10659770

DAA Assignment

// Selection sort code //

#include<iostream>

using namespace std;

void swapping (int &m, int &n) // swap the content of m and n

{

int temp;

temp = m;

m = n;

n= temp;

}

void display(int \*array, int size) {

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

cout << array[i] << " ";

cout << endl;

}

void selectionSort(int \*array, int size) {

int i, j, imin;

for(i = 0; i<size-1; i++) {

imin = i;

for(j = i+1; j<size; j++)

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

imin = j;

//placing in correct position

swap(array[i], array[imin]);

}

}

int main() {

int k;

cout << "Enter the number of elements: ";

cin >> k;

int arr[k]; //create an array with k number of elements

cout << "Enter members:" << endl;

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

cin >> arr[i];

}

cout << "Array before Sorting: ";

display(arr, n);

selectionSort(arr, n);

cout << "Array after Sorting: ";

display(arr, n);

}

Coding implementation for bubble sort

#include<iostream>

using namespace std;

int main ()

{

int i, j,temp,pass=0;

int a[10] = {10,2,0,14,43,25,18,1,5,45};

cout <<"Input list ...\n";

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

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

}

cout<<endl;

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

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

{

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

temp = a[i];

a[i] = a[j];

a[j] = temp;

}

}

pass++;

}

cout <<"Sorted Element List ...\n";

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

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

}

cout<<"\kNumber of passes taken to sort the list:"<<pass<<endl;

return 0;

}

// C++ program for implementation of Bubble sort //

The insertion sort algorithm is the sort used by most card players when sorting the cards in their hands. When holding a hand of cards, players will often scan their cards from left to right, looking for the first card that is out of place. For example, if the first three cards of a player's hand are 6,8,3 he will often be satisfied that the 6 and the 8 are in order relative to each other, but, upon getting to the 3, desires to place it before the 6 and the 8. In that case, the player typically removes the 3 from the list, shifts the 6 and the 8 one spot to the right, and then places the 2 into the first slot on the left. This is insertion sort. Unlike other simple sorts like selection sort and bubble sort which rely primarily on comparing and swapping, the insertion sort achieves a sorted data set by identifying an element that is out of order relative to the elements around it, removing it from the list, shifting elements up one place and then placing the removed element in its correct location.

In computer science, selection sort is an in-place comparison sorting algorithm. It has an O(n^2) time complexity where n is the number of elements, which makes it inefficient on large lists. The number of swaps of selection sort is O(n). The selection sort is based on the idea of finding the minimum or maximum element in an unsorted array and then putting it in its correct position. It is a combination of searching and sorting. During each pass, the unsorted element with the smallest or largest value is moved to its proper position in the array. The list is divided into two parts, the sorted part at the left end and the unsorted part at the right end. Initially, the sorted part is empty and the unsorted part is the whole list. The number of times the sort passes through the array is one less than the number of items in the array.