Creation of Threads

This lesson gives an introduction on how to create threads in C++ using callable units such as functions and lambda functions.

we'll cover the following ^
• Output

To launch a thread in C++, you have to include the <thread> header.

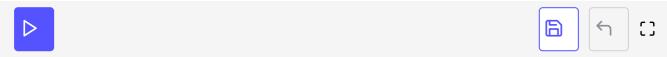
- A thread std::thread represents an executable unit. This executable unit, which the thread immediately starts, gets its work package as a callable unit.
- A callable unit is an entity that behaves like a function. Of course, it can be a function, but also a function object or a lambda function.

For example,

```
// createThread.cpp
#include <iostream>
#include <thread>
void helloFunction(){
  std::cout << "Hello from a function." << std::endl;</pre>
}
class HelloFunctionObject{
  public:
    void operator()() const {
      std::cout << "Hello from a function object." << std::endl;</pre>
    }
};
int main(){
  std::cout << std::endl;</pre>
  std::thread t1(helloFunction);
  HelloFunctionObject helloFunctionObject;
```

```
std::thread t2(helloFunctionObject);
std::thread t3([]{std::cout << "Hello from a lambda." << std::endl;});

t1.join();
t2.join();
t3.join();
std::cout << std::endl;
};</pre>
```



All three threads (t1, t2, and t3) write their messages to the console. The work package of thread t2 is a function object (lines 10 - 15), and the work package of thread t3 is a lambda function (line 26). In lines 28 - 30 the main thread is waiting until its children are done.

Output

The three threads are executed in an arbitrary order; even the three output operations can interleave. The creator of the child - the main thread in our case - is responsible for the lifetime of the child.