1. Write a program to print all the locations at which a particular element (taken as input) is found in a list and also print the total number of times it occurs in the list. The location starts from 1.  
   For example: if there are 4 elements in the array  
   5  
   6  
   5  
   7  
   If the element to search is 5 then the output will be  
   5 is present at location 1  
   5 is present at location 3  
   5 is present 2 times in the array.

#include <stdio.h>

int main()

{

int array[100], search, n, count = 0;

//"search" is the key element to search and 'n' is the total number of element of the array

// "count" is to store total number of elements

scanf("%d", &n); //Number of elements is taken from test case

int c;

for (c = 0; c < n; c++)

scanf("%d", &array[c]);

scanf("%d", &search); // The element to search is taken from test case

/\* Use the printf statements as below:

"%d is present at location %d.\n" for each locations

"%d is not present in the array.\n" if the element is not found in the list

"%d is present %d times in the array.\n"

\*/

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 the array.", search);

else

printf("%d is present %d times in the array.", search, count);

return 0;

}

1. Write a C program to search a given element from a 1D array and display the position at which it is found by using linear search function. The index location starts from 1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Private Test cases used for evaluation** | **Input** | **Expected Output** | **Actual Output** | **Status** |
| Test Case 1 | 4  45  65  85  25  95 | 95 is not present in the array. | 95 is not present in the array. | Passed |
| Test Case 2 | 5  6  9  5  4  7  6 | 6 is present at location 1. | 6 is present at location 1. | Passed |

#include <stdio.h>

int linear\_search(int[], int, int);

int main()

{

int array[100], search, c, n, position;

/\* search - element to search, c - counter, n - number of elements in array,

position - The position in which the element is first found in the list. \*/

scanf("%d", &n); // Number of elements in the array is read from the test case data

for (c = 0; c < n; c++)

scanf("%d", &array[c]); //Elements of array is read from the test data

scanf("%d", &search); //Element to search is read from the test case data

/\* Use the following in the printf statement to print the output

printf("%d is not present in the array.\n", search);

printf("%d is present at location %d.\n", search, position+1); //As array[0] has the position 1

\*/

position = linear\_search(array, n, search);

if (position == -1)

printf("%d is not present in the array.", search);

else

printf("%d is present at location %d.", search, position+1);

return 0;

}

int linear\_search(int a[], int n, int find) {

int c;

for (c = 0 ;c < n ; c++ )

{

if (a[c] == find)

return c;

}

return -1;

}

1. Write a C program to search a given number from a sorted 1D array and display the position at which it is found using binary search algorithm. The index location starts from 1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Private Test cases used for evaluation** | **Input** | **Expected Output** | **Actual Output** | **Status** |
| Test Case 1 | 6  1  2  3  4  5  6  2 | 2 found at location 2. | 2 found at location 2. | Passed |
| Test Case 2 | 7  40  50  60  70  80  90  100  100 | 100 found at location 7. | 100 found at location 7. | Passed |

#include <stdio.h>

int main()

{

int c, n, search,

array[100];

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

for (c = 0; c < n; c++)

scanf("%d",&array[c]);

scanf("%d", &search); //The element to search is read from test case.

/\* Use the printf statements as below:

printf("%d found at location %d.\n", search, variable\_name);

printf("Not found! %d isn't present in the list.\n", search);

\*/

int first, last, middle;

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.", 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.", search);

return 0;

}

1. Write a C program to reverse an array by swapping the elements and without using any new array.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Private Test cases used for evaluation** | **Input** | **Expected Output** | **Actual Output** | **Status** |
| Test Case 1 | 7  8  9  10  6  4  7  11 | Reversed array elements are:\n  11\n  7\n  4\n  6\n  10\n  9\n  8 | Reversed array elements are:\n  11\n  7\n  4\n  6\n  10\n  9\n  8\n | Passed |

#include <stdio.h>

int main() {

int array[100], n, c;

scanf("%d", &n); // n is number of elements in the array.

for (c = 0; c < n; c++) {

scanf("%d", &array[c]);

}

int temp, end;

end = n - 1;

for (c = 0; c < n/2; c++) {

temp = array[c];

array[c] = array[end];

array[end] = temp;

end--;

}

printf("Reversed array elements are:\n");

for (c = 0; c < n; c++) {

printf("%d\n", array[c]);

}

return 0;

}