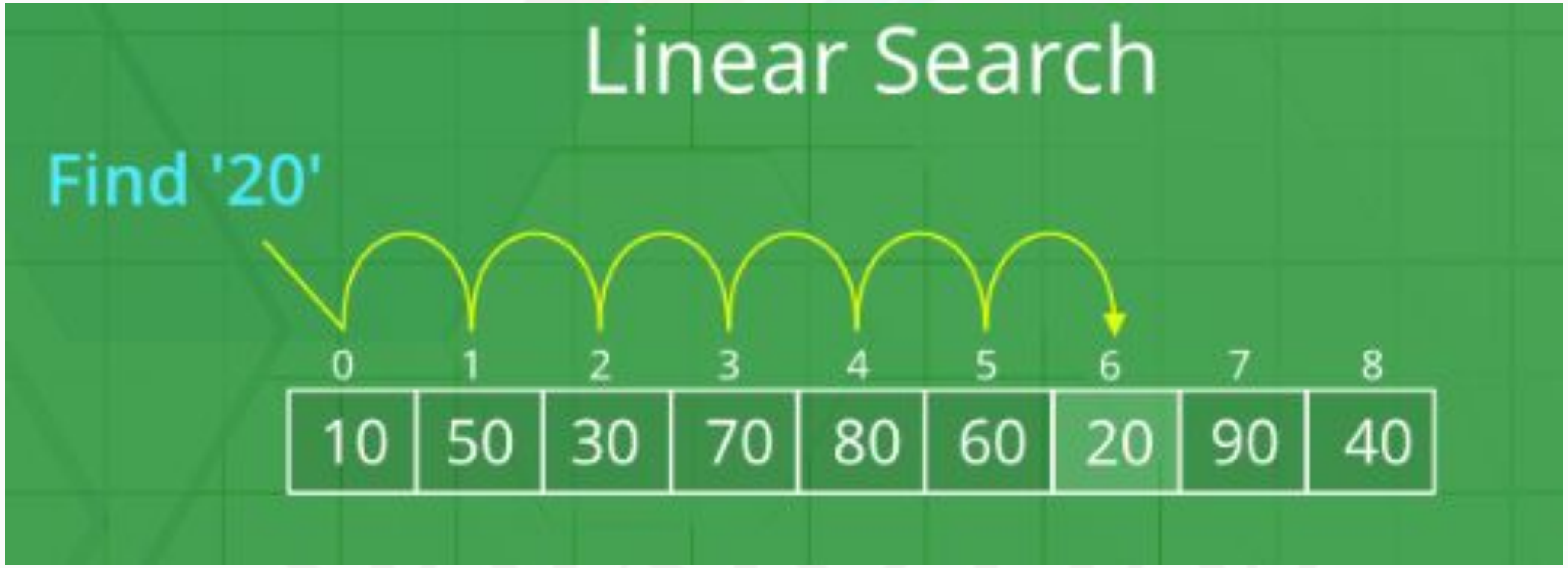
Based on the type of search operation, these algorithms are generally classified into two categories:

1. **Sequential Search:** In this, the list or array is traversed sequentially and every element is checked. **For example: Linear Search**

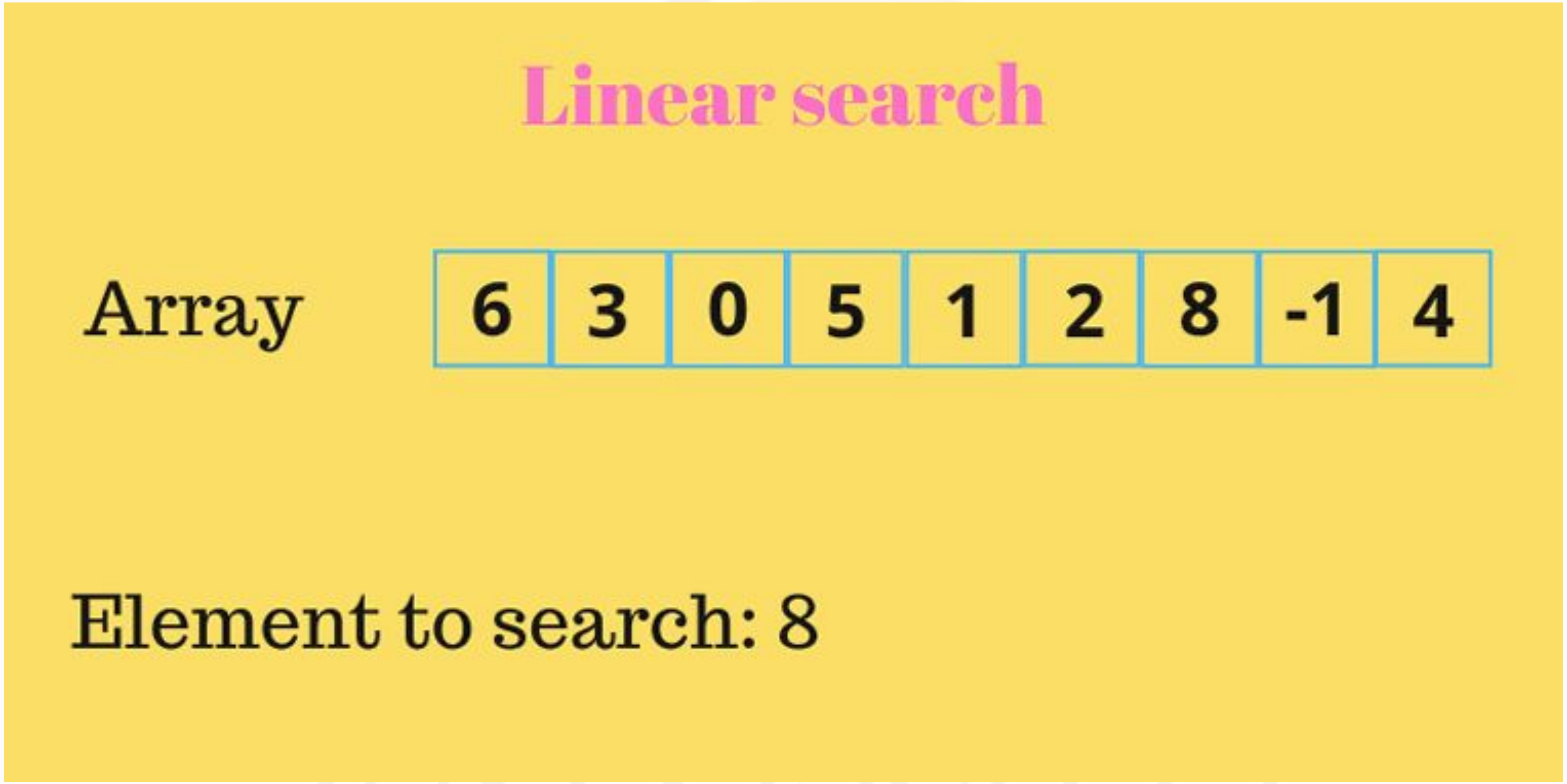
2. **Interval Search:** These algorithms are specifically designed for searching in sorted data-structures. These type of searching algorithms are much more efficient than Linear Search as they repeatedly target the centre of the search structure and divide the search space in half. **For Example: Binary Search**

Linear Search

Linear Search to find the element “20” in a given list of numbers



Linear Search



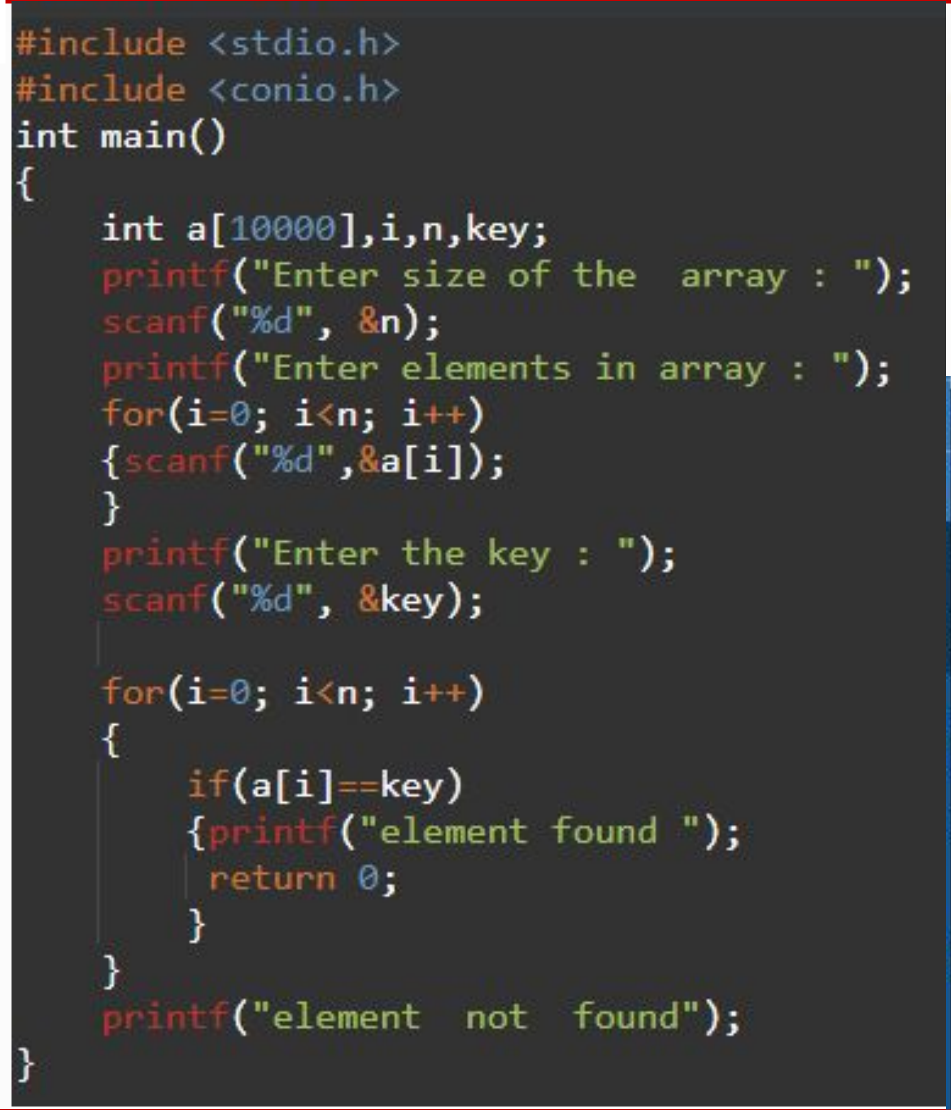
Linear Search Algorithm

Algorithm

- Start from the leftmost element of arr[] and one by one compare x with each element of arr[]
- If x matches with an element, return the index
- If x doesn’t match with any of elements, return -1

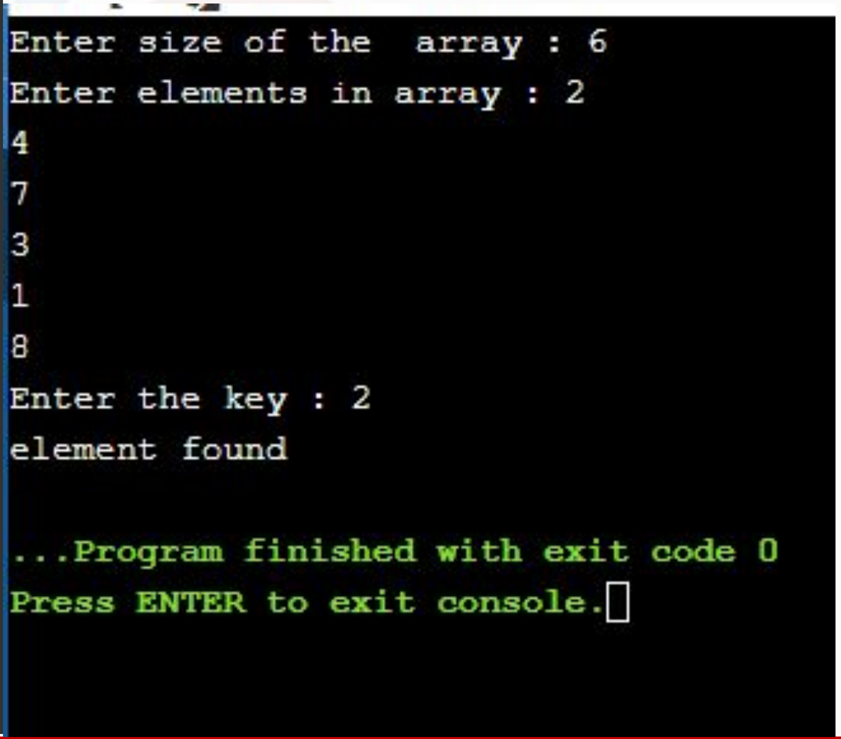
The time complexity of above algorithm is O(n)

Example: Linear Search



```c
#include <stdio.h>
#include <conio.h>
int main()
{
    int a[10000],i,n,key;
    printf("Enter size of the  array : ");
    scanf("%d", &n);
    printf("Enter elements in array : ");
    for(i=0; i<n; i++)
    {scanf("%d",&a[i]);
    }
    printf("Enter the key : ");
    scanf("%d", &key);

    for(i=0; i<n; i++)
    {
        if(a[i]==key)
        {printf("element found ");
         return 0;
        }
    }
    printf("element  not  found");
}
```

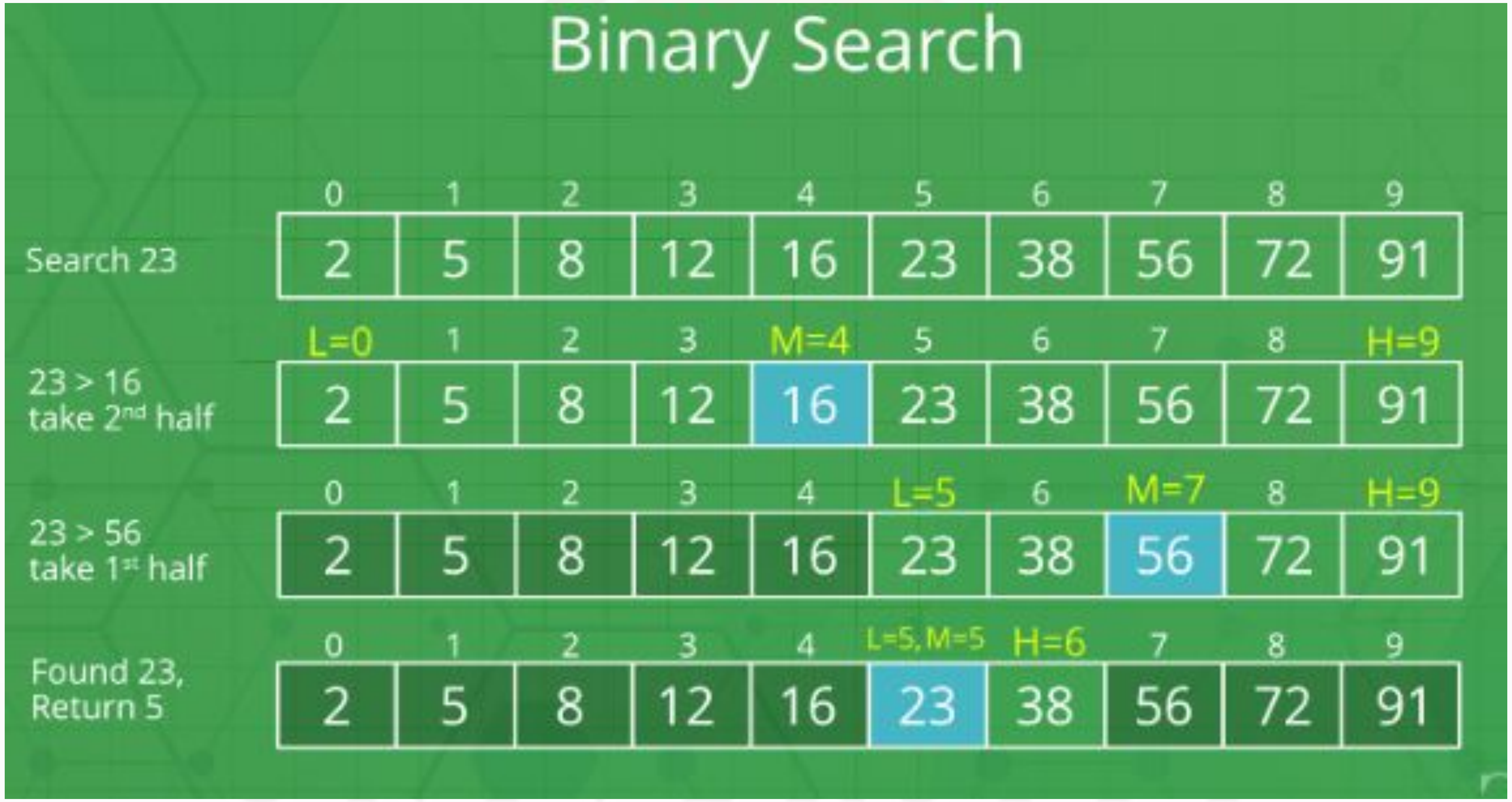
C Program to search for an element in single dimensional array using linear search



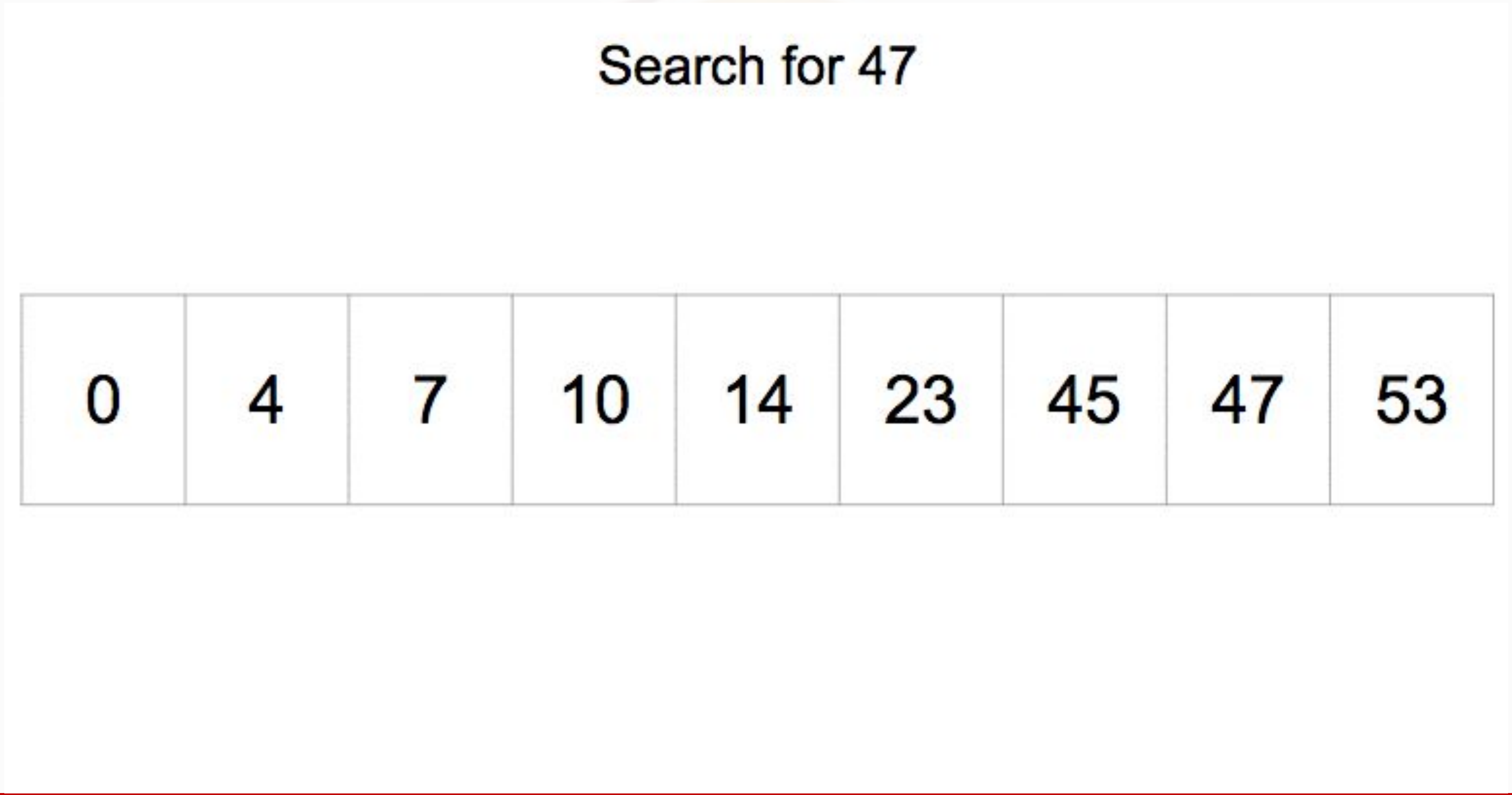
```
Enter size of the  array : 6
Enter elements in array : 2
4
7
3
1
8
Enter the key : 2
element found

...Program finished with exit code 0
Press ENTER to exit console.
```

Binary Search

Binary Search to find the element “23” in a given list of numbers



Binary Search



Search for 47

0	4	7	10	14	23	45	47	53

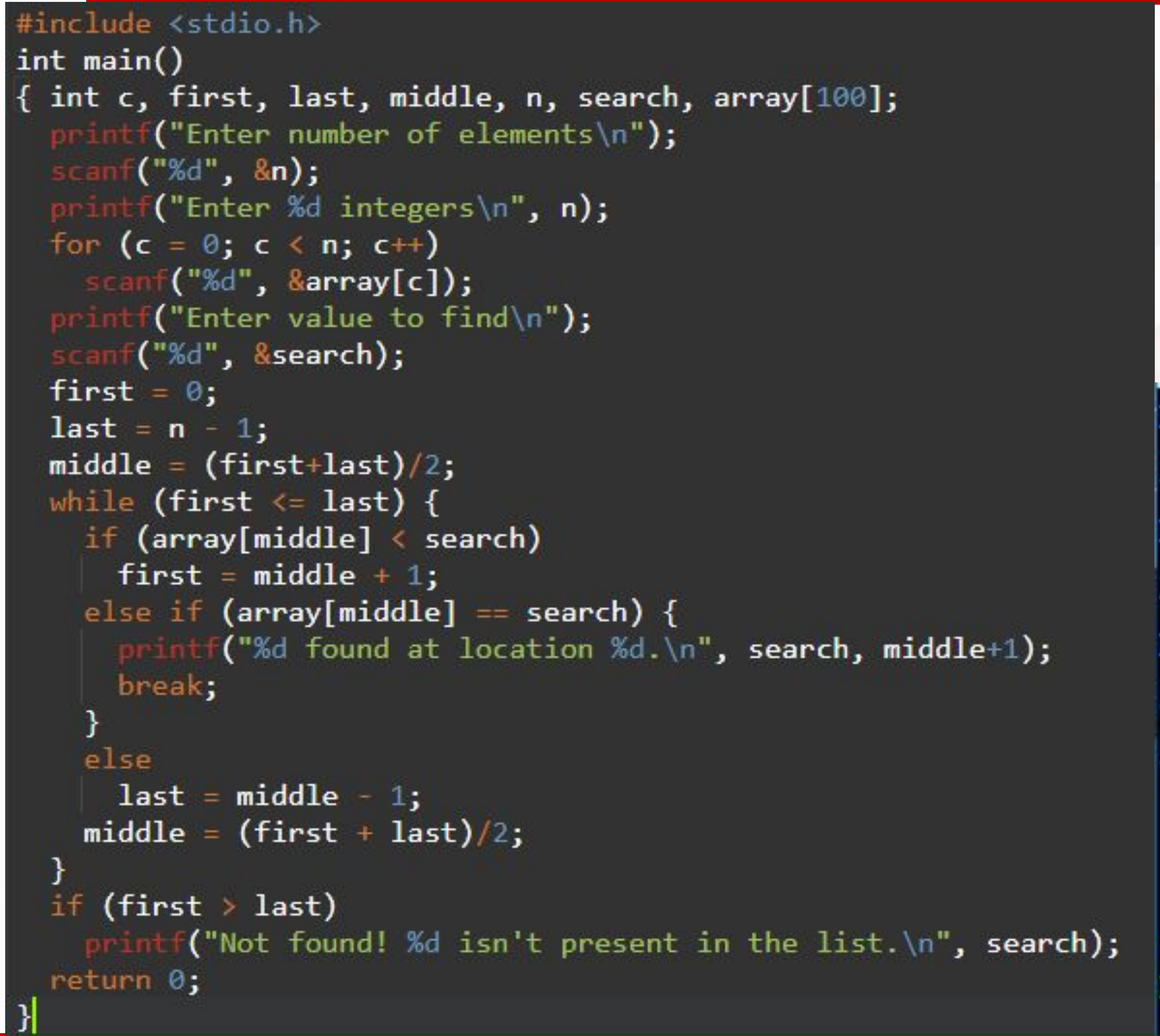
Binary Search Algorithm

Algorithm

- Compare x with the middle element.
- If x matches with middle element, we return the mid index.
- Else If x is greater than the mid element, then x can only lie in right half subarray after the mid element. So we recur for right half.
- Else (x is smaller) recur for the left half.

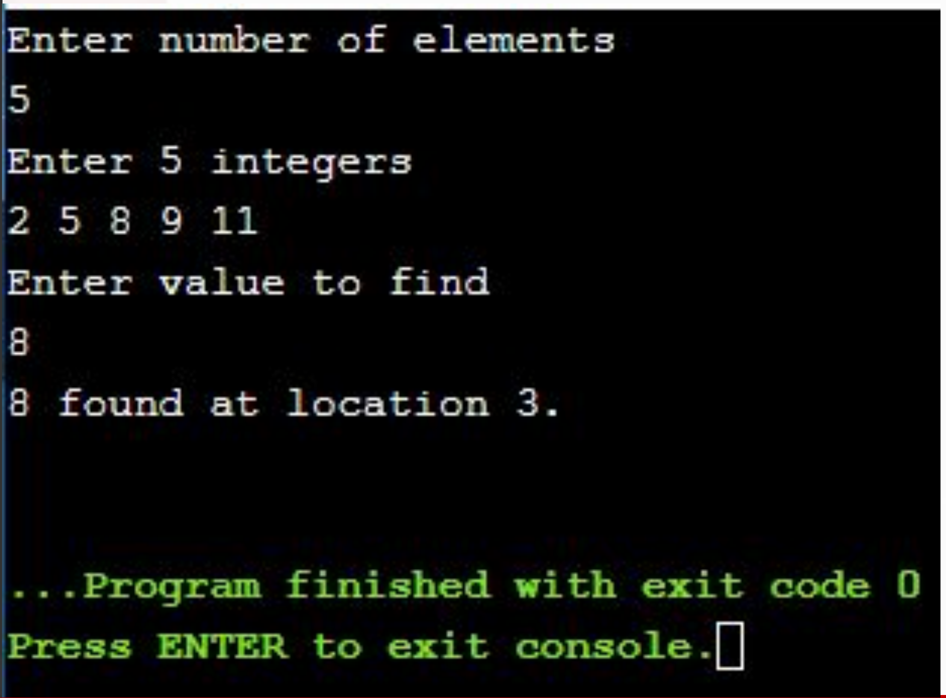
The idea of binary search is to use the information that the array is sorted and reduce the time complexity to O(Log n).

Example : Binary Search



```
#include <stdio.h>
int main()
{ int c, first, last, middle, n, search, array[100];
  printf("Enter number of elements\n");
  scanf("%d", &n);
  printf("Enter %d integers\n", n);
  for (c = 0; c < n; c++)
    scanf("%d", &array[c]);
  printf("Enter value to find\n");
  scanf("%d", &search);
  first = 0;
  last = n - 1;
  middle = (first+last)/2;
  while (first <= last) {
    if (array[middle] < search)
      first = middle + 1;
    else if (array[middle] == search) {
      printf("%d found at location %d.\n", search, middle+1);
      break;
    }
    else
      last = middle - 1;
    middle = (first + last)/2;
  }
  if (first > last)
    printf("Not found! %d isn't present in the list.\n", search);
  return 0;
}
```

C Program to search for an element in single dimensional array using Binary search



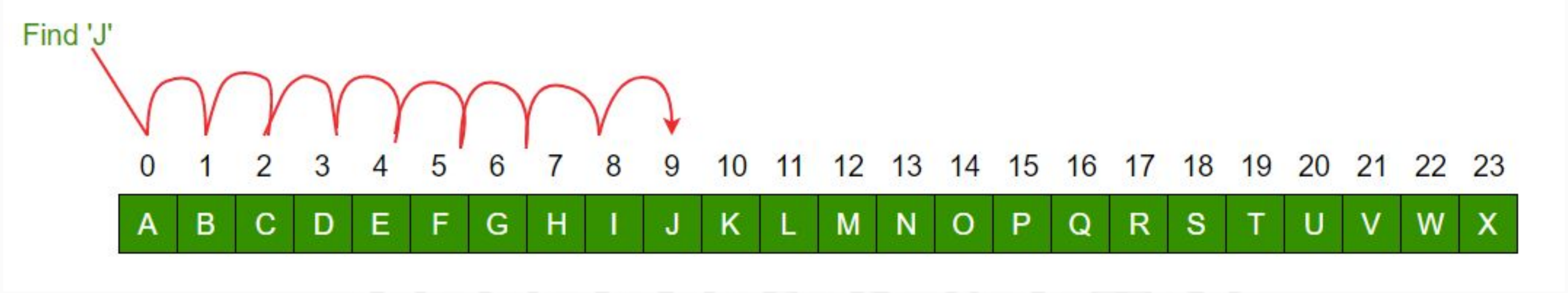
```
Enter number of elements
5
Enter 5 integers
2 5 8 9 11
Enter value to find
8
8 found at location 3.


...Program finished with exit code 0
Press ENTER to exit console.
```

Program Name: B.Tech (CSE)

Linear Search vs Binary Search

A linear search scans **one item at a time**, without jumping to any item .

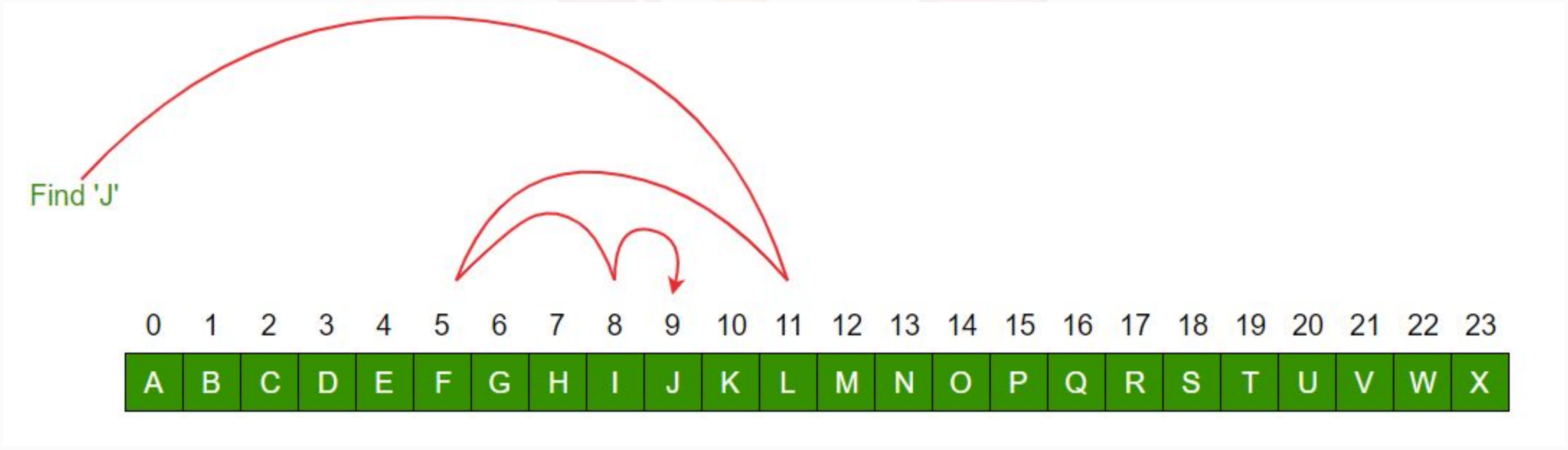
1. The worst case complexity is O(n), sometimes known an O(n) search
2. Time taken to search elements keep increasing as the number of elements are increased.



Linear Search vs Binary Search

A binary search however, cut down your **search to half** as soon as you find **middle** of a sorted list.

1. The middle element is looked to check if it is greater than or less than the value to be searched.
2. Accordingly, search is done to either half of the given list



Important Differences

- Input data needs to be sorted in Binary Search and not in Linear Search
- Linear search does the sequential access whereas Binary search access data randomly.
- Time complexity of linear search O(n) , Binary search has time complexity O(log n).
- Linear search performs equality comparisons and Binary search performs ordering comparisons

Summary

- ❖ A linear search scans one item at a time, without jumping to any item
- ❖ A binary search cut down search to half
- ❖ Input data needs to be sorted in Binary Search

References

- E. Balaguruswamy 7th Edition, Programming ANSI C, McGraw-Hill
- Brian W. Kernighan and Dennis M. Ritchie, The C programming Language, Prentice-Hall in 1988
- Byron Gottfried, Programming with C, Schaum's Outline

Contact Information

Dr. Saumya Chaturvedi

saumya.chaturvedi@galgotiasuniversity.edu.in

Thank You