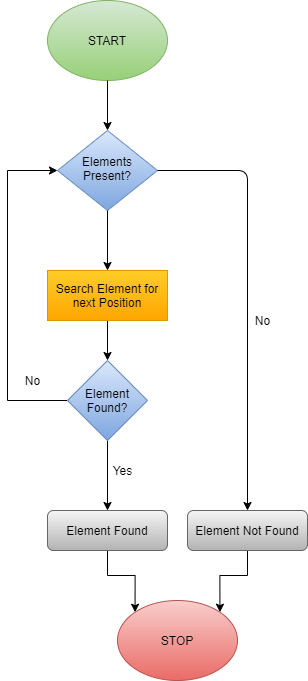
DSU Practicals

1. Draw a flowchart and algorithm and to develop a program for searching an given key elements using the linear search from list of n numbers.

#include <stdio.h>

int main()

{ int n, key, i, found = 0;

printf("Enter the number of elements in the list: "); 

scanf("%d", &n);

int arr[n];

printf("Enter %d elements:\n", n);

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

scanf("%d", &arr[i]);

}

printf("Enter the key element to search: ");

scanf("%d", &key);

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

if(arr[i] == key) {

found = 1;

printf("Element %d found at position %d.\n", key, i + 1);

break;

}

}

if(!found) {

printf("Element %d not found in the list.\n", key)

}

return 0;

}

**Linear Search Algorithm**

1. **Input**: A list of n elements and a key to search for.
2. **Initialize**:
3. Set found = 0 (meaning key not found initially).
4. **For** each element arr[i] from 0 to n-1:
5. If arr[i] == key:
   1. Print "Element found at position i + 1".
   2. Set found = 1.
   3. Exit the loop.
6. **If** found == 0:
7. Print "Element not found".
8. **End**.
9. Draw a flowchart and algorithm and to develop a program for searching an given key elements using the binary search from list of n numbers.

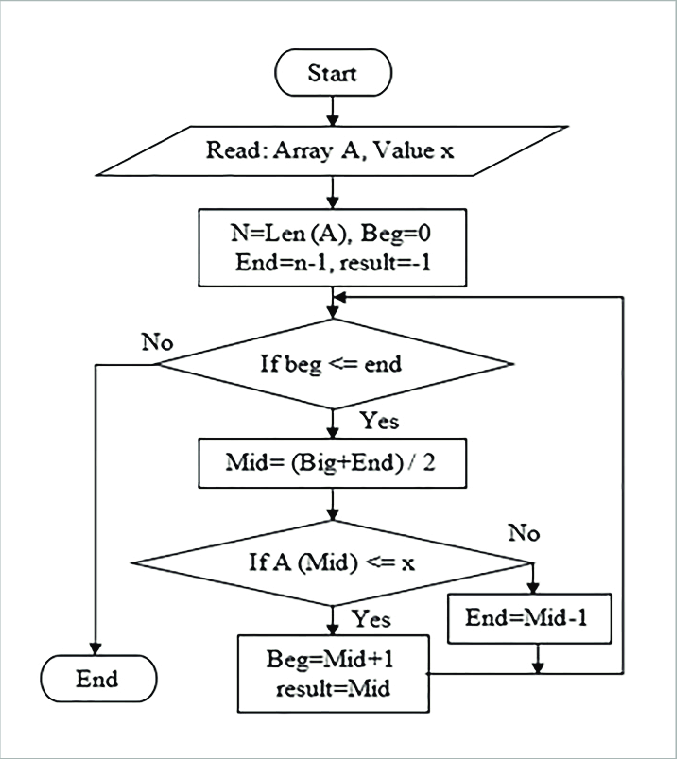
#include <stdio.h>

int binarySearch(int arr[], int n, int key) {

int low = 0, high = n - 1, mid;

while (low <= high) {

mid = (low + high) / 2; // Calculate middle index

 if (arr[mid] == key) {

return mid;

}

else if (arr[mid] < key) {

low = mid + 1;

}

else {

high = mid - 1;

}

}

return -1;

}

int main() {

int n, key, result;

printf("Enter the number of elements in the list: ");

scanf("%d", &n);

int arr[n];

printf("Enter %d sorted elements:\n", n);

for (int i = 0; i < n; i++) {

scanf("%d", &arr[i]);

}

printf("Enter the key element to search: ");

scanf("%d", &key);

result = binarySearch(arr, n, key);

if (result != -1) {

printf("Element %d found at position %d (1-based index).\n", key, result + 1);

} else {

printf("Element %d not found in the list.\n", key);

}

return 0;

}

**1.Input**: A sorted array of n numbers and a key to search for.

* + 1. Set low to the first index (0) and high to the last index (n-1).

1. **Repeat** the following steps while low <= high:
2. Find the middle index: mid = (low + high) / 2.
3. Compare the key with arr[mid]:
   1. If arr[mid] == key, print the position and exit (key found).
   2. If arr[mid] < key, update low = mid + 1 (search in the right half).
   3. If arr[mid] > key, update high = mid - 1 (search in the left half).
4. If low > high, print that the key is not found (key not in the array).

6. **End** the program.