P3498R0 2024-11-13 Reply-To: gdr@microsoft.com
Audience: EWG, SG23

Stop the Bleeding but, First, Do No Harm

Gabriel Dos Reis, Thomas Wise, Zachary Henkel
Microsoft

Microsoft, as a software company, consumes and produces a wide variety of business software written in C++ that runs on a wide range of platforms and environments. It has its own C++ compiler on Windows platforms, but it also uses non-Microsoft C++ compilers for cross-platform products. As such, Microsoft considers it essential that ISO C++ provide standard features and guarantees that address the "memory safety" concerns expressed by various regulatory bodies (Cybersecurity & Infrastructure Security Agency, 2024). Those safety issues currently look like existential threats to the continued vibrancy and success of C++ in the application domains where it has been deployed over the last four decades, and beyond. This is not a Microsoft-only problem. Any software provider that uses C++ and transitively software that depends on C++ - subject to various safety regulations – faces or will face the same issues. That cluster is growing and will keep growing. The C++ standards committee, the steward of C++ evolution, must act with a sense of urgency and purpose.

We strongly encourage WG21 to address safety issues as a high priority: ***Stop the bleeding***. As WG21 tends to that task, we also encourage it to remember that the various judgments and recommendations made against C++ are based on the quality of existing software written in C++ as it is specified and implemented today. Those billions of lines of code need to continue to function while their quality is shown to substantially improve thanks to the evolution of C++. Gradual adoption of safety features into existing code bases is essential. Revolutionary language facilities that only address future yet-to-be-written code, or that are economically non-viable to deploy, cause active harm to the C++ ecosystem because of delays and fragmentation of the code bases. Consequently, ***WG21 must avoid doing harm*** in its endeavor to improve the safety posture of ISO C++. We need to coalesce around a single framework and a set of standard features to avoid loss of portability. For example, we need a standard way of requiring enforcement of range checking.

Concrete suggestions:

- During the C++20 development cycle, Microsoft proposed `std::span` (MacIntosh, 2018) with bound checking of the indexing operator. That guarantee – the primary raison d'être of that abstraction facility – was unfortunately removed as an attempt to find consensus. In retrospect, Microsoft considers that decision counter-productive and an example for detractors to point at as why WG21's work does not prioritize safety. We suggest that WG21 finds a way to restore the bound checking guarantee, as part of standard C++.
- For its safety-focused features/solutions, WG21 should consider how practical a solution is for adoption at scale for existing C++ codebase. We consider that the proposals along the lines of C++ Profiles are promising. In particular, they address a wide range of safety concerns and aim to minimize annotations. One of the lessons learned from Microsoft SAL

(Microsoft Corp., 2021) deployment and more than 25 years of field experience is that proliferating annotations are often perceived by developers as a hindrance (when they get them right). Techniques and tools that rely on information already available in C++'s semantics should be favored – they scale better while being less intrusive than pervasive annotation.

- Microsoft is a proponent for a contract facility in C++ to increase practical safety at scale. However, the current proposal, as detailed in P2900 (Berne, 2024), severely misses the mark regarding safety and code analysis. As such, rushing it into C++26 will constitute active harm. It needs further work before moving to the next stage of standardization.

WG21 needs to effectively deliver the safety guarantees that regulatory bodies and the industry require. The means of achieving various forms of safety must be standard to preserve portability. They must prioritize contemporary C++ semantics and development techniques to the greatest extent possible while fulfilling these guarantees. That must be a coherent framework, rather than an ad hoc collection of features. The solution must be manageable at scale by any C++ codebase aiming to enhance its safety outlook. WG21 should unite to deliver this in a timely fashion. Critical and important parts of the C++ industry are not patient. "Safety" has become a distinguishing lever for contemporary programming languages in domains relevant to C++. Failures to address safety concerns with meaningful shared solutions will lead to greater fragmentation of the software industry, not just the C++ community.

References

Berne, J. (2024, October 12). *Contracts for C++*. https://www.open-std.org/jtc1/sc22/wg21/docs/papers/2024/p2900r10.pdf

Cybersecurity & Infrastructure Security Agency. (2024, October 16). *Product Security Bad Practices*. Retrieved from CISA: https://www.cisa.gov/resources-tools/resources/product-security-bad-practices

MacIntosh, N. (2018, Mars 16). *span: bounds-safe views for sequence of objects*. https://www.open-std.org/jtc1/sc22/wg21/docs/papers/2018/p0122r7.pdf

Microsoft Corp. (2021, August 02). *Using SAL Annotations to Reduce C/C++ Code Defects*. https://learn.microsoft.com/en-us/cpp/code-quality/using-sal-annotations-to-reduce-c-cpp-code-defects?view=msvc-170