- Example

Let's explore the versatility of std::array in this example.

WE'LL COVER THE FOLLOWING

Example

One additional value of a `std::array` in comparison to a C array is it that a `std::array` functions similarly to a `std::vector`.

```
// array.cpp

#include <algorithm>
#include <array>
#include <iostream>

int main(){

  std::cout << std::endl;

  // output the array
  std::array <int,8> array1{1,2,3,4,5,6,7,8};
  std::for_each( array1.begin(),array1.end(),[](int v){std::cout << v << " ";});

  std::cout << std::endl;

  // calculate the sum of the array by using a global variable
  int sum = 0;
  std::for_each(array1.begin(), array1.end(),[&sum](int v) { sum += v; });
  std::cout << "sum of array{1,2,3,4,5,6,7,8}: " << sum << std::endl;

  // change each array element to the second power
  std::for_each(array1.begin(), array1.end(),[](int& v) { v=v*v; });
  std::for_each( array1.begin(),array1.end(),[](int v){std::cout << v << " ";});
  std::cout << std::endl;

  std::cout << std::endl;

}
```

Explanation

- In line 13, you can output `array1` with a lambda-function and the range-based for-loop. By using the summation variable sum in line 19, you can ``sum up the elements of the `std::array`.
- The lambda-function in line 23 takes its arguments by reference and maps each element to its square. These arithmetics are not necessarily special, but remember that we are implementing these arithmetics with an `std::array`.

Test your understanding of this example with exercise in the next lesson.