Introduction

The type traits library helps us optimize our code. This section will cover the library in depth.

The type traits library enables you, to check, to compare and to modify types at compile time. So, there is no overhead on the runtime of your program. There are two reasons for using the type traits library: Optimization and Correctness. **Optimization**, because the introspection capabilities of the type traits library make it possible to choose the faster code automatically. **Correctness**, because you can specify requirements for your code, which is checked at compile time.

> 🗝️ **The type traits library and static_assert are a powerful pair**
> The type traits library and the function `static_assert` are a powerful pair. On one side, the functions of the type traits library provide the type information at compile time. On the other side, the `static_assert` function checks the given information at compile time. All this happens transparently to the runtime of the program:
>
> ```
> #include <type_traits>
> template <typename T> T fac(T a){
>     static_assert(std::is_integral<T>::value, "T not integral");
>     //...
> }
> fac(10);
> fac(10.1); // with T= double; T not integral
> ```
>
> The GCC compiler quits the function invocation `fac(10.1)`. The message at compile is that T is of type double and therefore no integral type.

Now that you've learned the importance of the time traits library, let's learn how to use it.