Introduction

C++ takes reference functionality one step higher by introducing reference wrappers!

A reference wrapper is a copy-constructible and copy-assignable wrapper for a object of type&, which is defined in the header `<functional>`. So you have an object, that behaves like a reference, but can be copied. Contrary to classic references, `std::reference_wrapper` objects support two additional use cases:

- You can use them in containers of the Standard Template Library. `std::vector<std::reference_wrapper<int>> myIntRefVector`
- You can copy instances of classes, which have `std::reference_wrapper` objects. That is in general not possible with references.

To access the reference of a `std::reference_wrapper<int> myInt(1)`, the get method can be used: `myInt.get()`. You can use a reference wrapper to encapsulate and invoke a callable.

```
// referenceWrapperCallable.cpp
#include <iostream>
#include <functional>

void foo(){
  std::cout << "Invoked" << std::endl;
}

int main() {
  typedef void callableUnit();
  std::reference_wrapper<callableUnit> refWrap(foo);

  refWrap(); // Invoked
  return 0;
}
```

Reference wrappers

Now, we will learn how to create reference wrappers.