SEARCHING ALGORITHM

Lecture 3

OUTLINES

- Introduction
- Sequential Search (Linear Search)
- Binary Search

INTRODUCTION

- Searching for data is one of the fundamental fields of computing. Often, the difference between a fast program and a slow one is the use of a good algorithm for the data set
- Has Numerous Engineering applications!

SEQUENTIAL SEARCH (LINEAR SEARCH)

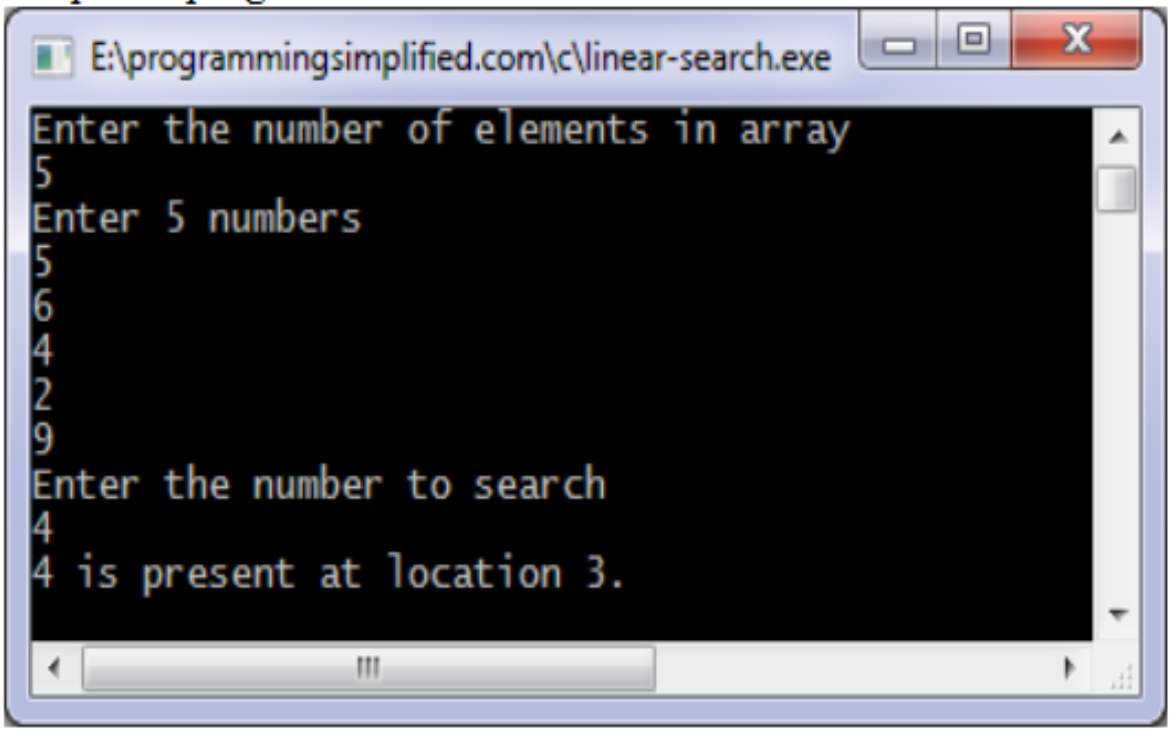
- The most obvious algorithm is the linear search which means to start at the beginning and walk to the end, testing for a match at each item
- The algorithm itself is simple. A familiar 0 - n-1 loop to walk over every item in the array, with a test to see if the current item in the list matches the search key

SEQUENTIAL SEARCH (LINEAR SEARCH)

```c
#include <stdio.h>

int main()
{
   int array[100], search, c, n;

   printf("Enter the number of elements in array\n");
   scanf("%d",&n);

   printf("Enter %d integer(s)\n", n);

   for (c = 0; c < n; c++)
      scanf("%d", &array[c]);

   printf("Enter the number to search\n");
   scanf("%d", &search);

   for (c = 0; c < n; c++)
   {
      if (array[c] == search)      /* if required element found */
      {
         printf("%d is present at location %d.\n", search, c+1);
         break;
      }
   }
   if (c == n)
      printf("%d is not present in array.\n", search);

   return 0;
}
```

SEQUENTIAL SEARCH (LINEAR SEARCH)

Output of program:

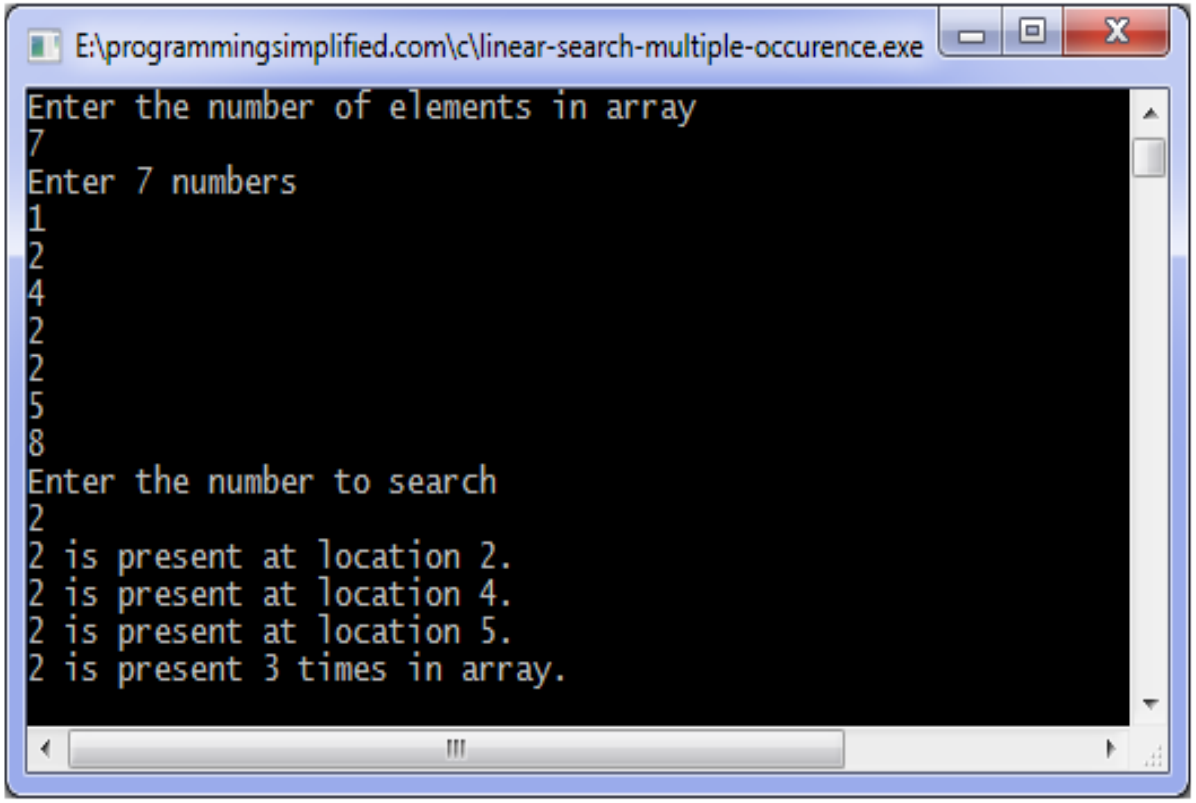


SEQUENTIAL SEARCH (LINEAR SEARCH- FOR MULTIPLE OCCURRENCES)

```c
#include <stdio.h>

int main()
{
   int array[100], search, c, n, count = 0;

   printf("Enter the number of elements in array\n");
   scanf("%d", &n);

   printf("Enter %d numbers\n", n);

   for ( c = 0 ; c < n ; c++ )
      scanf("%d", &array[c]);

   printf("Enter the number to search\n");
   scanf("%d", &search);

   for (c = 0; c < n; c++) {
      if (array[c] == search) {
         printf("%d is present at location %d.\n", search, c+1);
         count++;
      }
   }
   if (count == 0)
      printf("%d is not present in array.\n", search);
   else
      printf("%d is present %d times in array.\n", search, count);

   return 0;
}
```

SEQUENTIAL SEARCH (LINEAR SEARCH- FOR MULTIPLE OCCURRENCES)

Output of code:

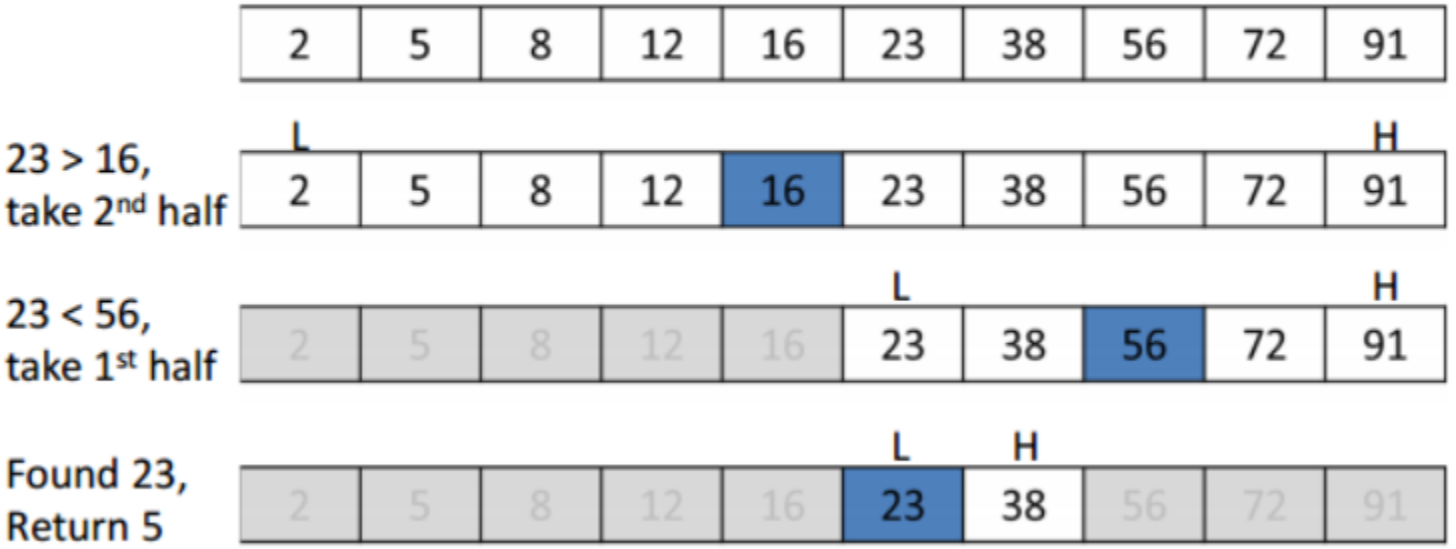


BINARY SEARCH

- **Binary Search:** Search a sorted array by repeatedly dividing the search interval in half. Begin with an interval covering the whole array. If the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half. Otherwise narrow it to the upper half. Repeatedly check until the value is found or the interval is empty.

BINARY SEARCH

If searching for 23 in the 10-element array:

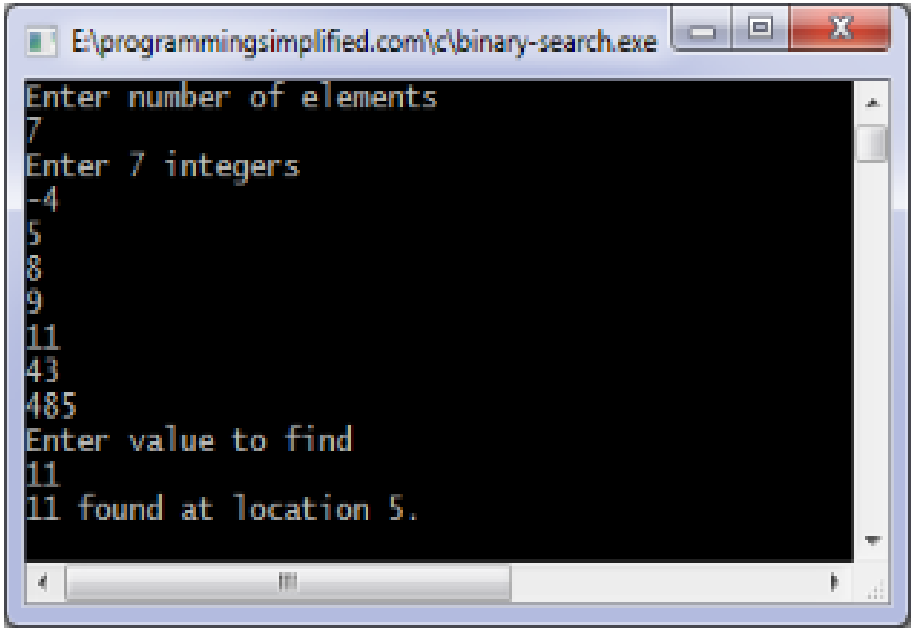


BINARY SEARCH

```c
#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 is not present in the list.\n", search);

   return 0;
}
```

BINARY SEARCH

Output of program:



BINARY SEARCH

Binary search is faster than linear search but the list should be sorted!