- Solution

Let's review the solution of the previous problem in this lesson.

Solution Review

```
// templateClassTemplateMethods.cpp
#include <type_traits>
#include <algorithm>
#include <iostream>
#include <vector>

template <typename T, int N>
class Array{

public:
  Array()= default;

  template <typename T2>
  Array<T,N>& operator=(const Array<T2, N>& arr){
          static_assert(std::is_convertible<T2, T>::value, "Cannot convert source type to des
    elem.clear();
          elem.insert(elem.begin(), arr.elem.begin(), arr.elem.end());
          return *this;
  }

  int getSize() const;

  std::vector<T> elem;
};

template <typename T, int N>
int Array<T, N>::getSize() const {
  return N;
}


int main(){

  Array<double, 10> doubleArray{};
  Array<int, 10> intArray{};
  doubleArray= intArray;


  Array<std::string, 10> strArray{};
  Array<int, 100> bigIntArray{};

  // doubleArray= strArray;            // ERROR: cannot convert ‘const std::basic_string<char
  // doubleArray= bigIntArray;         // ERROR: no match for ‘operator=’ in ‘doubleArray = b

}
```

Explanation

In the code above, we have created two arrays of `int` and `double` types in lines 34 and 35. We’re copying the data of the integer array to the double array in line 37. If we try to copy the `string` array data to the `double` type in line 42, this gives us an error because of the type mismatch. To observe this, uncomment the line and run to check the error. The function std::is_convertible in line 15 from the **type-traits** library checks if one type can be converted to the other. Of course, it is not possible to convert `string` into `double`.

Let’s move on to template parameters in the next lesson.