Allow repeating contract annotations on non-first declarations

Timur Doumler (papers@timur.audio)

Document #: P3066R0
Date: 2023-12-04
Project: Programming Language C++
Audience: SG21

Abstract

In this paper, we propose to re-introduce the allowance to repeat the list of precondition and postcondition specifiers of a function on a non-first declaration to the Contracts MVP, as long as it is the same list. Since we now have a definition of what it means for a list of contract annotations to be *the same*, and we need that definition anyway to reason about multiple first definitions of a function in different translation units, there is no longer a reason to keep the restriction that contract annotations can appear on first declarations only.

1 Motivation

The Contracts MVP ([P2900R2]), which is currently in development and targeting C++26 (as per SG21's roadmap in [P2695R1]), only allows precondition and postcondition specifiers on the first declaration of a function in a translation unit.

This can be surprising to users. Normally it is expected to repeat the same list of function qualifiers and specifiers on subsequent declarations (including definitions), and it is very common to have a declaration in a header file and a definition in a source file. However, according to the rules in the Contracts MVP, the user would have to omit `pre` and `post` on the definition, otherwise the program is ill-formed.

This is especially unfortunate since some preconditions or postconditions might be directly relevant in the function body, as they contain predicates on entities used there, and it would greatly add comprehension of the code to have them repeated there:

```
// Widget.h
struct Widget
{
  int f() const noexcept pre(i > 0);

  // much further below:
private:
  int i;
};

// Widget.cpp
int Widget::f() const noexcept pre(i > 0)  // error! cannot repeat pre here
{
  return i * i; // using i here!
}
```

C++20 Contracts ([P0542R5]) allowed to repeat the list of precondition and postcondition specifiers on a non-first declaration, provided that the list was the same. The same allowance exists in an earlier version ([P2388R4]) of the Contracts MVP, which contains the following wording:

> The first declaration of a function D shall specify all contract annotations (if any) of the function. Other declarations of the function reachable from D shall either specify no contract annotations or the same list of contract annotations.

This specification was successfully implemented in GCC, where the above code example compiles and works as expected (https://godbolt.org/z/c4f1oqd4e).

The allowance to repeat contract annotations on non-first declarations was dropped from the Contracts MVP in [P2521R5], which says:

> if a given function f has declared preconditions and postconditions, they shall be visible in the first declaration of f in a translation unit (TU): otherwise the program is ill-formed. Subsequent declarations must omit contract annotations.

This change is motivated as follows:

> The reason for this restriction is implementability issues, similar to those for default function arguments. This decision departs from [P0542R5], which allowed putting odr-identical contract annotations on redeclarations. The reason we do not allow contract annotations on redeclarations is because this way we avoid the reason to define the notion of being "odr-identical".

However, we later discovered that we need to define sameness of two contract annotations anyhow, because we need to specify when a program is well-formed that contains multiple first declarations of the same function in different translation units. The need to define sameness of contract annotations is captured in [P2896R0] as an open question that must be resolved before we can ship the Contracts MVP. [P2932R2] section 3.6 proposes a definition that works and does not seem to have any implementability issues:

> A contract-checking annotation (CCA), $c1$, on a function declaration, $d1$, is the same as a CCA, $c2$, on a function declaration, $d2$, if their predicates, $p1$ and $p2$, would satisfy the one-definition rule if placed in function definitions on the declarations $d1$ and $d2$, respectively, except for the renaming of parameters, return value identifiers, and template parameters.

If this definition gets accepted into the Contracts MVP, then the motivation for not allowing to repeat the list of contract annotations on a non-first declaration evaporates. Nevertheless, [P2932R2] proposal 6.1 proposes to keep the restriction of allowing contract annotations on first declarations only, without providing any motivation beyond wanting to keep the MVP "minimal". However, dropping this restriction and reverting to the previous allowance is an opportunity to give tangible benefit to users of Contracts at no cost, as this allowance had already been in the Contracts MVP for a long time and we already have both wording and implementation experience with it.

2 Proposal

In case SG21 approves [P2932R2] proposal 6.2, "the list of contract annotations on all first declarations (in all translation units) for a function shall be *the same*, no diagnostic required" — with the definition of *the same* given in [P2932R2] — we propose to re-introduce the allowance in [P2388R4] that non-first declarations shall either specify no contract annotations or *the same* list of contract annotations. The status quo in the Contacts MVP is that they shall specify no contract annotations.

However, we are *not* proposing an allowance to omit the list of contract annotations from the first declaration. This would be a new feature that has not yet been studied in detail, and that we do not have implementation experience with, so it should only be considered post-MVP.

References

[P0542R5] G. Dos Reis, J. D. Garcia, J. Lakos, A. Meredith, N. Myers, and B. Stroustrup. Support for contract based programming in C++. http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2018/p0542r5.html, 2018-06-08.

[P2388R4] Andrzej Krzemieński and Gašper Ažman. Minimum Contract Support: either *No_eval* or *Eval_and_abort* contracts. http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2021/p2388r4.html, 2021-11-15.

[P2521R5] Andrzej Krzemieński, Gašper Ažman, Joshua Berne, Bronek Kozicki, Ryan McDougall, and Caleb Sunstrum. Contract support – Record of SG21 consensus. https://www.open-std.org/jtc1/sc22/wg21/docs/papers/2023/p2521r5.html, 2023-08-15.

[P2695R1] Timur Doumler and John Spicer. A proposed plan for Contracts in C++. https://www.open-std.org/jtc1/sc22/wg21/docs/papers/2023/p2695r1.pdf, 2023-02-09.

[P2896R0] Timur Doumler. Outstanding design questions for the Contracts MVP. https://wg21.link/p2896r0, 2023-08-22.

[P2900R2] Joshua Berne, Timur Doumler, and Andrzej Krzemieński. Contracts for C++. https://wg21.link/p2900r2, 2023-11-11.

[P2932R2] Joshua Berne. A Principled Approach to Open Design Questions for Contracts. https://wg21.link/p2932r2, 2023-11-14.