Utilities

To use C++ to its full potential, we must use the multitude of utilities it provides.

WE'LL COVER THE FOLLOWING

As C++11 has a lot of libraries, it is often not so easy to find the convenient one for each use case.

Utilities are libraries which have a general focus and therefore can be applied in many contexts.

Calculating the Minimum and Maximum

Examples of utilities are functions to calculate the minimum or maximum of values or functions to swap or move values.

Functional Programming

Other utilities are std::function and std::bind. With `std::bind` you can easily create new functions from existing ones. In order to bind them to a variable and invoke them later, you have `std::function`.

Pairs

With `std::pair` and it's generalization `std::tuple` you can create heterogeneous pairs and tuples of arbitrary length.

Reference Wrappers

The reference wrappers `std::ref` and `std::cref` are pretty handy. One can use them to create a reference wrapper for a variable, which for `std::cref` is const.

Smart Pointers

Of course, the highlights of the utilities are the smart pointers. They allow explicit automatic memory management in C++. You can model the concept of explicit ownership with `std::unique_ptr` and model shared ownership with `std::shared_ptr`. `std::shared_ptr` uses reference counting for taking care of its resource. The third one, `std::weak_ptr`, helps to break the cyclic dependencies among `std::shared_ptr`s, the classic problem of reference counting.

Type Traits

The type traits library is used to check, compare and manipulate type information at compile time.

Multithreading

The time library is an important addition of the new multithreading capabilities of C++. But it is also quite handy to make performance measurements.

Values of Datatypes

With std::any, std::optional, and std::variant, we get with C++17 three special datatypes that can have any, an optional value, or a variant of values respectively.

Now, let’s talk about the components of the C++ Standard Library.