**Logo, company name

Description automatically generated**

**Department of (Computer Science)**

**Pak-Austria Fachhochschule: Institute of Applied Sciences and Technology, Haripur, Pakistan**

**COMP-112L Data Structure** **& Algorithm Lab**

**Lab Journal**

**Class: BS Computer Science**

**Name: Ahmed Raza**

**Registration No.: B20F0436CS031**

**Semester: 4th**

**Submission Date: 26th April 2022**

**Submitted to: Engr. Rafi-Ullah**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Instructor Signature**

**Lab No. 05**

**Sorting Algorithms Quick, Merge and Insertion Sort**

**Objectives:**

In this lab we will be discussing about Sorting Algorithms like Insertion Sort, Quick Sort & Merge Sort in detail. This is one of the most important concepts in C++ language.

The Objective of Lab is:

* To understand & implement the working of sorting algorithms using arrays in C++.

**Tools/Software Required:**

* All the tasks are implemented on DEV C++.

**Introduction:**

**QUICK SORT:**

Quicksort is a fast-sorting algorithm, which is used not only for educational purposes, but widely applied in practice. On the average, it has O (n log n) complexity, making quicksort suitable for sorting big data volumes. The idea of the algorithm is quite simple and once you realize it, you can write quicksort as fast as bubble sort.

**Merge Sort**

Merge sort is based on the divide-and-conquer paradigm. Its worst-case running time has a lower order of growth than insertion sort. Since we are dealing with sub problems, we state each sub problem as sorting a sub array A[p .. r]. Initially, p = 1 and r = n, but these values change as we recurse through sub problems.

**Lab Tasks:**

**Task 1:**

Writing a function to sort array elements using insertion sort. [1,8,4,6,0,3,5,2,7,9]

**Code:**

**#include<iostream>**

**using namespace std;**

**void InsertionSort(int a[],int n);**

**int main()**

**{**

**int n=10;**

**int a[n]={1,8,4,6,0,3,5,2,7,9};**

**cout<<"Original Array :\n";**

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

**{**

**cout<<a[i]<<"\t";**

**}**

**InsertionSort(a,n);**

**}**

**void InsertionSort(int a[],int n)**

**{**

**int key;**

**int i,j;**

**for(i=1;i<n;i++)**

**{**

**key =a[i];**

**j=i-1;**

**while(j>=0 && a[j]>key)**

**{**

**a[j+1] = a[j];**

**j--;**

**}**

**a[j+1]=key;**

**}**

**cout<<"\n\nSorted Array :\n";**

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

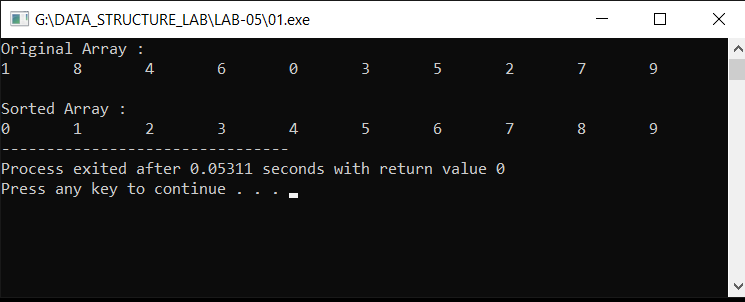
**{**

**cout<<a[i]<<"\t";**

**}**

**}**

**Output:**

****

**Task # 02:**

Write a code to sort the following arrays using Quick sort method. [10, 34, 45, 33, 23, 47, 31, 23, 45, 69, 2, 56,7,67, 88, 42].

**Code:**

**#include <iostream>**

**using namespace std;**

**void swapping(int\* a,int\* b)**

**{**

**int temp=\*a;**

**\*a=\*b;**

**\*b=temp;**

**}**

**int partition(int a[],int l,int h)**

**{**

**int pivot = a[h]; // pivot**

**int i=(l-1);**

**for (int j=l; j<=h-1;j++)**

**{**

**if (a[j]<pivot)**

**{**

**i++;**

**swapping(&a[i],&a[j]);**

**}**

**}**

**swapping(&a[i+1], &a[h]);**

**return (i+1);**

**}**

**void quickSort(int a[],int low,int high)**

**{**

**if (low<high)**

**{**

**int pi=partition(a,low,high);**

**quickSort(a,low,pi-1);**

**quickSort(a,pi+1,high);**

**}**

**}**

**void Show(int a[],int size)**

**{**

**int i;**

**for (i=0;i<size;i++)**

**cout<<a[i]<<"\t";**

**}**

**int main()**

**{**

**int a[] = {10, 34, 45, 33, 23, 47, 31, 23, 45, 69, 2, 56,7,67, 88, 42};**

**int size = sizeof(a)/sizeof(a[0]);**

**cout << "\nThe array is\n";**

**Show(a,size);**

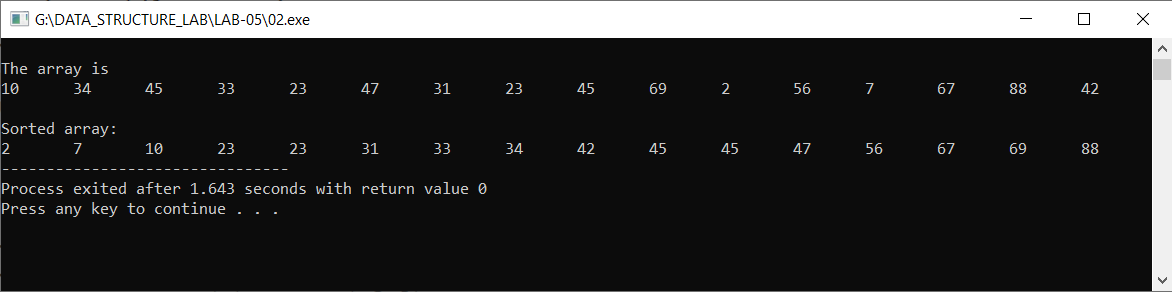
**quickSort(a,0,size-1);**

**cout<<"\n\nSorted array:\n";**

**Show(a,size);**

**}**

**Output:**

****

**Task # 03:**

Write a code to sort the following arrays using Merge sort method. [10, 34, 33, 22, 77, 98, 2, 56, 7, 55, 56, 67, 88, 42]

**Code:**

**#include <iostream>**

**using namespace std;**

**void Merge(int array[],int const left,int const mid,int const right)**

**{**

**int const subArrayOne=mid -left+1;**

**int const subArrayTwo=right-mid;**

**int \*leftArray = new int[subArrayOne];**

**int \*rightArray = new int[subArrayTwo];**

**for (int i=0; i<subArrayOne;i++)**

**leftArray[i]=array[left+i];**

**for (int j=0; j<subArrayTwo;j++)**

**rightArray[j]=array[mid+1+j];**

**int indexOfSubArrayOne=0;**

**int indexOfSubArrayTwo=0;**

**int indexOfMergedArray=left;**

**while (indexOfSubArrayOne<subArrayOne && indexOfSubArrayTwo<subArrayTwo) {**

**if (leftArray[indexOfSubArrayOne]<=rightArray[indexOfSubArrayTwo]) {**

**array[indexOfMergedArray]=leftArray[indexOfSubArrayOne];**

**indexOfSubArrayOne++;**

**}**

**else {**

**array[indexOfMergedArray]=rightArray[indexOfSubArrayTwo];**

**indexOfSubArrayTwo++;**

**}**

**indexOfMergedArray++;**

**}**

**while (indexOfSubArrayOne < subArrayOne) {**

**array[indexOfMergedArray]=leftArray[indexOfSubArrayOne];**

**indexOfSubArrayOne++;**

**indexOfMergedArray++;**

**}**

**while (indexOfSubArrayTwo < subArrayTwo) {**

**array[indexOfMergedArray]=rightArray[indexOfSubArrayTwo];**

**indexOfSubArrayTwo++;**

**indexOfMergedArray++;**

**}**

**}**

**void Merge(int array[],int const begin,int const end)**

**{**

**if (begin>=end)**

**return;**

**int mid=begin+(end-begin)/2;**

**Merge(array,begin,mid);**

**Merge(array,mid+1,end);**

**Merge(array,begin,mid,end);**

**}**

**void Show(int a[],int size)**

**{**

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

**cout<<a[i]<<"\t";**

**}**

**int main()**

**{**

**int a[]={10, 34, 33, 22, 77, 98, 2, 56, 7, 55, 56, 67, 88, 42};**

**int size = sizeof(a)/sizeof(a[0]);**

**cout << "\nThe array is\n";**

**Show(a,size);**

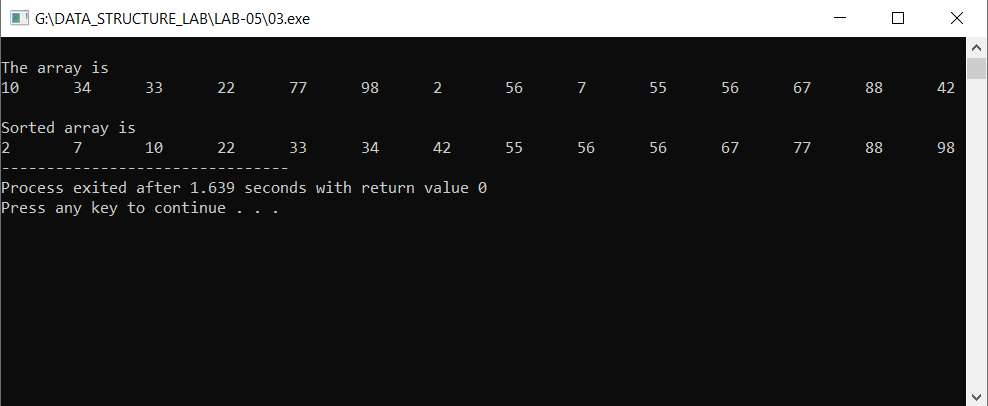
**Merge(a,0,size-1);**

**cout << "\n\nSorted array is \n";**

**Show(a, size);**

**}**

**Output:**

****

**Task # 04:**

Write a code to sort the following arrays using Insertion sort method. [12, 31, 41, 37, 49, 45, 69, 2, 56,76,67, 98, 52]

**Code:**

**#include <iostream>**

**using namespace std;**

**void Merge(int array[],int const left,int const mid,int const right)**

**{**

**int const subArrayOne=mid -left+1;**

**int const subArrayTwo=right-mid;**

**int \*leftArray = new int[subArrayOne];**

**int \*rightArray = new int[subArrayTwo];**

**for (int i=0; i<subArrayOne;i++)**

**leftArray[i]=array[left+i];**

**for (int j=0; j<subArrayTwo;j++)**

**rightArray[j]=array[mid+1+j];**

**int indexOfSubArrayOne=0;**

**int indexOfSubArrayTwo=0;**

**int indexOfMergedArray=left;**

**while(indexOfSubArrayOne<subArrayOne && indexOfSubArrayTwo<subArrayTwo) {**

**if (leftArray[indexOfSubArrayOne]<=rightArray[indexOfSubArrayTwo]) {**

**array[indexOfMergedArray]=leftArray[indexOfSubArrayOne];**

**indexOfSubArrayOne++;**

**}**

**else {**

**array[indexOfMergedArray]=rightArray[indexOfSubArrayTwo];**

**indexOfSubArrayTwo++;**

**}**

**indexOfMergedArray++;**

**}**

**while (indexOfSubArrayOne < subArrayOne) {**

**array[indexOfMergedArray]=leftArray[indexOfSubArrayOne];**

**indexOfSubArrayOne++;**

**indexOfMergedArray++;**

**}**

**while (indexOfSubArrayTwo < subArrayTwo) {**

**array[indexOfMergedArray]=rightArray[indexOfSubArrayTwo];**

**indexOfSubArrayTwo++;**

**indexOfMergedArray++;**

**}**

**}**

**void Merge(int array[],int const begin,int const end)**

**{**

**if (begin>=end)**

**return;**

**int mid=begin+(end-begin)/2;**

**Merge(array,begin,mid);**

**Merge(array,mid+1,end);**

**Merge(array,begin,mid,end);**

**}**

**void Show(int a[],int size)**

**{**

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

**cout<<a[i]<<"\t";**

**}**

**int main()**

**{**

**int a[]={10, 34, 33, 22, 77, 98, 2, 56, 7, 55, 56, 67, 88, 42};**

**int size = sizeof(a)/sizeof(a[0]);**

**cout << "\nThe array is\n";**

**Show(a,size);**

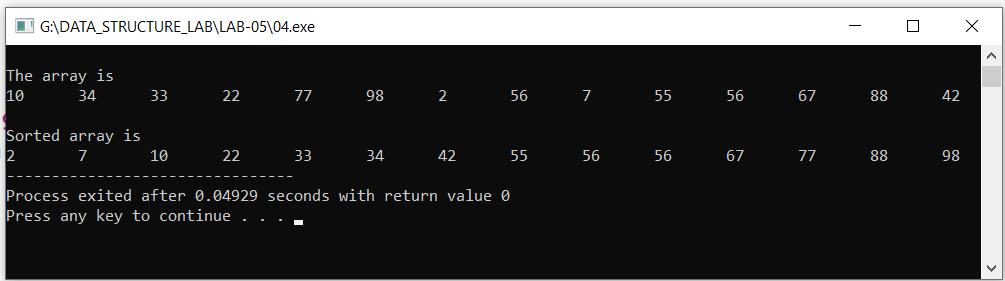
**Merge(a,0,size-1);**

**cout << "\n\nSorted array is \n";**

**Show(a, size);**

**}**

**Output:**

****

**Results & Observations:**

In this Lab I've learned about the concept of Sorting Algorithms like Insertion Sort, Quick Sort & Merge Sort & also understand the difference between Quick Sort and Merge Sort. Further I understand the concept Insertion Sort. In the first task, I've used InsertionSort function in which I've passed two variables one is the size of array and other is the reference of array. In the second task, I've used quickSort function in which I've passed first and last index of array and reference of array. In the Third task, I've used Merge function passing parameters array, left, right and mid index of the array. In the Fourth task, I’ve again use Merge Function.