Hands-on Activity 4.2	
Arrays	
Course Code: CPE007	**Program:** Computer Engineering
Course Title: Programming Logic and Design	**Date Performed: September 11, 2025**
Section: CPE11S1	**Date Submitted: September 13, 2025**
Name(s): Jaime Luis M. Demain	**Instructor: Engr. Jimlord M. Quejado**

6. Output

Example of initializing an array:

Code:

```cpp
#include <iostream>
using namespace std;

int main() {
    int n[10];

    // Initialize array elements to 0
    for (int i = 0; i < 10; i++) {
        n[i] = 0;
    }

    cout << "Element   Value" << endl;

    // Print index and value
    for (int i = 0; i < 10; i++) {
        cout << "    " << i << "         " << n[i] << endl;
    }

    return 0;
}
```

Output:

```
Element   Value
    0         0
    1         0
    2         0
    3         0
    4         0
    5         0
    6         0
    7         0
    8         0
    9         0

--------------------------------
Process exited after 0.01513 seconds with return value 0
Press any key to continue . . .
```

Analysis: We will know how to initialize arrays, display our intended message with the number of indexes and its respective value. But in this part, we will not declare a value aside from 0 as per instruction. The first thing to do is we will input the data type of our array, the array name, and its intended size. And then we will initialize the array to 0 so that the only index value that will display is 0 and it will print the value of index 0-9. After that, we will print the "Element" (number of index) and "Value" (value of respective index) message with space between them plus don't forget the endl. And then, we will print the index and its value under the previous messages with proper spacing using a for-loop statement and also input the endlt. And that's only what we will do in this part.

Example of initializing an array with a declaration:

Code:

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

int main() {
    int n[10] = {32, 27, 64, 18, 95, 14, 90, 70, 60, 37};

    cout << "Element  Value" << endl;

    for (int i = 0; i < 10; i++) {
        cout << "    " << i << "       " << n[i] << endl;
    }

    return 0;
}
```

Output:

```
Element  Value
   0       32
   1       27
   2       64
   3       18
   4       95
   5       14
   6       90
   7       70
   8       60
   9       37

--------------------------------
Process exited after 0.02399 seconds with return value 0
Press any key to continue . . .
```

Analysis: In this part, we will now declare a multiple value of indexes aside from 0. To do this, type a bracket first after declaring the array name and its intended size before declaring the intended values. And then, type the intended values within the bracket. After that, print the intended message that instructs where the index and value are. And then, initialize a for-loop to display the number of indexes and its respective value. Print the number and its respective value after with proper spacing and endl on it.

Example of computing sum of elements of the array:

Code:

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

#define SIZE 12

int main() {
    int a[SIZE] = {1, 3, 5, 4, 7, 2, 99, 16, 45, 67, 89, 45};
    int total = 0;

    for (int i = 0; i < SIZE; i++) {
        total += a[i];
    }

    cout << "Total of array element values is " << total << endl;
    return 0;
}
```

Output:

```
Total of array element values is 383

--------------------------------
Process exited after 0.01015 seconds with return value 0
Press any key to continue . . .
```

Analysis: In this part, we will compute the sum of values in the array. Before this, input a #define directive first to initialize a “SIZE” variable to 12 before the compilation of code so that the “SIZE” variable will be pre-instructed to be “12”. Initialize the array after so that the values to be stored in array will be initialize. And then, declare the “total” as variable and initialize it to 0 so that the value starts at 0. Initialize a for-loop after to execute a sum computation operation for array values. Then, print the intended message with “total” variable to display the total of array values.

7. Supplementary Activity

1.

Code:

```cpp
#include <iostream>
using namespace std;

int main() {
    int n[10] = {19, 3, 15, 7, 11, 9, 13, 5, 17, 1};
    cout << "Element      Value      Histogram" << endl;

    for (int i = 0; i < 10; i++) {
        cout << "\t" << i << "\t" << n[i] << "\t";
        for (int j = 0; j < n[i]; j++){
            cout << "*";
        }
        cout << endl;
    }

    return 0;
}
```

Output:

```
Element      Value      Histogram
        0       19      ********************
        1       3       ***
        2       15      ***************
        3       7       *******
        4       11      ***********
        5       9       *********
        6       13      *************
        7       5       *****
        8       17      *****************
        9       1       *

--------------------------------
Process exited after 0.01633 seconds with return value 0
Press any key to continue . . .
```

Analysis:

In this part, we will initialize an array with a declaration and execute a histogram for the respective array values using *. But in order to execute a histogram with * as visualization, we will use a nested loop. Before going to the nested loop for histogram execution, we will first initialize an array, display the intended top messages for instruction on where the index, its value, and the value's histogram is. After this, we can now proceed to the nested loop. Nested loop consists of two parts, namely outer loop and inner loop. In this case, the outer loop will be used to process the index and value, to display the index, value, and the histogram, and also to provide the proper horizontal spacing between them. Inner loop will be used to process the value's histogram and also to use * to visualize the histogram. Finally, the index, value, and the histogram will now be displayed.

2.

Code:

```cpp
#include <iostream>
using namespace std;

int main() {
    const int RESPONSE_SIZE = 40;
    int responses[RESPONSE_SIZE] = { 1, 2, 6, 4, 8, 5, 9, 7, 8, 10, 1, 6, 3, 8, 6, 10, 3, 8, 2, 7, 6, 5, 7, 6, 8, 6, 6, 5, 6, 6, 5, 6, 7, 5, 6, 4, 8, 6, 8, 10};
    cout << "Response Summary: " << endl;
    int number [11] = {};
    for (int i = 0; i < RESPONSE_SIZE; i++){
        number[responses[i]]++;
    }
    for (int i = 1; i < 11; i++) {
        cout << "Response " << i << ":" << ' ' << number[i] << " students" << endl;
    }

    return 0;
}
```

Output:

```
Response Summary:
Response 1: 2 students
Response 2: 2 students
Response 3: 2 students
Response 4: 2 students
Response 5: 5 students
Response 6: 12 students
Response 7: 4 students
Response 8: 7 students
Response 9: 1 students
Response 10: 3 students

--------------------------------
Process exited after 0.01484 seconds with return value 0
Press any key to continue . . .
```

Analysis:

In this part, we will count how many times each number has been responded by the students using an array. Before thhat, we need to initialize an array first. The first step is to declare a specific variable called “RESPONSE_SIZE” so that it will represent the size of an array after initializing and to be used multiple times. After that, print out the “Response Summary: “ message with endl. And then we can now proceed to program the counting mechanism. First, we will execute a frequency array so that it will process on counting the specific number on the array to have an enabling mechanism for counting responses later on. I inputted int number[11] = {} for frequency array, [11] represents the index from index 1-10 (index 0 will not be used in actual counting and it is a placeholder to ensure smooth counting) and {} to initialize all the values of every index to 0 for proper counting. And then, we will proceed and process a for-loop statement to actually count the frequency of every number in the array. frequency[responses[i]]++ represents the incrementing process from index 1-10 to represent the number and its respective frequencies and the frequency of that number will print after the loop. After this, print out the information that represents the frequencies of student responses on a particular number.

8. Conclusion

In this activity, I learned the different executing mechanisms on arrays to display its value like how to display its values using a for-loop and also display the value’s histogram, perform computation operations on values of an array, and count the frequency of a value in an array. It is a complicated activity but worth studying. The first part is how I will display an array index and its respective elements or values. It is only a single value because I initialize the array to only 0. The

second part is how I will display the same with multiple values declared. The third part may be the most simple part to me as it only requires performing a sum computation operation of an array and I already learned the mathematical operations prior. The fourth one, I learned something new there and it is a nested loop. Nested loop can be used to perform multi-level loops like this one on executing the histogram of values in an array. The fifth part can be the most complicated part to me and it is counting the frequency of a student's response on each number. I learned about frequency array and a for-loop mechanism to execute the counting of value frequencies on an array.