constexpr Additions to the Standard Library

C++17 brings a diverse range of new methods to the constexpr class. We'll talk about them below.

With this enhancement you can work with iterators, `std::array`, range-based for loops in `constexpr` contexts.

> The main referencing paper is P0031 - Proposal to Add Constexpr Modifiers to reverse_iterator, move_iterator, array, and Range Access.

Have a look at a basic example that shows a basic implementation of the accumulate algorithm as `constexpr` (C++11/14/17 version of `std::accumulate` is not `constexpr`):

```
#include <array>
#include <iostream>
template<typename Range, typename Func, typename T>
constexpr T SimpleAccumulate(const Range& range, Func func, T init) {
  for (auto &&obj : range) // begin/end are constexpr
  {
    init += func(obj);
  }
  return init;
}
constexpr int square(int i) { return i*i; }
int main() {
  constexpr std::array arr{ 1, 2, 3 }; // class deduction...
  // with constexpr lambda
  static_assert(SimpleAccumulate(arr, [](int i) constexpr { return i * i;}, 0) == 14);
  // with constexpr function
  static_assert(SimpleAccumulate(arr, &square, 0) == 14);
  std::cout << arr[0] << '\n';
  return arr[0];
}
```

C++14 compilers would not compile the above example, but it's now possible with C++17 support.

There are several features used in this code:

- `SimpleAccumulate` is a `constexpr` template function and uses range access (hidden in range based for loop) to iterate over the input range.
- `arr` is deduced as `std::array<3, int>` - class template deduction works here.
- the code uses a `constexpr` lambda.
- `static_assert` without any message.
- `std::begin()` and `std::end()` are also `constexpr` since C++17

You can also see another example of `constexpr` additions in Chapter about General Language Features: Constexpr Lambda.

Each C++ Standard allows more and more code to be `constexpr`. In C++17 we can start using basic containers in constant expressions. In C++20 we'll get more standard algorithms that are declared as `constexpr`.

> **Extra Info:** The main referencing paper is P0031 - Proposal to Add Constexpr Modifiers to reverse_iterator, move_iterator, array and Range Access

---

C++ has also added a way of locking a variadic number of mutexes simultaneously. Find out more in the next lesson.