Solution Review: Sort Elements of an Array in Descending Order

Let's go over the solution review of the challenge given in the previous lesson.

We'll cover the following

- Solution
- Explanation
  - sort_elements function

Solution

Press the **RUN** button and see the output!

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

// sort_elements function
void sort_elements(int arr[], int size) {
  // Outer loop
  for (int i = 0; i < size; i++) {
    // Inner loop
    for (int j = i + 1; j < size; j++) {
      // If condition
      if (arr[i] < arr[j]) {
        // Swap elements
        // Store the value at index j in temp
        int temp = arr[j];
        // Store the value at index i at index j
        arr[j] = arr[i];
        // Store the value of temp at index i
        arr[i] = temp;
      }
    }
  }
}

// Function to print values of an array
void print_array(int arr[], int size) {
  // Traverse array
  for (int i = 0; i < size; i++) {
    // Print value at index i
    cout << arr[i] << " ";
  }
  cout << endl;
}
```

```
// main function
int main() {

  // Initialize size of an array
  int size = 4;
  // Initialize array elements
  int arr[size] = {10, 67, 98, 31};

  cout << "Array before sorting: " << endl;
  // Call print_array function
  print_array(arr, size);

  // Call sort_elements function
  sort_elements(arr, size);

  cout << "Array after sorting: " << endl;
  // Call print_array function
  print_array(arr, size);

  return 0;
}
```

Explanation

To sort the elements in the descending order, we compare the first element of an array with the rest of the elements. If the value of the first element is smaller than the value of any other element in an array, we swap these two values. After the first iteration, the largest value in an array is at the first index of an array. We repeat the same procedure for the rest of the elements in an array.

sort_elements function

The `sort_elements` function takes the array `arr[]` of type `int` and value of type `int` in its input parameters.

We initialize the outer loop from `i=0` to `i = size-1`. As we need to compare, each element will all the elements present after it in the array, initialize the inner loop from `j=i+1` to `j=size-1`. Compare the inner loop (j) element with the outer loop (I). If the element at j is greater, we swap the elements. In order to swap the elements, store the element at `j` in a `temp` variable, store the value at index `i` at `j`, and then store the value of `temp` at index `i`.

Let’s solve some challenges related to two-dimensional arrays in C++.