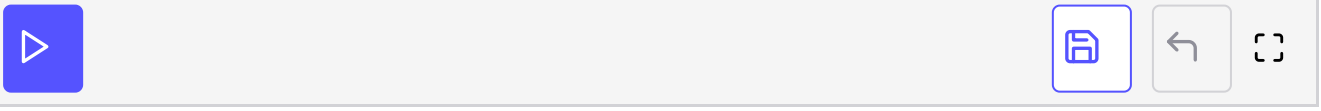
Introduction

This chapter brings us to the various computing algorithms supported by C++17.

The Standard Template Library has a large number of algorithms to work with containers and their elements. As the algorithms are function templates, they are independent of the type of container elements. The glue between the containers and algorithms are the iterators. If your container supports the interface of an STL container, you can apply the algorithms to your container.

```
// algorithm.cpp
#include <iostream>
#include <algorithm>
#include <deque>
#include <iostream>
#include <list>
#include <string>
#include <vector>

template <typename Cont, typename T>
void doTheSame(Cont cont, T t){
  for (const auto c: cont) std::cout << c << " ";
  std::cout << std::endl;
  std::cout << cont.size() << std::endl;
  std::reverse(cont.begin(), cont.end());
  for (const auto c: cont) std::cout << c << " ";
  std::cout << std::endl;
  std::reverse(cont.begin(), cont.end());
  for (const auto c: cont) std::cout << c << " ";
  std::cout << std::endl;
  auto It= std::find(cont.begin(), cont.end(), t);
  std::reverse(It, cont.end());
  for (const auto c: cont) std::cout << c << " ";
}

int main(){
  std::vector<int> myVec{1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
  std::deque<std::string> myDeq({"A", "B", "C", "D", "E", "F", "G", "H", "I"});
  std::list<char> myList({'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'});

  doTheSame(myVec, 5);
  std::cout << "\n\n";
  // 1 2 3 4 5 6 7 8 9 10
  // 10
  // 10 9 8 7 6 5 4 3 2 1
  // 1 2 3 4 5 6 7 8 9 10
  // 1 2 3 4 10 9 8 7 6 5
```

```
  doTheSame(myDeq, "D");
  std::cout << "\n\n";
  // A B C D E F G H I
  // 9
  // I H G F E D C B A
  // A B C D E F G H I
  // A B C I H G F E D

  doTheSame(myList, 'd');
  std::cout << "\n\n";
  // a b c d e f g h
  // 8
  // h g f e d c b a
  // a b c d e f g h
  // a b c h g f e d

  return 0;
}
```



Generic programming with the algorithmn