dependency, the aforementioned free functions in the `<iterator>` header were also added to many other headers, including those for all of the standard containers.

As subsequent editions of the Standard have added to the set of related, free function overloads (for example, `empty`, added in C++17), those additional overloads have been added to multiple headers so that idiomatic generic code does not need to include additional headers. However, due to the freestanding requirement, none of those additional overloads are available through the `<initializer_list>` header.

## See Also

- “Braced Init” (§2.1, p. 215) provides further details regarding object initialization and construction using braced lists and `std::initializer_list`s.

## Further Reading

- An overview of the myriad problems associated with initialization is presented in **stroustrup05a**.

- The original proposal to achieve a uniform initialization syntax based on initializer lists can be found in **stroustrup05b**.

- Andrzej Krzemieński details why and how `std::initializer_list` can incur insidious runtime overhead due to excessive copying in **krzemienski16**.

## Appendix

### A brief history of user customization to support range-based `for`

The original incarnation of the range-based **`for`** loop was built around the proposed **`concepts`** language feature. The range argument to the **`for`** loop would satisfy the range `concept`, which the user could customize with a `concept_map` to adapt third-party libraries that did not have `begin` and `end` member functions.

When concepts ~~were~~ pulled after the first ISO ballot, the committee did not want to lose range-based **`for`**, so a new scheme was invented: The compiler would look for `begin` and `end` free functions. The Standard Library would provide the primary template for these functions, and **`#include`** `<iterator>` would be required for the range-based **`for`** loop to work, much like **`#include`** `<typeinfo>` is required to enable the **`typeid`** operator. Suggestions were made — and rejected — that the core feature should look for the member functions (just like the template) before looking for the free functions as the last resort. To make the standard library easy to use, the primary template for the `begin` and `end` functions was added to every standard container header, to `<regex>`, to `<string>`, and would be added on an ongoing basis to any new headers for types representing a range.