W8\_1a

Linear Search explanation:

Overall:

* The linear search algorithm is straightforward: it compares each element of the array to the search key.
* Because the array is not in any order, the value could be found in either the first or last element.
* On average, the program will have to compare the search key with half of the array's elements.
* It can also be used for linked lists.

Logic:

1. A linear search will loop through each item in the list (array or linked list), with a test to see if the current item matches the search key.
2. The loop may end in one of two ways:

* The loop condition fails if it reaches the end of the list.
* The loop is terminated early with a break statement if the current item in the list matches the key.

Example:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 5 | 4 | 2 | 1 | 7 | 6 |

We have an array of number like this:

For example, we want to search for the number 2, which is in the array [2].

The search key will be (key=2)

The key will be compared to every value of the array from the beginning to the end until it found the equal value.

1st comparison: 2 != 5

* Come to 2nd comparison : 2 != 4
* Come to 3rd comparison: 2 = 2

The number 2 has been found in the array [2], so this is the end of the linear search.

Binary Search explanation:

Overall:

* The binary search algorithm locates a specified input value, referred to as a search key, within an array sorted by key value.
* After each comparison, it removes one-half of the elements in a sorted array from consideration.

Logic:

1. Locate the middle element of the array.
2. Compare the middle element to the search key.
3. If they are equal, return the array subscript of the middle element.
4. If they are not equal, determine which half of the array to search next.
   1. If the search key is less than the middle element of the array, search the first half of the array.
   2. If the search key is greater than the middle element of the array, search the second half of the array.
5. Repeat steps 1-6 on the selected subarray until the search key is found or the subarray consists of only one element that is not equal to the search key.
6. If the search key is not found, return an indication that the key is not in the array.

Example:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 5 | 4 | 2 | 1 | 7 | 6 |

To use binary search, we need to sort this array into correct order.

To do that we can use any sorting method that we have learned in the previous week.

In the case that I did, I use selection sort.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 4 | 5 | 6 | 7 |

The low element will be array [0] and the high element will be array [6].

We need to choose the middle element: middle = (0+6)/2 = 3 --> middle element will be 4.

For example: we want to find the value 5.

1st comparison: compare the middle element to number 5.

4 < 5 --> the first half of the array (1,2,4) won’t need to be checked.

We will continue to check with the second half (5,6,7)

|  |  |  |
| --- | --- | --- |
| 5 | 6 | 7 |

The middle element will be array [1] in this sub-array.

2nd comparison: 6>5 🡪 Come to the array [0] to compare.

3rd comparison: 5 =5 🡪 The search key has been found

W8\_2a.c

/\*

Unit Code: COS10007

Unit Name: Developing Techinical Software

Student ID: 103488515

Name: Hai Nam Ngo

Date Created: 05/01/2023

Date Modified: 05/01/2023

Problem: Week 8 Question 2a

Problem Description:

Linear Search

a. The function should receive an integer array, a search key and the size of the

array as arguments.

b. If the search key is found, return the location in the array where the search key is

found; otherwise, return -1.

\*/

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#define SIZE 10

/\*The function receive an integer array, a search key and the size of the

array as arguments\*/

int linearSearch(const int array[], int key,int array\_size)

{

int i;

/\*/\*If the search key is found, return the location in the array where the search key is

found; otherwise, return -1\*/

for (i=0; i<array\_size; i++)

{

if (array[i]==key)

{

printf("Found value in element: %d",i);

return i;

}

}

printf("Value not found.\n");

return -1;

}

int main()

{

int array[SIZE],i,key;

srand(time(NULL));

for (i=0; i<SIZE; i++)

array[i]=rand()%100;

//print out the array

printf("The array: \n");

for (i=0; i<SIZE; i++)

{

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

}

printf("\nEnter integer search key: ");

scanf("%d",&key);

linearSearch(array,key,SIZE);

return 0;

}

OUTPUT FOR W8\_2a.c

Text

Description automatically generated

W8\_2b.c

/\*

Unit Code: COS10007

Unit Name: Developing Techinical Software

Student ID: 103488515

Name: Hai Nam Ngo

Date Created: 05/01/2023

Date Modified: 05/01/2023

Problem: Week 8 Question 2b

Problem Description:

Binary Search

a. The function should receive an integer array, a search key, the starting subscript

and ending subscript as arguments.

b. If the search key is found, return the location in the array where the search key is

found; otherwise, return -1.

\*/

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#define SIZE 10

int binarySearch(const int array[], int key, int low, int high)

{

int middle,i;

while (low <= high)

{

middle = (low + high)/2;

if (key == array[middle])

{

printf("\nFound value in element: %d",middle);

return middle;

}

else if (key < array[middle])

{

high = middle - 1;

}

else

{

low = middle + 1;

}

//print out sub-array

printf("\nThe sub-array: ");

for (i = low; i <= high; i++)

{

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

}

}

printf("\nValue not found.\n");

return -1;

}

//using selection sort to sort the array

void SelectionSort(int array[], int array\_size)

{

int i;

for (i = 0; i < array\_size-1; ++i)

{

int j, min;

int temp;

min = i;

for (j = i+1; j < array\_size; ++j)

{

if (array[j]< array[min])

min = j;

}

temp = array[i];

array[i] = array[min];

array[min] = temp;

}

}

int main()

{

int array[SIZE],i,key;

srand(time(NULL));

for (i=0; i<SIZE; i++)

array[i]=rand()%100;

//print out the array in the order from the smallest to largest

SelectionSort(array,SIZE);

printf("The array: \n");

for (i=0; i<SIZE; i++)

{

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

}

printf("\nEnter integer search key: ");

scanf("%d",&key);

int low = 0;

int high = SIZE - 1;

binarySearch(array, key, low, high);

return 0;

}

OUTPUT FOR W8\_2b.c

