## Shuffle Ranges

Rearrange the values in a range randomly, using std::random\_shuffle and std::shuffle.

## WE'LL COVER THE FOLLOWING ^

- std::random\_shuffle
- std::shuffle

We can randomly shuffle ranges with std::random\_shuffle and std::shuffle.

## std::random\_shuffle

Randomly shuffles the elements in a range.

```
void random_shuffle(RanIt first, RanIt last)
```

Randomly shuffles the elements in the range, by using the random number generator gen.

```
void random_shuffle(RanIt first, RanIt last, RanNumGen&& gen)
```

## std::shuffle

Randomly shuffles the elements in a range, using the uniform random number generator gen.

```
void shuffle(RanIt first, RanIt last, URNG&& gen)
```

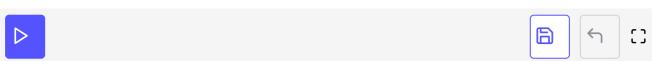
The algorithms need random access iterators. RanNumGen&& gen has to be a callable, taking an argument and returning a value within its arguments.

URNG&& gen has to be a *uniform random number generator*.

```
Prefer std::shuffle
```

Use std::shuffle instead of std::random\_shuffle.std::random\_shuffle has been *deprecated* since C++14 and removed in C++17, because it uses the C function rand internally.

```
#include <algorithm>
                                                                                              G
#include <chrono>
#include <iostream>
#include <random>
#include <vector>
int main(){
  std::cout << std::endl;</pre>
  std::vector<int> vec1{0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
  std::vector<int> vec2(vec1);
  for (auto v: vec1) std::cout << v << " ";</pre>
  std::cout << std::endl;</pre>
  unsigned seed= std::chrono::system_clock::now().time_since_epoch().count();
  std::cout << std::endl;</pre>
  std::random_shuffle(vec1.begin(), vec1.end());
  for (auto v: vec1) std::cout << v << " ";
  std::cout << std::endl;</pre>
  std::shuffle(vec2.begin(), vec2.end(), std::default_random_engine(seed));
  for (auto v: vec2) std::cout << v << " ";</pre>
  std::cout << "\n\n";</pre>
```



Randomly shuffle algorithms

seed initialises the random number generator.

In the next lesson, we'll study ways to make sure each element in our range is unique.