Remove Deprecated `std::allocator` Typedef From C++26

Document #:	P2868R0
Date:	2023-05-15
Project:	Programming Language C++
Audience:	Library Evolution Incubator
Revises:	N/A

Reply-to:	Alisdair Meredith <ameredith1@bloomberg.net>

Contents

1 Abstract

The Standard Library `allocator` class contains a deprecated `typedef` member than can cause problems for derived classes. This paper proposes removing that member from the C++ Standard Library.

2 Revision history

2.1 R0: Varna 2023

Initial draft of the paper.

3 Introduction

At the start of the C++23 cycle, [P2139R2] tried to review each deprecated feature of C++ to see which we would benefit from actively removing and which might now be better undeprecated. Consolidating all this analysis into one place was intended to ease the (L)EWG review process but in return gave the author so much feedback that the next revision of the paper was not completed.

For the C++26 cycle, a much shorter paper, [P2863R0], will track the overall analysis, but for features that the author wants to actively progress, a distinct paper will decouple progress from the larger paper so that the delays on a single feature do not hold up progress on all.

This paper takes up the deprecated 'is_always_equal' member typedef in `std::allocator`, D.17 [depr.default.allocator].

4 Analysis

The Standard Library `allocator` class has a typedef member that could be synthesized from the primary `allocator_traits` template, and was deprecated in C++20 by [#LWG3170]. When `std::allocator` provides this member directly, any classes that derive from it will not synthesize this type name correctly, but use the `true_type` value provided directly by `std::allocator`. If the derived allocator type is not empty, its value for this trait will not match the expected default behavior, forcing the allocator author (if they are aware) to add their own typedef that should not be needed to restore the default behavior; typically, this leads to subtle bugs.

While this is a small corner for misuse, the concern is embarrassing to explain, and the Standard Library allocator is a common example folks will follow when trying to write their first allocators. Hence, this paper recommends the removal of this deprecated typedef for C++26.

5 Proposal

Remove the deprecated typedef `std::allocator<T>::is_always_equal` from C++26.

6 Wording

All wording is relative to [N4944], the latest working draft at the time of writing.

D.17 [depr.default.allocator] The default allocator

1 The following member is defined in addition to those specified in 20.2.10 [default.allocator]:

```
namespace std {
  template<class T> class allocator {
  public:
    using is_always_equal = true_type;
  };
}
```

7 Acknowledgements

Thanks to Michael Parks for the pandoc-based framework used to transform this document's source from Markdown.

8 References

[N4944] Thomas Köppe. 2023-03-22. Working Draft, Standard for Programming Language C++.
https://wg21.link/n4944

[P2139R2] Alisdair Meredith. 2020-07-15. Reviewing Deprecated Facilities of C++20 for C++23.
https://wg21.link/p2139r2