Thread-Safe Initialization: call_once and once_flag

This lesson gives an overview of thread-safe initialization in the perspective of concurrency in C++.

By using the `std::call_once` function you can register a callable. The `std::once_flag` ensures that only one registered function will be invoked, but we can register additional functions via the same `std::once_flag`. That being said, only one function from that group is called.

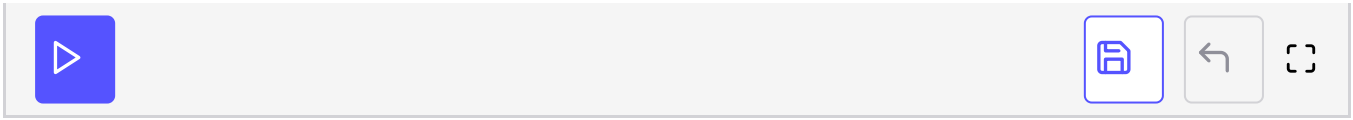
`std::call_once` obeys the following rules:

- Exactly one execution of precisely one of the functions is performed. It is undefined which function will be selected for execution. The selected function runs in the same thread as the `std::call_once` invocation it was passed to.
- No invocation in the group returns before the above-mentioned execution of the selected function completes successfully.
- If the selected function exits via an exception, it is propagated to the caller. Another function is then selected and executed.

This short example demonstrates the application of `std::call_once` and the `std::once_flag`. Both of them are declared in the header `<mutex>`.

```
// callOnce.cpp

#include <iostream>
#include <thread>
#include <mutex>

std::once_flag onceFlag;

void do_once(){
  std::call_once(onceFlag, [](){ std::cout << "Only once." << std::endl; });
}

int main(){

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

```
  std::thread t1(do_once);
  std::thread t2(do_once);
  std::thread t3(do_once);
  std::thread t4(do_once);

  t1.join();
  t2.join();
  t3.join();
  t4.join();

  std::cout << std::endl;

}
```



The program starts with four threads (lines 17 - 20); each of them invokes `do_once`. The expected result is that the string “only once” is displayed only once.

The famous singleton pattern guarantees that only one instance of an object will be created. This is a challenging task in multithreading environments, but `std::call_once` and `std::once_flag` make the job a piece of cake. Now, the singleton is initialized in a thread-safe way.

In the next lesson, we’ll solve exercise to better understand the concept.