Project C.A.S.P. - Weekly Update 1

Ryan Tedeschi, Dylan Carson

January 30, 2017

1 Introduction

During the winter break between the Fall 2016 and Spring 2017 semesters at California University of Pennsylvania, development on Project C.A.S.P. (Code Analyzer Software Package) began. The team initiated the implementation phase by setting up source control and beginning development on the core module. The design document was used as a roadmap for the initial setup, and a custom build environment was created for ease of integration.

2 Source Control Setup

Source control was set up using a GitHub repository to ensure that each member had access to the most recent code without worry of duplication. This was done so that it was not necessary for the team members to collaborate simply to retrieve the latest updates on the project. With the repository in place, each member can work autonomously as permitting. This reduces time wasted on keeping each member up to date throughout their busy schedules with work, school, etc.

3 Development Progress

Development completed was minimal. However, the program currently accepts arguments from the command line in the format that they should eventually be accepted. The program does not yet perform necessary operations on the input data, but the parse functionality has been started. A file has been created defining a partial context-free grammar of C++ to allow some C++ code to be parsed.

4 Directory Structure

The directory structure was set up to accommodate all three implementations – command line interface, standalone application, and Microsoft Visual Studio plugin. Each implementation was given a separate directory in the repository

to separate concerns of code bases. In addition, within the command line interface's directory, multiple subdirectories were created. Directories for individual builds, build logs, object files, plugin modules, shared code, core code, and miscellaneous tools were created. In this directory, scripts for automated project builds also exist.

Individual builds and build logs are created by the custom build environment so that the team has progress records throughout the implementation phase. Object files are created by the custom build environment so that each part of the project can be compiled independently of the others. Plugins are kept in a separate directory so that the line of abstraction between the core module and the plugin modules is kept very distinct, and, in turn, this makes it easier for developers to create new plugins for the software in the future. The shared code directory contains files of classes that will be used in both the core modules and the plugin modules. The core code directory contains all files necessary for the core software to operate. Lastly, the miscellaneous tools directory contains any tools that play an integral part in construction of the project, including the Microsoft batch script that automates the build.

The custom build environment was setup using Microsoft Visual Studio Build tools, utilizing the "nmake" command provided by these tools. This command uses a makefile, also created by the team and found in the base of the command line interface directory, to automate the build process. This approach – using a makefile to automate the build – is similar to that used by Unix environments, and is why the approach was chosen.

5 Weekly Tasks

In the coming week, the team will be focused on completion of the core module. All other modules depend on the operation of this core module, which gives it a very high priority for completion. However, this module will require more than just the week to complete. The team will target the *Parse* functionality of the code this week, as it contains the most complicated logic in the core module.