`assert` Should Be A Keyword In C++26

Supporting standard C++23 macros in module `std`

Document #:	P2884R0
Date:	2023-05-15
Project:	Programming Language C++
Audience:	Evolution Incubator
Revises:	N/A
Reply-to:	Alisdair Meredith <ameredith1@bloomberg.net>

Contents

1 Abstract

Macros cannot be exported from a C++ module. This proposal claims that the C++23 Standard Library macro `assert` would be better specified as a keyword, removing it from the set of library features that are *not* made available by importing the standard library module `std`.

2 Revision history

2.1 R0: Varna 2023

Initial draft of the paper.

3 Introduction

C++23 introduced the standard library module `std` that is intended to import the whole standard library, see [P2465R3]. However, this module leaves a gap for all the library facilities that are specified as macros. Paper [P2654R0] is tracking progress of a set of papers that attempt to support all language facilities that are currently specified with the aid of macros with that single import. This paper addresses the macro `assert`.

Any serious discussion of `assert` belongs in SG21, Contract Programming. However, it is worth sufficient discussion here to determine whether this is a problem we want to see solved directly, and send that feedback to SG21.

Ultimately, we expect SG21 may prefer to deprecate `<cassert>` in favor of the new Contracts facility, rather than adopt `assert` as a keyword, but the timing of SG21 discussions on syntax mean this paper must be submitted for initial consideration now.

4 Proposal

This paper proposes reserving the token `assert` as a keyword for an operator. The proposal satisfies the need to enable existing assertion in modern C++ code that does not rely on `#include` to import libraries. We would then explicitly *not* define `assert` as a macro in `<cassert>` and `<assert.h>`, much like we did not define `bool`, `true`, and `false` as macros in `<cstdbool>`.

In addition to resolving the macro import issue, adopting `assert` as an operator would resolve issues surrounding C macros not recognizing C++ matched brackets syntax for brackets other than parens, such as <>, {}, and [], which are significantly more common in C++ source than in C.

Note that any attempt to turn assert into an operator is going to run into issues around build environments enabling and disabling its use, interaction with users defining and undefining `NDEBUG` in existing code, and users relying on `assert` expanding to nothing and so containing non-compiling code in certain uses. It is likely that a C++ `assert` operator would want to integrate with the configurable violation handling under discussion in SG21, and so we would recommend moving any non-superficial discussion into that study group.

4.1 Simple Example

```
import std;
#include <cassert>  // not an imoportable heder unit

int main() {
  assert( std::is_same_v<int, int> );          // too many macro arguments
  assert((std::is_same_v<int, int>));          // OK

  assert( std::vector{1,2,3}.size() == 3 );    // too many macro arguments
  assert((std::vector{1,2,3}.size() == 3));    // OK

  int x = 0;
  int y = 0;
  assert( [x,y]{ return test(x,y);}() );       // too many macro arguments
  assert(([x,y]{ return test(x,y);}()));       // OK, lambda expression
}
```

5 Wording

Wording is deferring until this proposal makes sufficient progress that the full design is clear.

6 Acknowledgements

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

7 References

[P2465R3] Stephan T. Lavavej, Gabriel Dos Reis, Bjarne Stroustrup, Jonathan Wakely. 2022-03-11. Standard Library Modules std and std.compat.
https://wg21.link/p2465r3

[P2654R0] Alisdair Meredith. 2023-05-15. Macros And Standard Library Modules.
https://wg21.link/p2654r0