Doc. No.: P2829R0
Project: Programming Language - C++ (WG21)
Audience: SG21 Contracts
Date: 2023-02-27
Author: Andrew Tomazos <andrewtomazos@gmail.com>

Proposal of Contracts Supporting Const-On-Definition Style

Summary

We propose that any (non-reference) function parameters used in MVP postconditions: (a) must be declared explicitly const in function definitions (by the programmer); and (b) they are implicitly declared const in other function declarations (by the implementation).

This proposes amending the first sentence of P2521R2/3.12 as follows: “If a non-reference function parameter is named in a postcondition, that parameter shall be declared const in every ~~declaration~~ definition of the function, and const is added when forming their type from every other non-const declaration.”

Example

```
// declaration
void f(int x) // <-- x is implicitly const
    POST(is_const_v<decltype(x)>); // true

// definition
void f(const int x) // <-- x is explicitly const
{
    /*...*/
}
```

Motivation

We feel this both implements SG21s decision on issue P2521R1/4.3 and supports the const-on-definition style.

What is the const-on-definition style?

There is a rule in C++ that reads along the lines of “After [analyzing] the list of parameter types [of a function type], any [const] modifying a parameter type are deleted when forming the function type.” ([dcl.fct]/5)

What this means is that the following declares and defines, respectively, the same function:

```
void f(int x);

void f(const int x) { /*...*/ }
```

So if you want to make a parameter const, you only have to do it on the definition. You can leave it off the declaration. Whether or not parameters are const are not part of the “interface” of a function.

We call this the *const-on-definition* style.

We suspect (but cannot prove empirically because it’s hard to search for) that this feature has significant use in the field. The present-day desire to mark some parameters const comes from const-correctness (see “const correctness” in the ISOCPP C++ FAQ: https://isocpp.org/wiki/faq/const-correctness), and in such cases, the desire to leave const off the non-definition declaration comes from not wanting to clutter the headers with unnecessary and superfluous information. Notice that whether or not the callee modifies its private copy of the argument object (the parameter object) is usually not of interest to the caller. There are exceptions to that, but they are exceedingly rare.

With SG21s decision on issue P2521R2/4.3 of the contracts working paper, it was decided that in order to avoid the pitfall of function parameters used in postconditions being modified during function execution from their initial value (the argument value), such parameters would need to be const.

We propose that contracts should support this const-on-definition style for parameters used in postconditions.

Concretely, the way we propose doing that is as follows:

For any function **parameters** that are used in a postcondition of any of the functions declarations:

- In non-definition **function declarations** they are **implicitly declared const** by the implementation

- In **function definitions** they shall have been **explicitly declared const** by the programmer

References

P2521R1/4.3 Contract support — Working Paper
https://www.open-std.org/jtc1/sc22/wg21/docs/papers/2022/p2521r1.html

P2521R2/3.12 Contract support — Working Paper
https://www.open-std.org/jtc1/sc22/wg21/docs/papers/2022/p2521r2.html