6000CEM Individual Project Preparation - Proposal

Fred Cook - 8463955

November 2021

Contents

1 Project title and research question

The project of my title is *"Implementing the Rust Borrow Checker in C++"*

The research question for my project is: *"Is it possible to recreate the behaviour driven by the "Rust borrow checker" in C++ as a proof of concept?"*

2 Project topic

The areas of computing this project is associated with are:

- (C++) compiler implementation - The primary two compilers for C++ being GCC and Clang, these are FOSS (Free and Open Source) pieces of software which I am free to build upon.
- C++ and Rust language design - Examining why these two languages are designed in the way they do, why they offer some features and don't others, and the philosophies that are behind the development of these languages and how they differ from one another.
- Introducing modern ideas to a programming language - Although C++ has continued to be developed and has many modern features behind it now such as co-routines, languages that have been designed more recently don't have the issue of being forced (or heavily pushed) towards keeping older bad decisions as part of the language. For example, ABI (Application Binary Interface) breaking changes.
- Improving memory safety in C++ - C++ has a reputation for being a relatively error prone programming language (Swift, 2017), however, because it is such a widely used language and an industry standard it is easier to create tools to improve C++ code than it is to create a new language, teach all the developers around the world the new language and all it's features, philosophies, quirks, and then rewrite all the software in said new language.

The main idea behind my project is porting over a feature from one programming language to another, in this case the Rust Borrow Checker to C++. This will involve adding additional source code to existing C++ compilers to add the functionality required.

3 Expected outputs

The output of my project will be additional open source source code to the Clang C++ compiler. This will allow for a demonstration of the changed behaviour due to my implementation. It will also mean that interested parties can download the code and add it to their projects, or inspect / add to it etc.

The demonstration could include a presentation which shows an overview and brief comparison of the two codebases which will be created, an explanation

of the borrow checker and the benefits of it, and showing how the C++ compiler (with my additional source code) can catch memory errors at compile time.

4 Motivation for the project: client or audience

This project is beneficial to members of the C++ programming community and by extension, the programming community as a whole. The aim is to prove that it's possible to bring together some the best of the features of the C++ and Rust programming language features. The rationale behind this is that C++ has 40 years worth of libraries, community development, educational material etc behind it, as opposed to Rust which has little in comparison. C++ also has the advantage of being an industry leader / standard, holding the 7th spot on the TIOBE index, compared to Rust which holds the 27th spot (*TIOBE Index for December 2021*, 2021). It's therefore possible to make the case that this benefits companies hiring C++ and rust developers as it can improve to codebase of any given project by improving memory safety, for example, memory safety is the cause of over 65% of *"serious security"* bugs in Chromium, the browser engine behind Google Chrome, Microsoft Edge and many others (Google, n.d.).

I can determine whether my solution is successful by demonstrating a working proof of concept of the rust borrow checker in C++, this will be formally defined within the tests that will be developed mentioned above, formally classifying the behaviour. This is mentioned more in section 5.

5 Primary research plan

My primary research plan is to create a small code base in both C++ and Rust which highlights the difference created by the borrow checker. I can then create my implementation of the borrow checker in C++, and build a test suite for both codebases and examine the assembly outputs of each codebase to compare the differences between them as well as comparing the behaviours of the different programs.

Steps to take to reach my end goal are as follows:

1. Create a small codebase in C++, and illustrate where the potential memory safety mistakes are which could be resolved by the borrow checker
2. Create the equivalent code in Rust (note: for the two codebases a series of tests will be written to ensure identical behaviour aswell as assembly output analysis)
3. Create code to add to the C++ compiler, which will warn the user about previously mentioned memory safety errors and enforce borrow checker-like fixes
4. Demonstrate working code after fixes

6 Bibliography

Google. (n.d.). Memory safety. https://www.chromium.org/Home/chromium-security/memory-safety

Swift, J. (2017). *Which languages are bug prone.* https://www.i-programmer.info/news/98-languages/11184-which-languages-are-bug-prone.html

Tiobe index for december 2021 (tech. rep.). (2021). TIOBE. https://www.tiobe.com/tiobe-index/

Word count: 773