

Activity No. 7.1	
SORTING ALGORITHMS: BUBBLE, SELECTION, AND INSERTION SORT	
Course Code: CPE010	Program: Computer Engineering
Course Title: Data Structures and Algorithms	Date Performed: 10/16/24
Section: CPE21S4	Date Submitted: 10/16/24
Name(s): CALVIN EARL PLANTA	Instructor: Prof. Sayo

6. Output

Code + Console Screenshot	<div><div>C/C++</div><pre>#include <iostream> #include <cstdlib> #include <ctime> using namespace std; int main(){ const int size = 100; int arr[size]; srand(static_cast<unsigned int>(time(0))); for (int i = 0; i < size; i++){ arr[i] = rand() % 100; } cout << "Random generated array (unsorted): " << endl; for (int i = 0; i < size; i++){ cout << arr[i] << " "; } return 0; }</pre><div><div>main.cpp</div><div>1 #include <iostream> 2 #include <cstdlib> 3 #include <ctime> 4 using namespace std; 5 6- int main(){ 7 const int size = 100; 8 int arr[size]; 9 srand(static_cast<unsigned int>(time(0))); 10 11- for (int i = 0; i < size; i++){ 12 arr[i] = rand() % 100; 13 } 14 15 cout << "Random generated array (unsorted): " << endl; 16- for (int i = 0; i < size; i++){ 17 cout << arr[i] << " "; 18 } 19 20 return 0; 21 }</div><div>Output</div><div>/tmp/v3R0regyXc.o Random generated array (unsorted): 58 97 0 42 38 45 17 89 94 40 89 11 16 83 40 37 97 78 67 42 19 91 91 7 41 51 17 8 44 85 1 3 34 53 97 25 98 14 14 44 54 3 55 71 86 48 60 36 26 27 78 45 71 69 53 12 72 70 72 17 7 25 72 94 78 69 19 28 83 33 24 89 88 79 12 26 79 25 62 5 4 40 3 75 62 8 39 34 30 11 3 89 36 27 83 66 96 2 94 31 === Code Execution Successful ===</div></div></div>
Observations	<p>The output of the program seemed complex yet the code was kept simple. The array was generated using rand() inside a for loop function that iterates as long as the iteration count is less than the size of the array.</p>

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

Code + Console Screenshot

```
C/C++

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

int main() {
    const int size = 100;
    int arr[size];
    srand(static_cast<unsigned int>(time(0)));

    for (int i = 0; i < size; i++) {
        arr[i] = rand() % 100;
    }

    cout << "Random generated array (unsorted): " << endl;
    for (int i = 0; i < size; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;

    for (int i = 0; i < size - 1; i++) {
        for (int j = 0; j < size - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                // Swap arr[j] and arr[j + 1]
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }

    cout << "Sorted array: " << endl;
    for (int i = 0; i < size; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;

    return 0;
}
```

The screenshot shows a C++ IDE with a code editor on the left and an output console on the right. The code editor contains the same C++ code as shown in the previous block. The output console shows the execution results, including the unsorted array, the sorted array, and a success message.

```
main.cpp 1 #include <iostream>
2 #include <cstdlib>
3 #include <ctime>
4 using namespace std;
5
6 int main() {
7     const int size = 100;
8     int arr[size];
9     srand(static_cast<unsigned int>(time(0)));
10
11     for (int i = 0; i < size; i++) {
12         arr[i] = rand() % 100;
13     }
14
15     cout << "Random generated array (unsorted): " << endl;
16     for (int i = 0; i < size; i++) {
17         cout << arr[i] << " ";
18     }
19     cout << endl;
20
21     for (int i = 0; i < size - 1; i++) {
22         for (int j = 0; j < size - i - 1; j++) {
23             if (arr[j] > arr[j + 1]) {
24                 // Swap arr[j] and arr[j + 1]
25                 int temp = arr[j];
26                 arr[j] = arr[j + 1];
27                 arr[j + 1] = temp;
28             }
29         }
30     }
31
32     cout << "Sorted array: " << endl;
33     for (int i = 0; i < size; i++) {
34         cout << arr[i] << " ";
35     }
36     cout << endl;
37 }
```

Output

```
/tmp/1JUVK00e.o
Random generated array (unsorted):
29 32 8 39 9 73 85 78 78 6 58 91 84 47 86 61 55 38 30 17 23 56 72 89 61 80 68 7 56 26 22 76 10 82
59 20 8 44 98 38 50 8 29 66 55 16 27 10 54 10 80 29 66 52 70 27 84 39 86 40 17 8 69 28 43 28 48
51 24 98 89 26 59 71 92 66 39 19 29 45 29 9 27 47 13 97 26 98 88 13 90 6 73 11 86 16 91 34 19
15
Sorted array:
6 6 7 8 8 8 8 9 9 10 10 10 11 13 13 15 16 16 17 17 19 19 20 20 22 23 24 26 26 26 27 27 27 28 28 29
29 29 29 30 30 32 34 38 38 39 39 40 43 44 45 47 47 48 50 51 52 54 55 55 56 56 58 59 59 61 61 64
66 66 66 68 69 70 71 72 73 73 76 78 78 80 82 84 85 86 86 86 88 89 89 90 91 91 92 97 98 98 98
15
--- Code Execution Successful ---
```

Observations	While it is educational and practical for small datasets, its inefficiency for larger arrays highlights the need for more advanced sorting algorithms in real-world applications.
--------------	---

Table 7-2. Bubble Sort Technique

Code + Console Screenshot	<pre>C/C++ #include <iostream> #include <cstdlib> #include <ctime> using namespace std; int main() { const int size = 100; int arr[size]; srand(static_cast<unsigned int>(time(0))); for (int i = 0; i < size; i++) { arr[i] = rand() % 100; } cout << "Random generated array (unsorted): " << endl; for (int i = 0; i < size; i++) { cout << arr[i] << " "; } cout << endl; for (int i = 0; i < size - 1; i++) { int minIndex = i; for (int j = i + 1; j < size; j++) { if (arr[j] < arr[minIndex]) { minIndex = j; } } if (minIndex != i) { int temp = arr[i]; arr[i] = arr[minIndex]; arr[minIndex] = temp; } } cout << "Sorted array: " << endl; for (int i = 0; i < size; i++) { cout << arr[i] << " "; } cout << endl; return 0; }</pre>
---------------------------	--

	<div><div>main.cpp</div><div><div><div>1 #include <iostream></div><div>2 #include <cstdlib></div><div>3 #include <ctime></div><div>4 using namespace std;</div><div>5</div><div>6 int main() {</div><div>7 const int size = 100;</div><div>8 int arr[size];</div><div>9 srand(static_cast<unsigned int>(time(0)));</div><div>10</div><div>11 for (int i = 0; i < size; i++) {</div><div>12 arr[i] = rand() % 100;</div><div>13 }</div><div>14</div><div>15 cout << "Random generated array (unsorted): " << endl;</div><div>16 for (int i = 0; i < size; i++) {</div><div>17 cout << arr[i] << " ";</div><div>18 }</div><div>19 cout << endl;</div><div>20</div><div>21 for (int i = 0; i < size - 1; i++) {</div><div>22 int minIndex = i;</div><div>23 for (int j = i + 1; j < size; j++) {</div><div>24 if (arr[j] < arr[minIndex]) {</div><div>25 minIndex = j;</div><div>26 }</div><div>27 }</div><div>28 if (minIndex != i) {</div><div>29 int temp = arr[i];</div><div>30 arr[i] = arr[minIndex];</div><div>31 arr[minIndex] = temp;</div><div>32 }</div><div>33 }</div><div>34</div><div>35 cout << "Sorted array: " << endl;</div><div>36 for (int i = 0; i < size; i++) {</div></div></div><div><div>Output</div><div><div>/tmp/c08Q5RgK48.o</div><div>Random generated array (unsorted):</div><div>71 58 90 19 3 11 33 69 2 51 48 56 54 5 43 90 75 72 23 16 29 21 72 55 58 95 60 11 62 81 98 85 92 40</div><div>4 95 51 90 16 54 41 16 10 95 21 5 37 49 77 61 65 6 82 89 14 92 84 26 4 98 7 54 83 99 94 40 46</div><div>98 30 15 52 71 31 14 19 5 19 56 54 48 69 71 6 4 60 72 96 96 98 52 95 6 58 30 57 5 22 56 3 52</div><div>Sorted array:</div><div>2 3 4 4 4 5 5 5 5 6 6 7 10 11 11 14 14 15 16 16 16 19 19 19 21 21 22 23 26 29 30 30 31 33 37 40</div><div>40 41 43 46 48 48 49 51 51 52 52 54 54 54 55 56 56 57 58 58 58 60 60 61 62 65 69 69 71</div><div>71 71 72 72 72 75 77 81 82 83 84 85 89 90 90 90 92 92 94 95 95 95 95 96 96 98 98 98 99</div></div><div>=== Code Execution Successful ===</div></div></div>
Observations	While selection sort may not be suitable for large datasets due to its inefficiency, it offers a clear understanding of sorting mechanics and serves well in educational contexts.

Table 7-3. Selection Sort Algorithm

Code + Console Screenshot	<div><div>C/C++</div><div><pre>#include <iostream> #include <cstdlib> #include <ctime> using namespace std; int main() { const int size = 100; int arr[size]; srand(static_cast<unsigned int>(time(0))); for (int i = 0; i < size; i++) { arr[i] = rand() % 100; } cout << "Random generated array (unsorted): " << endl; for (int i = 0; i < size; i++) { cout << arr[i] << " "; } cout << endl; // Insertion sort algorithm for (int i = 1; i < size; i++) { int key = arr[i]; int j = i - 1; while (j >= 0 && arr[j] > key) { arr[j + 1] = arr[j]; j--; } } }</pre></div></div>
---------------------------	--

```

arr[j + 1] = key;
}

cout << "Sorted array: " << endl;
for (int i = 0; i < size; i++) {
    cout << arr[i] << " ";
}
cout << endl;

return 0;
}

```

The screenshot shows a C++ IDE with a file named 'main.cpp'. The code implements an insertion sort algorithm. It starts by including `<iostream>` and `<cstdlib>`, and using the `std` namespace. In the `main` function, a constant `size` of 100 is defined, and an array `arr` of that size is declared. A random seed is generated using `srand(time(0))`. A loop generates 100 random numbers between 0 and 100, storing them in the `arr` array. The program then prints the 'Random generated array (unsorted)'. Next, it implements the insertion sort algorithm: for each element from index 1 to 99, it finds its correct position in the sorted subarray (elements from index 0 to `j-1`) and inserts it. Finally, it prints the 'Sorted array'.

The output window shows the following results:

```

/tmp/lysXRS16Ag.o
Random generated array (unsorted):
6 12 34 38 82 96 5 44 50 68 85 92 16 71 79 58 81 13 41 64 17 91 89 21 28 29 19 42 85 71 48 92 35 35
30 17 83 87 14 86 55 51 30 24 23 61 34 56 74 28 72 43 71 14 65 0 43 36 42 29 59 42 21 95 29 3
64 65 90 78 3 97 82 33 21 57 94 8 65 21 36 37 64 7 51 29 59 95 18 53 76 77 48 49 24 29 52 89 94
94
Sorted array:
0 3 3 5 5 7 8 12 13 14 14 16 17 17 18 19 21 21 21 23 24 24 28 28 29 29 29 29 30 30 33 34 34
35 35 36 36 37 38 41 42 42 42 43 43 44 48 48 49 50 51 51 52 53 55 56 57 58 59 59 61 64 64 64 65
65 65 68 71 71 71 72 74 76 77 78 79 81 82 82 83 85 85 86 87 89 89 90 91 92 92 94 94 94 95 95 96
97

=== Code Execution Successful ===

```

Observations	While insertion sort may not be optimal for large datasets, it is an excellent educational tool and performs well with small or partially sorted datasets.
--------------	--

Table 7-4. Insertion Sort Algorithm

7. Supplementary Activity

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

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.

```
C/C++
#include <iostream>
#include <cstdlib>
#include <ctime>

using namespace std;

void selectionSort(int arr[], int size) {
    for (int i = 0; i < size - 1; i++) {
        int minIndex = i;
        for (int j = i + 1; j < size; j++) {
            if (arr[j] < arr[minIndex]) {
                minIndex = j;
            }
        }

        int temp = arr[minIndex];
        arr[minIndex] = arr[i];
        arr[i] = temp;
    }
}

void countVotes(int votes[], int size, int voteCount[], int numCandidates) {

    for (int i = 0; i < numCandidates; i++) {
        voteCount[i] = 0;
    }

    for (int i = 0; i < size; i++) {
        voteCount[votes[i] - 1]++;
    }
}

int findWinner(int voteCount[], int numCandidates) {
    int maxVotes = voteCount[0];
    int winnerIndex = 0;

    for (int i = 1; i < numCandidates; i++) {
        if (voteCount[i] > maxVotes) {
            maxVotes = voteCount[i];
            winnerIndex = i;
        }
    }

    return winnerIndex;
}
```

```

void printResults(int voteCount[], int numCandidates, string candidates[]) {
    for (int i = 0; i < numCandidates; i++) {
        cout << candidates[i] << " received " << voteCount[i] << " votes." << endl;
    }
}

void displayArray(int arr[], int size, const string& message) {
    cout << message << ": [ ";
    for (int i = 0; i < size; i++) {
        cout << arr[i] << " ";
    }
    cout << "]" << endl;
}

int main() {
    const int SIZE = 101;
    int votes[SIZE];
    const int NUM_CANDIDATES = 5;
    int voteCount[NUM_CANDIDATES];

    string candidates[NUM_CANDIDATES] = {
        "Bo Dalton Capistrano",
        "Cornelius Raymon Agustín",
        "Deja Jayla Bañaga",
        "Lalla Brielle Yabut",
        "Franklin Relano Castro"
    };

    srand(time(0));

    for (int i = 0; i < SIZE; i++) {
        votes[i] = (rand() % NUM_CANDIDATES) + 1;
    }

    displayArray(votes, SIZE, "Generated Array");

    selectionSort(votes, SIZE);

    displayArray(votes, SIZE, "Sorted Array");

    countVotes(votes, SIZE, voteCount, NUM_CANDIDATES);

    printResults(voteCount, NUM_CANDIDATES, candidates);

    int winnerIndex = findWinner(voteCount, NUM_CANDIDATES);

    cout << "The winner is: " << candidates[winnerIndex] << " with " <<
    voteCount[winnerIndex] << " votes." << endl;

    return 0;
}

```

