Activity No. 7	
Sorting Algorithms	
Course Code: CPE010	**Program:** Computer Engineering
Course Title: Data Structures and Algorithms	**Date Performed: 10/16/2024**
Section: CPE21S4	**Date Submitted:10/18/2024**
Name(s): Danica T. Guariño	**Instructor: Ma'am Rizette Sayo**

6. Output

Code + Console Screenshot

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

int main() {
    const int size = 100;
    int arr[size];

    std::srand(std::time(0)); // Seed for random number generation
    for (int i = 0; i < size; ++i) {
        arr[i] = std::rand() % 1000 + 1; // Random values between 1 and 1000
    }

    // Display the unsorted array
        cout << "Unsorted Array:\n";
    for (int i = 0; i < size; ++i) {
        cout << arr[i] << " ";
    }
        cout << std::endl;

    return 0;
}
```

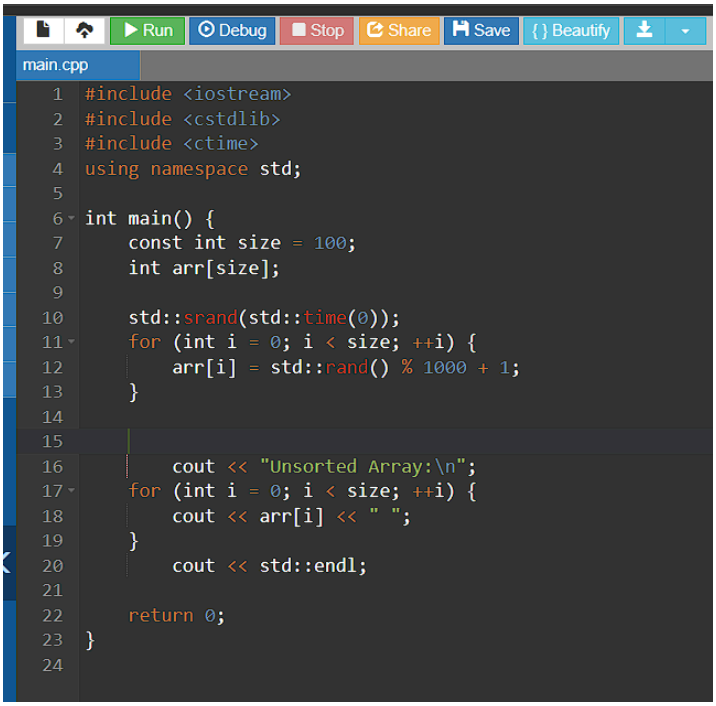
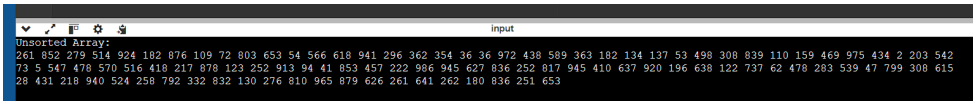
	 
Observations	In this program, the array elements contains 100 random values that are not yet sorted.

Table 7-1. Array of Values for Sort Algorithm Testing

Code + Console Screenshot	bubble.h
	
	#ifndef BUBBLESORT_H
	#define BUBBLESORT_H
	
	#include <algorithm>
	
	template <typename T>
	void bubble(T arr[], size_t arrSize) {
	for (size_t i = 0; i < arrSize - 1; i++) {
	for (size_t j = 0; j < arrSize - i - 1; j++) {
	if (arr[j] > arr[j + 1]) {
	std::swap(arr[j], arr[j + 1]);
	}
	}
	}
	}
	
	#endif // BUBBLESORT_H

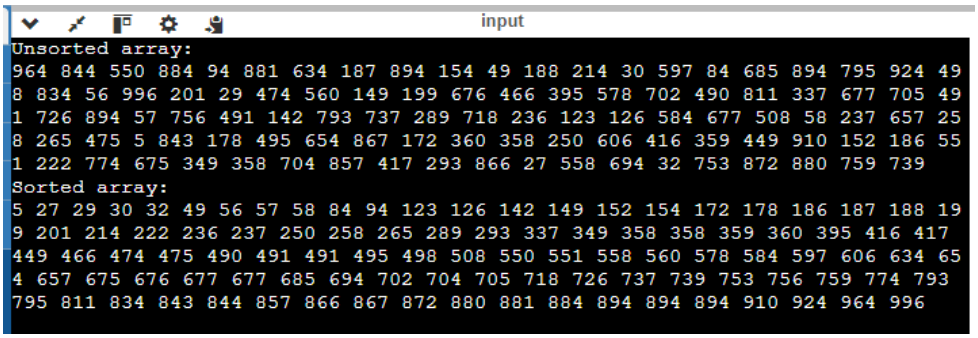
	
Observations	bubble.h function is used to sort an array, it's function is to compare the elements that has similar side or point. It will swap them if it is not in the right order.

Table 7-2. Bubble Sort Technique

Code + Console Screenshot	selection.h
	
	#ifndef SELECTIONSORT_H
	#define SELECTIONSORT_H
	
	// Find the position of the smallest element in the unsorted portion
	template <typename T>
	int Routine_Smallest(T arr[], int start, int size) {
	int pos = start;
	for (int j = start + 1; j < size; j++) {
	if (arr[j] < arr[pos]) {
	pos = j;
	}
	}
	return pos;
	}
	
	// Selection sort implementation
	template <typename T>
	void selectionSort(T arr[], const int size) {
	for (int i = 0; i < size - 1; i++) {
	int pos = Routine_Smallest(arr, i, size);
	if (pos != i) {
	std::swap(arr[i], arr[pos]); // Swap the smallest found with the current position
	}
	}
	}
	
	#endif // SELECTIONSORT_H

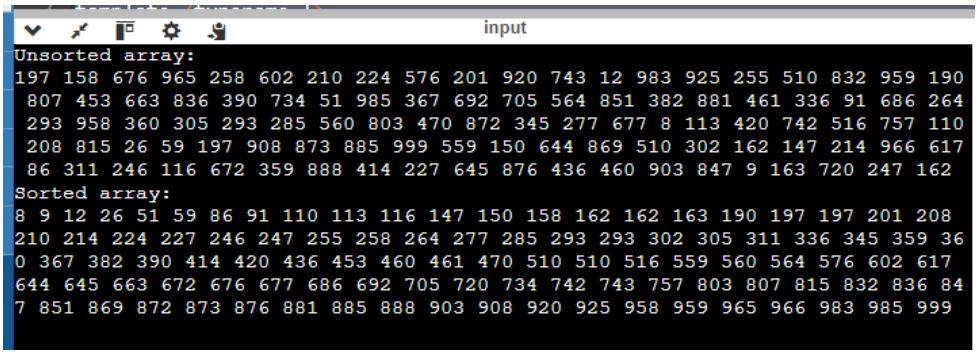
	
Observations	This function sorts the array of random numbers. From smallest element to the first unsorted element.

Table 7-3. Selection Sort Algorithm

Code + Console Screenshot	insertion.h
	
	#ifndef INSERTIONSORT_H
	#define INSERTIONSORT_H
	
	// Insertion sort function template
	template <typename T>
	void insertionSort(T arr[], const int N) {
	  for (int K = 1; K < N; K++) {
	  T temp = arr[K];
	  int J = K - 1;
	
	  // Move elements greater than temp to one position ahead
	  while (J >= 0 && temp < arr[J]) {
	  arr[J + 1] = arr[J];
	  J--;
	  }
	
	  // Place temp in the correct position
	  arr[J + 1] = temp;
	  }
	}
	
	#endif // INSERTIONSORT_H

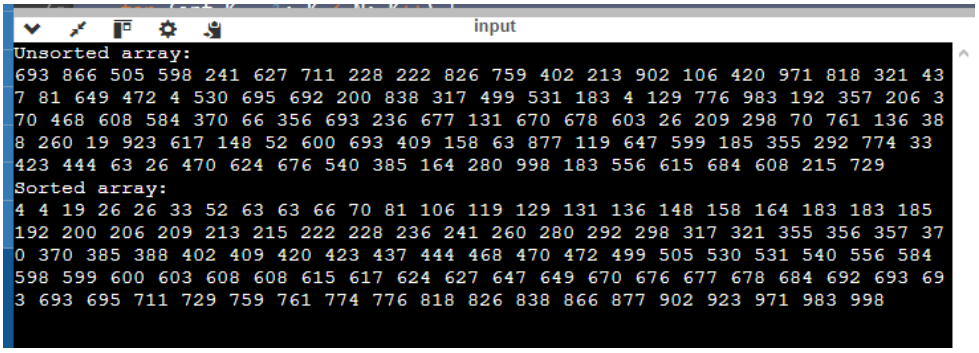
	
Observations	Using insertion.h, an element from the unsorted array was selected and placed into its correct position within the sorted array.

Table 7-4. Insertion Sort Algorithm

7. Supplementary Activity

ILO B: Solve given data sorting problems using appropriate basic sorting algorithms

Candidate 1	Bo Dalton Capistrano
Candidate 2	Cornelius Raymon Agustín
Candidate 3	Deja Jayla Bañaga
Candidate 4	Lalla Brielle Yabut
Candidate 5	Franklin Relano Castro

List of Candidates

Problem: Generate an array A[0...100] of unsorted elements, wherein the values in the array are indicative of a vote to a candidate. This means that the values in your array must only range from 1 to 5. Using sorting and searching techniques, develop an algorithm that will count the votes and indicate the winning candidate.

NOTE: The sorting techniques you have the option of using in this activity can be either bubble, selection, or insertion sort.

Justify why you chose to use this sorting algorithm.

I used bubble sort since it's simple to implement and understand, which makes it a good choice for small datasets. It is much easier to use and fast to understand the output of the program created.

• Pseudocode of Algorithm

Initialize SIZE to 100

Function generateVotes(arr, size):
  Seed random number generator with current time

```
    For i from 0 to size-1:
        arr[i] = random number between 1 and 5

Function bubbleSort(arr, size):
    For i from 0 to size-2:
        For j from 0 to size-i-2:
            If arr[j] > arr[j+1]:
                Swap arr[j] and arr[j+1]

Function countVotes(arr, size, counts, numCandidates):
    For i from 0 to size-1:
        counts[arr[i] - 1]++

Function findWinner(counts, numCandidates):
    maxVotes = counts[0]
    winner = 1
    For i from 1 to numCandidates-1:
        If counts[i] > maxVotes:
            maxVotes = counts[i]
            winner = i + 1
    Return winner

Main function:
    Declare votes array of size SIZE
    Call generateVotes(votes, SIZE)
    Print "Unsorted Votes: " followed by votes array
    Call bubbleSort(votes, SIZE)
    Print "Sorted Votes: " followed by votes array
    Declare voteCounts array of size 5 and initialize to 0
    Call countVotes(votes, SIZE, voteCounts, 5)
    Print "Vote Counts: " followed by counts for each candidate
    Call findWinner(voteCounts, 5)
    Print the winning candidate
```

• **Screenshot of Algorithm Code**

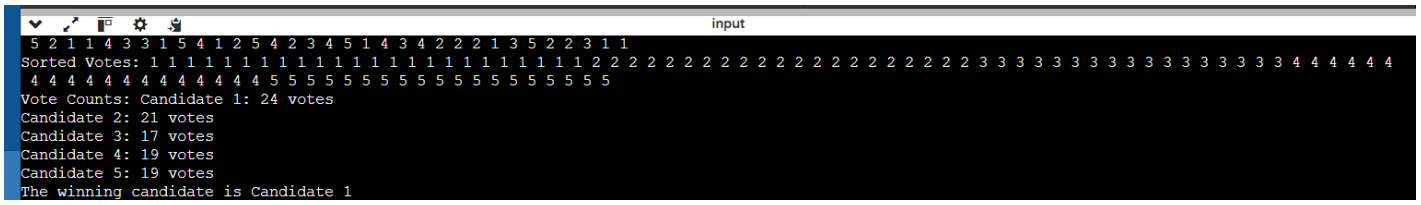
C/C++

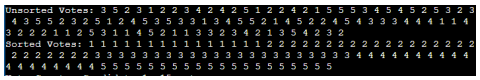
```
#include <iostream>
#include <cstdlib>
#include <ctime>

const int SIZE = 100;

void generateVotes(int arr[], int size) {
    srand(time(0));
    for (int i = 0; i < size; ++i) {
        arr[i] = rand() % 5 + 1;
```

```
    }
}

void bubbleSort(int arr[], int size) {
    for (int i = 0; i < size - 1; ++i) {
        for (int j = 0; j < size - i - 1; ++j) {
            if (arr[j] > arr[j + 1]) {
                std::swap(arr[j], arr[j + 1]);
            }
        }
    }
}

void countVotes(int arr[], int size, int counts[], int numCandidates) {
    for (int i = 0; i < size; ++i) {
        counts[arr[i] - 1]++;
    }
}

int findWinner(int counts[], int numCandidates) {
    int maxVotes = counts[0];
    int winner = 1;
    for (int i = 1; i < numCandidates; ++i) {
        if (counts[i] > maxVotes) {
            maxVotes = counts[i];
            winner = i + 1;
        }
    }
    return winner;
}

int main() {
    int votes[SIZE];
    generateVotes(votes, SIZE);

    std::cout << "Unsorted Votes: ";
    for (int i = 0; i < SIZE; ++i) {
        std::cout << votes[i] << " ";
    }
    std::cout << std::endl;

    bubbleSort(votes, SIZE);

    std::cout << "Sorted Votes: ";
    for (int i = 0; i < SIZE; ++i) {
        std::cout << votes[i] << " ";
    }
    std::cout << std::endl;

    int voteCounts[5] = {0};
    countVotes(votes, SIZE, voteCounts, 5);

    std::cout << "Vote Counts: ";
    for (int i = 0; i < 5; ++i) {
        std::cout << "Candidate " << i + 1 << ": " << voteCounts[i] << " votes" <<
std::endl;
```

```
    }

    int winner = findWinner(voteCounts, 5);
    std::cout << "The winning candidate is Candidate " << winner << std::endl;

    return 0;
}
```

• Output Testing



Output Console Showing Sorted Array	Manual Count	Count Result of Algorithm
	Vote Counts Candidate 1: 15 votes Candidate 2: 25 votes Candidate 3: 22 votes Candidate 4: 18 votes Candidate 5: 20 votes	The winning candidate is Candidate 2

Question: Was your developed vote counting algorithm effective? Why or why not?

Yes, it is effective since it is much easier to understand the program since I used bubble sort and it was sorted properly.

8. Conclusion

In conclusion, using sorting techniques in C++ provide a variety of approaches for effectively managing various data sets. While simpler algorithms, such as Bubble Sort, are simple to develop and understand, they may be inefficient for large datasets due to current complexity. Choosing the appropriate sorting algorithm is determined by the application's specific needs, such as dataset size, complexity, and necessary stability. To summarize, knowing a variety of sorting algorithms enables optimized and efficient data processing, making it an essential skill in C++ programming. Having to know how to sort out any unsorted elements is such a great tactic that I can use for my future codings.

9. Assessment Rubric