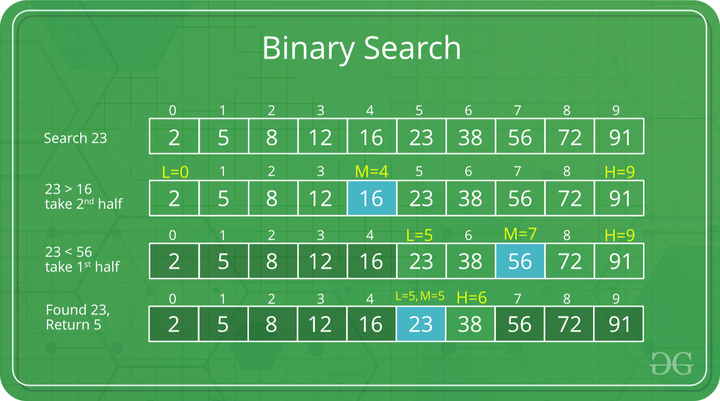
//COS10007 – Developing Technical Software//  
// Week 8 – Lab 8 //  
// Student ID – 103234103 – Son Nguyen //

1/

**Linear search** – Linear search in C is a sequential search algorithm that starts at one end of the array and goes through each element on the list until the desired element is found, otherwise it will continue until it reaches the end of the array/data set.

**Example/Diagram -**  
 **Binary Search** – Binary search is defined as a searching algorithm that used sorted array by repeatedly dividing the search interval in half. Binary search can be implemented only on a sorted list of items. If elements are nor sorted already, it would require the array to be sorted first  
  
  
  
  
  
2/  
a/

// Part A

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

// Function for Linear Search

int linear\_search(int arr[], int size, int key);

int main() {

// Array 10 intergers

int arr[10];

srand(time(NULL)); // Adding this so there is no duplicate result

for (int i = 0; i < 10; i++) {

arr[i] = rand() % 101; // Generate a random integer between 0 and 100

}

// Print the initial array

printf("Initial array: ");

for (int i = 0; i < 10; i++) {

printf("%d ", arr[i]);

}

printf("\n");

// Search key

int key;

printf("Enter the search key: ");

scanf("%d", &key);

// Call function for linear search

int location = linear\_search(arr, 10, key);

// Print the search result

if (location == -1) {

printf("%d not found in the array\n", key);

} else {

printf("%d found at location %d in the array\n", key, location);

}

return 0;

}

//Part B

// Function definition for linear search

int linear\_search(int arr[], int size, int key) {

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

if (arr[i] == key) {

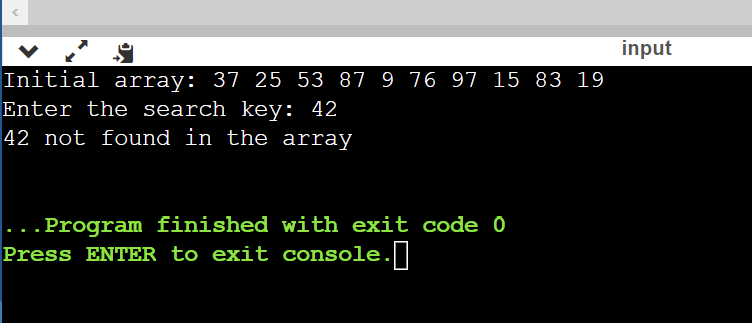
return i; // Return the number if found

}

}

return -1; // Return -1 if the element was not found

}

**Example Output** -  
  
  
  
  
  
  
  
**Not Found in Array**   
  
**Found in Array**   
Text

Description automatically generated  
  
B/  
//Part A

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

// Function for Binary Search

int binary\_search(int arr[], int start, int end, int key);

int main() {

// Initialize the array with random integers

int arr[10];

srand(time(NULL)); // Adding this so there is no duplicate result

for (int i = 0; i < 10; i++) {

arr[i] = rand() % 101; // Generate a random integer between 0 and 100

}

// Sorting array using loop

for (int i = 0; i < 9; i++) {

for (int j = i+1; j < 10; j++) {

if (arr[j] < arr[i]) {

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

}

// Print the sorted array using a for loop

printf("Initial sorted array: ");

for (int i = 0; i < 10; i++) {

printf("%d ", arr[i]);

}

printf("\n");

// Input from user for search key

int key;

printf("Enter the search key: ");

scanf("%d", &key);

// Call function for binary search

int location = binary\_search(arr, 0, 9, key);

// Print the search result

if (location == -1) {

printf("%d not found in the array\n", key);

} else {

printf("%d found at location %d in the array\n", key, location);

}

return 0;

}

// Part B

// Function definition for binary search

int binary\_search(int arr[], int begin, int end, int key) {

while (begin <= end) {

// Calculate the middle of the sub-array

int mid = (begin + end) / 2;

// Print the current sub-array being searched

printf("Sub-array: ");

for (int i = start; i <= end; i++) {

printf("%d ", arr[i]);

}

printf("\n");

// If the middle element is equal to the key, return the middle index

if (arr[mid] == key) {

return mid;

// If the middle element is less than the key, search the right half of the sub-array

} else if (arr[mid] < key) {

start = mid + 1;

// If the middle element is greater than the key, search the left half of the sub-array

} else {

end = mid - 1;

}

}

// If the key is not found in the sub-array, return -1

return -1;

}

**Example Output -  
Not found in array**   
Text

Description automatically generated  
**Found in array**   
Text

Description automatically generated