P3087R1

Make direct-initialization for enumeration types at least as permissive as direct-list-initialization

Tokyo 2024

東京

Author: Jan Schultke
Presenter: Jan Schultke
Audience: EWG
Project: ISO/IEC 14882 Programming Languages — C++, ISO/IEC JTC1/SC22/WG21

Contents

1. Introduction

Recent history of enum initialization

- P0138R2: Construction Rules for `enum class` Values (C++17)
- P0960R3: Allow initializing aggregates from a parenthesized list of values (C++20)
- *“What about parenthesized initialization of enumerations? “*

Status quo

```
enum class E {};
E a{0};    // OK, direct-list-initialization
E{0};      // OK, direct-list-initialization
E(0);      // OK, function-style cast
E b = 0;   // error: cannot convert ‘int‘ to ‘E‘ in initialization
E c(0);    // error: cannot convert ‘int‘ to ‘E‘ in initialization
new E(0); // error: cannot convert ‘int‘ to ‘E‘ in initialization
```

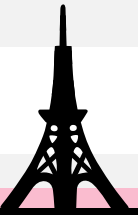
2. Motivation

Teachability issues

- Intuition violation:
  - *“Direct-list-init is direct-init, but without narrowing conversions. “*
- Enumeration initialization is inconsistent with …
  - class type initialization
  - arithmetic type initialization

Don‘t repeat yourself

```
std::vector<std::byte> bytes;
bytes.emplace_back(0xff);                // ill-formed
bytes.emplace_back(std::byte{0xff}); // OK, but DRY violation
```



3. Proposal

Wording

In [dcl.init.general], define direct-initialization for enumerations exactly like direct-list-initialization, except that narrowing conversions don‘t matter.

- (As a consequence, direct-init is defined in terms of `static_cast`.)

Editorial changes to [dcl.init.list] for enumerations:

- itemize (prose to bullets)
- define `T{...}` in terms of `static_cast<T>(...)` instead of `T(...)`

TODO: Feature-detection macro? (Topic for SG10/CWG)

4. Impact/Design

Questions

- *“Is existing code affected? “*
  - Only expression testing (SFINAE, etc.)
- *“What about enums without fixed underlying types?“*
  - No changes, direct-init remains ill-formed (consistent with direct-list-init)
- *“Shold implicit conversions to enums be allowed in general?“*
  - No!

Aggressive permissiveness

```
std::byte b(1000.f);   // OK ?!
```

4. Design - Alternatives

Comparison

Option	`std::byte b(0);`	`std::byte b(-1);`	`std::byte b(0.f);`
Leave everything as is (status quo)	✗	✗	✗
Exactly like list-initialization	✓	✗	✗
Restrict non-integral conversions	✓	✓	✗
Proposed (permissive)	✓	✓	✓

- Only proposed design is consistent with non-list-direct-initialization at large
- Only proposed design is symmetrical with explicit constructors
- More restrictive semantics can be achieved with warnings (QoI)

References

Jan Schultke; **P3087:** Make direct-initialization for enumeration types at least as permissive as direct-list-initialization (latest revision)
https://eisenwave.github.io/cpp-proposals/enum-direct-init.html

Gabriel Dos Reis; **P0138R2:** Construction Rules for enum class Values
https://www.open-std.org/jtc1/sc22/wg21/docs/papers/2016/p0138r2.pdf

Ville Voutilainen, Thomas Köppe; **P0960R3:** Allow initializing aggregates from a parenthesized list of values
https://www.open-std.org/jtc1/sc22/wg21/docs/papers/2019/p0960r3.html