User Defined Atomics

This lesson gives an overview of user-defined atomics used from the perspective of concurrency in C++.

There are a lot of deep restrictions on a user-defined type `MyType` if you use it for an atomic type `std::atomic<MyType>`. These restrictions are on the type `MyType`, but also on the operations that `std::atomic<MyType>` can perform.

Here are the restrictions for `MyType` to become an atomic type:

- The copy assignment operator for `MyType` (all base classes of `MyType` and all non-static members of `MyType`) must be trivial. This means that you must not define the copy assignment operator but request it by `default` from the compiler.
- `MyType` must not have virtual methods or virtual base classes.
- `MyType` must be bitwise comparable so that the C functions memcpy or memcmp can be applied.

> **Check the type properties at compile time**
>
> The type properties on `MyType` can be checked at compile time, by using the following functions: `std::is_trivially_copy_constructible`, `std::is_polymorphic` and `std::is_trivial`. All these functions are part of the very powerful type-traits library.

The user-defined atomic type `std::atomic<MyType>` supports only a limited interface.