



**Green University of Bangladesh**  
**Department of Computer Science and Engineering(CSE)**

**Faculty of Sciences and Engineering**

**Semester: (Summer , Year:2022), B.Sc. in CSE (Day)**

**LAB REPORT NO : 3**

**Course Title: Structured Programming Lab**

**Course Code: CSE 106                      Section:PC-213DA**

**Lab Experiment Name: Bubble Sort , Quick Sort , Merge Sort Using Array.**

**Student Details**

Name	ID
MD Dulal Hossain	213902116

**Lab Date : 06-07-2022**

**Submission Date : 20-07-2022**

**Course Teacher's Name : Farhana Akther Sunny**

**[For Teachers use only: Don't Write Anything inside this box]**

**Lab Report Status**

**Marks: .....**

**Signature:.....**

**Comments:.....**

**Date:.....**

## *//Quick Sort*

### **Algorithm :**

Step 1. Declare array size.

Step 2. User input array.

Step 3. Using structure to quicksort.

Step 4. Find the low and high elements then fix you let low = beg and compare the number if beg getter then compare number , beg go to last and compare number is new beg.

Step 5. This condition Continue until beg== compare number.

Step 6. Print the sorted array.

Step 7. End

### **Source Code :**

```
#include <stdio.h>
Void quicksort (int [], int, int);
Int main()
{
    Int list[50];
    Int size, i;
    Printf("Enter the number of elements: ");
    Scanf("%d", &size);

    Printf("\nEnter the elements to be sorted: ");

    For (i = 0; i < size; i++)
        Scanf("%d", &list[i]);

    Quicksort(list, 0, size - 1);
    Printf("After applying quick sort: \n");
    Printf("\n-----\n");
    For (i = 0; i < size; i++)
```

```

    Printf("%d ", list[i]);

    Printf("\n");
    Return 0;
}
Void quicksort(int list[], int low, int high)
{
    Int pivot, l, j, temp;
    If (low < high)
    {
        Pivot = low;
        l = low;
        j = high;
        While (l < j)
        {
            While (list[l] <= list[pivot] && l <= high)
            {
                l++;
            }
            While (list[j] > list[pivot] && j >= low)
            {
                j--;
            }
            If (l < j)
            {
                Temp = list[l];
                List[l] = list[j];
                List[j] = temp;
            }
        }
        Temp = list[j];
        List[j] = list[pivot];
        List[pivot] = temp;
        Quicksort(list, low, j - 1);
        Quicksort(list, j + 1, high);
    }
}

```

**My Code :**

# Coding C

RUN

MENU

Auto saved at 11:35:27

```
1 #include <stdio.h>
2 void quicksort (int [], int, int);
3 int main()
4 {
5     int list[50];
6     int size, i;
7     printf("Enter the number of elements\n");
8     scanf("%d", &size);
9
10    printf("\nEnter the elements to be sorted\n");
11
12    for (i = 0; i < size; i++)
13        scanf("%d", &list[i]);
14
15    quicksort(list, 0, size - 1);
16    printf("After applying quick sort:\n");
17    printf("\n-----\n");
18    for (i = 0; i < size; i++)
19        printf("%d ", list[i]);
20
21    printf("\n");
22    return 0;
23 }
24 void quicksort(int list[], int low, int high)
25 {
26     int pivot, i, j, temp;
27     if (low < high)
28     {
29         pivot = low;
30         i = low;
31         j = high;
32         while (i < j)
33         {
```

# Coding C

Auto saved at 11:35:38

RUN

MENU

```
23     }
24 void quicksort(int list[], int low, int high)
25 {
26     int pivot, i, j, temp;
27     if (low < high)
28     {
29         pivot = low;
30         i = low;
31         j = high;
32         while (i < j)
33         {
34             while (list[i] <= list[pivot] && i < j)
35             {
36                 i++;
37             }
38             while (list[j] > list[pivot] && j > i)
39             {
40                 j--;
41             }
42             if (i < j)
43             {
44                 temp = list[i];
45                 list[i] = list[j];
46                 list[j] = temp;
47             }
48         }
49         temp = list[j];
50         list[j] = list[pivot];
51         list[pivot] = temp;
52         quicksort(list, low, j - 1);
53         quicksort(list, j + 1, high);
54     }
55 }
```

**Out put :**

## Compile Result

```
Enter the number of elements: 4

Enter the elements to be sorted: 98 11
6 34 56
After applying quick sort:

-----
34 56 98 116

[Process completed - press Enter]
```

*//Merge Sort*

### **Algorithm :**

- Step 1. Declare array size.
- Step 2. User input array.
- Step 3. Using structure to mergesort.

Step 4. Find the middle and let middle is beg and fix this, now we got two side name is left side & right side.

Step 5. Now left side & right side and also two Middle and again two middle let beg and compare.

Step 6. This conditions are continue at last.

Step 7. Print the sorted array.

Step 8. End

## Source Code :

```
#include <stdio.h>
Void Merge(int * , int , int , int );
Void MergeSort(int *array, int left, int right)
{
    Int middle = (left+right)/2;
    If(left<right)
    {
        MergeSort(array, left, middle);
        MergeSort(array, middle + 1, right);
        Merge(array, left, middle, right);
    }
}
Void Merge(int *array, int left, int middle, int right)
{
    Int tmp[right – left + 1];
    Int pos = 0, leftposition = left, rightposition = middle + 1;
    While (leftposition <= middle && rightposition <= right)
    {
        If (array[leftposition] < array[rightposition])
        {
            Tmp[pos++] = array[leftposition++];
        }
        Else
        {
            Tmp[pos++] = array[rightposition++];
        }
    }
}
```



```

    }
    While (leftposition <= middle)
        Tmp[pos++] = array[leftposition++];
    While (rightposition <= right)
        Tmp[pos++] = array[rightposition++];
    Int l;
    For (l = 0; l < pos; l++)
    {
        Array[l + left] = tmp[l];
    }
    Return;
}
Int main()
{
    Int size;
    Printf("Enter the number of elements : ");
    Scanf("%d", &size);
    Int array[size];
    Int l, j, k;
    Printf("\nEnter the elements to be sorted : ");
    For (l = 0; l < size; l++)
    {
        Scanf("%d", &array[l]);
    }
    MergeSort(array, 0, size - 1);

    Printf("After applying Merge sort");

    Printf("\n-----\n");

    For (l = 0; l < size; l++)
    {
        Printf("%d ", array[l]);
    }
    Printf("\n");
    Return 0;
}

```

**My Code :**

# Coding C

Auto saved at 11:35:02

RUN

MENU

```
1 #include <stdio.h>
2 void Merge(int * , int , int , int );
3 void MergeSort(int *array, int left, int right)
4 {
5     int middle = (left+right)/2;
6     if(left<right)
7     {
8         MergeSort(array, left, middle);
9         MergeSort(array, middle + 1, right);
10        Merge(array, left, middle, right);
11    }
12}
13void Merge(int *array, int left, int middle, int right)
14{
15    int tmp[right - left + 1];
16    int pos = 0, leftposition = left, rightposition = middle + 1;
17    while (leftposition <= middle && rightposition <= right)
18    {
19        if (array[leftposition] < array[rightposition])
20        {
21            tmp[pos++] = array[leftposition++];
22        }
23        else
24        {
25            tmp[pos++] = array[rightposition++];
26        }
27    }
28    while (leftposition <= middle)
29        tmp[pos++] = array[leftposition++];
30    while (rightposition <= right)
31        tmp[pos++] = array[rightposition++];
32    int i;
33    for (i = 0; i < pos; i++)
```

# Coding C

Auto saved at 11:35:02

RUN

MENU

```
30     while (rightposition <= right)
31         tmp[pos++] = array[rightposition];
32     int i;
33     for (i = 0; i < pos; i++)
34     {
35         array[i + left] = tmp[i];
36     }
37     return;
38 }
39 int main()
40 {
41     int size;
42     printf("Enter the number of elements: ");
43     scanf("%d", &size);
44     int array[size];
45     int i, j, k;
46     printf("\nEnter the elements to be sorted: ");
47     for (i = 0; i < size; i++)
48     {
49         scanf("%d", &array[i]);
50     }
51     MergeSort(array, 0, size - 1);
52
53     printf("After applying Merge sort: ");
54     printf("\n-----\n");
55
56     for (i = 0; i < size; i++)
57     {
58         printf("%d ", array[i]);
59     }
60     printf("\n");
61     return 0;
62 }
```

Out put :

## Compile Result

```
Enter the number of elements : 5
```

```
Enter the elements to be sorted : 116  
98 32 12 90
```

```
After applying Merge sort
```

```
-----
```

```
12 32 90 98 116
```

```
[Process completed - press Enter]
```