

## - Exercise

In this lesson, we'll solve an exercise on template arguments.

### WE'LL COVER THE FOLLOWING ^

- Problem Statement

## Problem Statement #

The class `Matrix` holds its values in the container `Cont`.

- `Cont` should have a default argument `std::vector`.
- Instantiate `myIntVec` and `myDoubleVec` without specifying the container explicitly.

```
#include <initializer_list>
#include <iostream>
#include <list>
#include <vector>

template <typename T, template <typename, typename> class Cont >
class Matrix{
public:
    explicit Matrix(std::initializer_list<T> inList): data(inList){
        for (auto d: data) std::cout << d << " ";
    }
    int getSize() const{
        return data.size();
    }

private:
    Cont<T, std::allocator<T>> data;
};

int main(){

    std::cout << std::endl;

    // Define myIntVec and myDoubleVec with specifying containers explicitly
    // Call getSize() function on it to check the result

    Matrix<std::string, std::list> myStringList{"one", "two", "three", "four"};
```

```
std::cout << std::endl;
std::cout << "myStringList.getSize(): " << myStringList.getSize() << std::endl;

std::cout << std::endl;
}
```



---

We'll look at the solution of this problem in the next lesson.