Allowing trailing commas in *ctor-initializer*

Document #:	P2067R0
Date:	January 13, 2020
Project:	Programming Language C++ EWG Incubator

Reply-to: Marc Mutz <marc.mutz@kdab.com>

Abstract

C++11 inherited trailing commas in *enum-specifier* from C99, and trailing commas in *braced-init-list* from even earlier C versions.

C++ adds a list that does not allow trailing commas: *ctor-initalizer*.

We propose to allow a trailing comma, such that the following becomes valid C++:

```
Type() :
    m_one(~~~), // trailing-comma style
    m_two(~~~),
{}
```


1 Motivation and Scope

The author is unaware of the reasons the C committee decided to allow trailing commas in *enum-specifier*, but all reasons likely apply to *ctor-initalizer*, too, just more so, as for *enum-specifier*, one can always invent an artificial end enumerator to keep "production values" trailing-comma-seprarated:

```
enum E {
  One,
  Two,

  NumE
};
```

This option does not exist for *ctor-initalizer*.

In our experience, the users of trailing-comma lists aim to avoid touching unrelated lines of code when they perform changes using a version control system.

Consider the following example, where a new data member is being added at the end:

```
    Type() :
        m_one(~~~),
-       m_two(~~~)
+       m__two(~~~),
```

```
+       m_three(~~~)
    {}
```

The `m_two` line is changed, but not functionally. This is frowned upon when using VCSs, because it disturbs the history of changes, which, in widely-used VCSs such as git, svn, etc., is recorded line-by-line. In reports like `git blame`, the `m_two` initialisation will appear as if it has been changed when `m_three` was added, and a deeper inquiry (looking at the exact commit) is needed to find that just a comma was added.

Same diff with this proposal accepted:

```
    Type() :
        m_one(~~~),
        m_two(~~~),
+       m_three(~~~),
    {}
```

As a work-around, the following style has been adopted, at least in parts of Qt, and probably elsewhere:

```
    Type()
      : m_one(~~~)
      , m_two(~~~) // leading-comma style
    {}
```

This style, however, still treats one member special: If the first member is removed, the diff still touches the unrelated line initialising `m_two`:

```
    Type()
-     : m_one(~~~)
-     , m_two(~~~)
+     : m_two(~~~)
    {}
```

The reason why this style is still preferred, even though it doesn't solve all of the problem, is that removal of the first data member is statistically less likely than adding a new data member at the end.

The matter was previously discussed in 2015 on the `std-proposals` mailing-list[1]. Arthur O'Dwyer identified the changes required to the standard[2], and Ville developed a patch implementing this for GCC[3], but it appears that no proposal was put forth in the aftermath.

This aims to be that missing proposal.

While some participants of the the `std-proposals` 2015 discussion questioned whether the work involved in adapting all C++ parsers was worth the gain, the authors believe that the recurring nature of this topic, as well as the avidity with which users attempt to work around the issue, warrants a settlement by allowing trailing commas.

2 Impact on the Standard

Minimal. The syntax we propose to make valid was ill-formed before.

3 Proposed Wording

3.1 Changes to [N4820]

In [**class.base.init**]/**1**, as well as in [**gram.class**], change the production for *ctor-initalizer* as indicated:

ctor-initalizer:
: *mem-initializer-list* *,opt*

3.2 Feature Macro

We propose to use a new macro, `__cpp_ctor_trailing_commas`, to indicate an implementation's support for this feature.

4 Design Decisions

4.1 Extension to other lists

One of the points that was raised in the `std-proposals` 2015 discussion was whether to allow trailing commas for all lists, incl. e.g. function arguments. We welcome such discussion, but deem it outside the scope of this proposal for at least its first iteration, mainly because there is no general "list" grammar production in the standard to which a change could be uniformly applied, and other types of lists have not seen the same kinds effort being spent on work-arounds that *ctor-initalizer* has "enjoyed".

5 Acknowledgements

Arthur O'Dwyer provided the wording on which this proposal is based.[2]

6 References

[1] https://groups.google.com/a/isocpp.org/d/msg/std-proposals/I8N_75J9ZB8/9Mm27qvbYZkJ

[2] Arthur O'Dwyer
Permit mem-initializer-list to end with a trailing comma
https://github.com/cplusplus/draft/commit/5417003045bad50705847f3d20ff72d78aeec32a

[3] Ville Voutilainen
GCC Patch "Allow trailing comma in a mem-initializer-list"
https://github.com/villevoutilainen/gcc/commit/3e7dc8cc67ef5a3c302ff15c90b0c7cbc56760e9

[N4820] Richard Smith (editor)
Working Draft: Standard for Programming Language C++
http://wg21.link/N4820