City University

Department of CSE

Course Code : CSE 3116

Course Title : Algorithms Laboratory

Submitted To

Debobrata Chakraborty

Lecturer

Department of CSE

City University,Khagan,Savar,Dhaka

Submitted By

Name : **Fariha Ahmed Riya**

ID : **0272320005101296**

Batch & Section : **63 C**

Department : **CSE**

City University,Khagan,Savar,Dhaka

Linear Search

The code iterates through the array and checks for a match with the key. It prints the index if found and breaks. This is a standard linear search implementation.

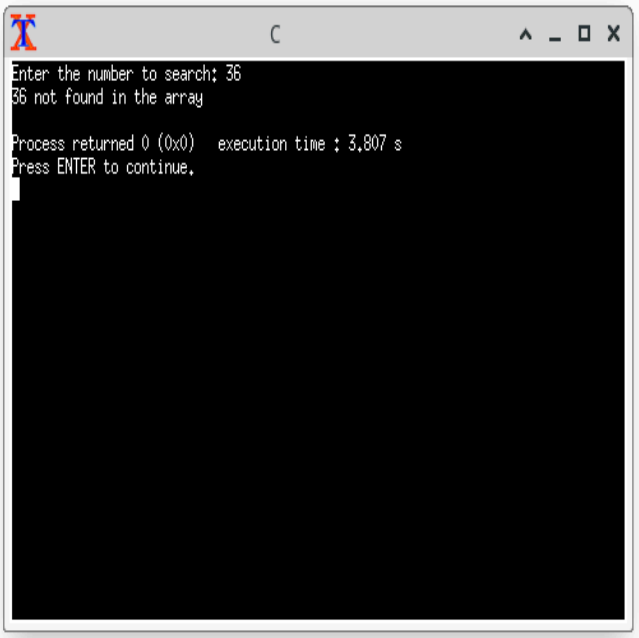
The Source Code :

```
#include <stdio.h>
int main() {
    int arr[] = {02, 72, 32, 00, 05, 10, 12, 96};
    int n = sizeof(arr) / sizeof(arr[0]);
    int key, i;
    printf("Enter the number to search: ");
    scanf("%d", &key);
    for (i = 0; i < n; i++) {
        if (arr[i] == key) {
            printf("Found %d at index %d\n", key, i);
            return 0;
        }
    }
    printf("%d not found in the array\n", key);
    return 0;
}
```

Screenshot of the Linear Search Code:

lab.c

```c
#include <stdio.h>
int main() {
    int arr[] = {02, 72, 32, 00, 05, 10, 12, 96};
    int n = sizeof(arr) / sizeof(arr[0]);
    int key, i;
    printf("Enter the number to search: ");
    scanf("%d", &key);
    for (i = 0; i < n; i++) {
        if (arr[i] == key) {
            printf("Found %d at index %d\n", key, i);
            return 0;
        }
    }
    printf("%d not found in the array\n", key);
    return 0;
}
```



Binary Search

This assumes the array is sorted. The provided code uses a sorted version ([00, 02, 05, 10, 12, 32, 72, 96]). It correctly implements the binary search algorithm with a while loop to find the target value. Note that for the original unsorted array, you need to sort it first or use Linear Search instead.

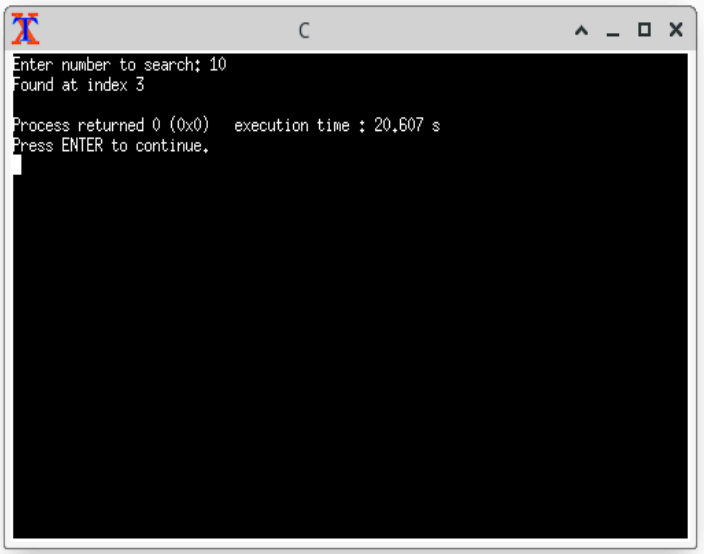
Source Code :

```
#include <stdio.h>

int main() {

    int arr[] = {00, 02, 05, 10, 12, 32, 72, 96}; // Assuming sorted array

    int n = sizeof(arr) / sizeof(arr[0]);

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

    printf("Enter number to search: ");

    scanf("%d", &x);

    while (low <= high) {

        mid = (low + high) / 2;

        if (arr[mid] == x) {

            printf("Found at index %d\n", mid);

            return 0;

        }

        else if (arr[mid] < x)

            low = mid + 1;

        else

            high = mid - 1;

    }

    printf("Not found\n");

    return 0;

}
```

Screenshot of the Binary Search Code:

*lab.c

```c
#include <stdio.h>
int main() {
    int arr[] = {00, 02, 05, 10, 12, 32, 72, 96};
    int n = sizeof(arr) / sizeof(arr[0]);
    int x, low = 0, high = n - 1, mid;
    printf("Enter number to search: ");
    scanf("%d", &x);
    while (low <= high) {
        mid = (low + high) / 2;
        if (arr[mid] == x) {
            printf("Found at index %d\n", mid);
            return 0;
        }
        else if (arr[mid] < x)
            low = mid + 1;
        else
            high = mid - 1;
    }
    printf("Not found\n");
    return 0;
}
```



Insertion Sort

The code uses nested loops to shift elements and insert the current element in its correct position. It sorts the array correctly from the example runs.

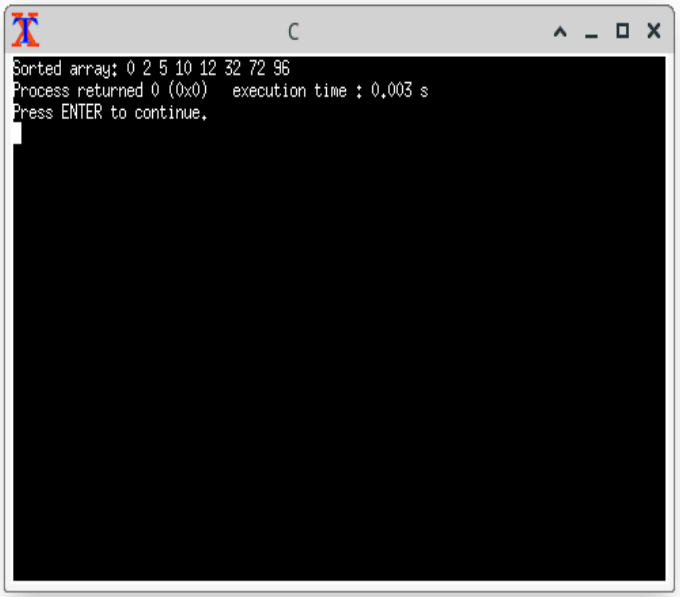
Source Code :

```
#include <stdio.h>

int main() {
    int arr[] = {02, 72, 32, 00, 05, 10, 12, 96};
    int n = sizeof(arr) / sizeof(arr[0]);
    int i, j, temp;
    for (i = 1; i < n; i++) {
        temp = arr[i];
        j = i - 1;
        while (j >= 0 && arr[j] > temp) {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = temp;
    }
    printf("Sorted array: ");
    for (i = 0; i < n; i++)
        printf("%d ", arr[i]);
    return 0;
}
```

Screenshot of the Insertion Sort Code:

lab.c

```c
#include <stdio.h>
int main() {
    int arr[] = {02, 72, 32, 00, 05, 10, 12, 96};
    int n = sizeof(arr) / sizeof(arr[0]);
    int i, j, temp;
    for (i = 1; i < n; i++) {
        temp = arr[i];
        j = i - 1;
        while (j >= 0 && arr[j] > temp) {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = temp;
    }
    printf("Sorted array: ");
    for (i = 0; i < n; i++)
        printf("%d ", arr[i]);
    return 0;
}
```



Bubble Sort

The code uses nested loops to compare adjacent elements and swap them if they are in the wrong order. It successfully sorts the array as expected.

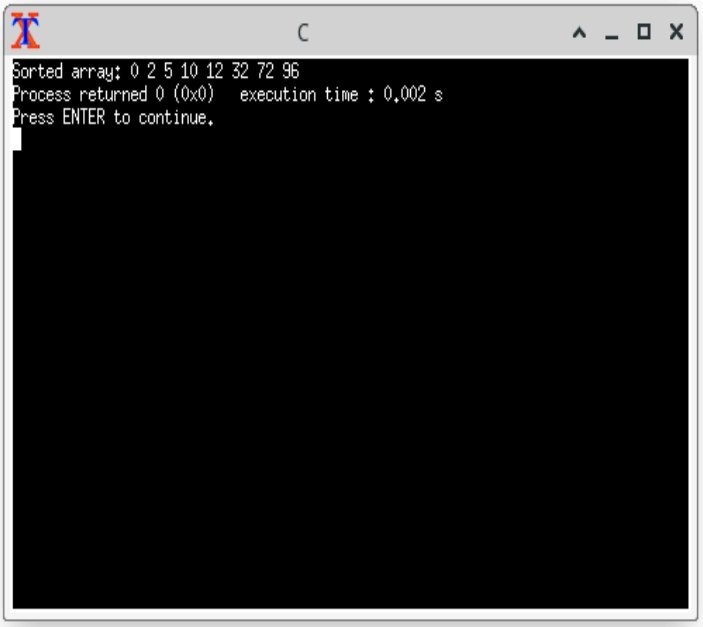
Source Code :

```
#include <stdio.h>
int main() {
    int arr[] = {02, 72, 32, 00, 05, 10, 12, 96};
    int n = sizeof(arr) / sizeof(arr[0]);
    int i, j, temp;
    for (i = 0; i < n - 1; i++)
        for (j = 0; j < n - i - 1; j++)
            if (arr[j] > arr[j + 1]) {
                temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
    printf("Sorted array: ");
    for (i = 0; i < n; i++)
        printf("%d ", arr[i]);
    return 0;
}
```

Screenshot of the Bubble Sort Code:

lab.c

```c
#include <stdio.h>
int main() {
    int arr[] = {02, 72, 32, 00, 05, 10, 12, 96};
    int n = sizeof(arr) / sizeof(arr[0]);
    int i, j, temp;

    for (i = 0; i < n - 1; i++)
        for (j = 0; j < n - i - 1; j++)
            if (arr[j] > arr[j + 1]) {
                temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
    printf("Sorted array: ");
    for (i = 0; i < n; i++)
        printf("%d ", arr[i]);
    return 0;
}
```



Merge Sort

The recursive divide-and-conquer approach with a merge function is correctly implemented. It sorts the array by splitting it into halves, sorting them, and merging them back.

Source Code :

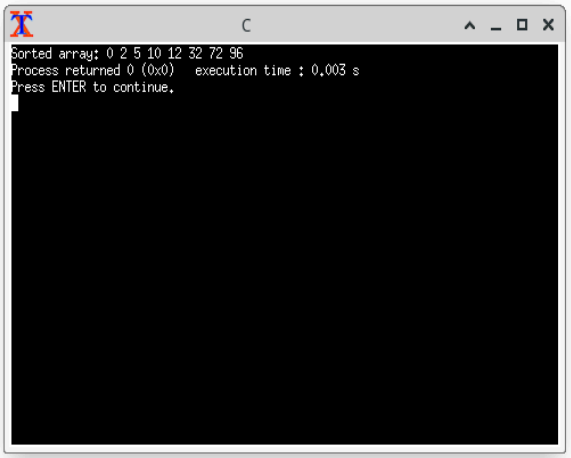
```
#include <stdio.h>
void merge(int arr[], int l, int m, int r) {
    int i, j, k;
    int n1 = m - l + 1;
    int n2 = r - m;
    int L[n1], R[n2];
    for (i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for (j = 0; j < n2; j++)
        R[j] = arr[m + 1 + j];
    i = 0; j = 0; k = l;
    while (i < n1 && j < n2) {
        if (L[i] <= R[j])
            arr[k++] = L[i++];
        else
            arr[k++] = R[j++];
    }
```

```
    while (i < n1)
        arr[k++] = L[i++];
    while (j < n2)
        arr[k++] = R[j++];
}
void mergeSort(int arr[], int l, int r) {
    if (l < r) {
        int m = l + (r - l) / 2;
        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);
        merge(arr, l, m, r);
    }
}
int main() {
    int arr[] = {02, 72, 32, 00, 05, 10, 12, 96};
    int n = sizeof(arr) / sizeof(arr[0]);
    mergeSort(arr, 0, n - 1);
    printf("Sorted array: ");
    for (int i = 0; i < n; i++)
        printf("%d ", arr[i]);
    return 0;
}
```

Screenshot of the Merge Sort Code:

*lab.c

```c
#include <stdio.h>
void merge(int arr[], int l, int m, int r) {
    int i, j, k;
    int n1 = m - l + 1;
    int n2 = r - m;
    int L[n1], R[n2];
    for (i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for (j = 0; j < n2; j++)
        R[j] = arr[m + 1 + j];
    i = 0; j = 0; k = l;
    while (i < n1 && j < n2) {
        if (L[i] <= R[j])
            arr[k++] = L[i++];
        else
            arr[k++] = R[j++];
    }
    while (i < n1)
        arr[k++] = L[i++];
    while (j < n2)
        arr[k++] = R[j++];
}
void mergeSort(int arr[], int l, int r) {
    if (l < r) {
        int m = l + (r - l) / 2;
        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);
        merge(arr, l, m, r);
    }
}
int main() {
    int arr[] = {02, 72, 32, 00, 05, 10, 12, 96};
    int n = sizeof(arr) / sizeof(arr[0]);
```




```c
int main() {
    int arr[] = {02, 72, 32, 00, 05, 10, 12, 96};
    int n = sizeof(arr) / sizeof(arr[0]);
    mergeSort(arr, 0, n - 1);
    printf("Sorted array: ");
    for (int i = 0; i < n; i++)
        printf("%d ", arr[i]);
    return 0;
}
```