COMP 496

Deliverables

Jason Bell 3078931

February 27, 2016

Important note: a version of Visual Studio 2015 may need to be installed for the Pie compiler to execute!

Because gmail won’t allow emailing of executables, the pie compiler will be named piec.dat, and will need to be renamed to piec.exe. Irony.dat also needs to be renamed to Irony.dll, and Pie.VisualStudio.dat to Pie.VisualStudio.vsix

All programs and batch files are best executed from the command prompt.

The following can be found in the root folder:

1. LanguageSpec.docx: a Pie language specification document.
2. Pie.VisualStudio.vsix: a Pie language syntax highlighter for Visual Studio 2015. This extension should work for any 2015 version.
3. Proposal.docx: The original project proposal
4. Report.docx: the final project report

The source folder contains the following:

1. piec.exe: This is the Pie compiler, the source code for which can be found in the Pie folder. It executes with the command arguments:

piec.exe <outputfile> <test|notest> <source files>

The output file can be an exe or dll, though an exe requires an entry point in the format:

act Main(args):

The second argument can be either test or notest, indicating whether to perform unit testing and coverage analysis.

The third argument is a space separated list of Pie source files.

1. AllSensors.txt and PassedSensors.txt: these are repositories used to store code coverage analysis information: I recommend not doing anything with them.
2. CompileCompiler.bat: this is a batch file that will compile a new Pie compiler from the Pie source code found in the Pie folder. The new compiler is called piecnew.exe, and can be renamed to piec.exe to replace the prior one.
3. TestCompiler.bat: this is a batch file that compiles a new Pie compiler with code coverage sensors installed, and performs the coverage analysis. The test compiler is output as a DLL (pietest.dll) to avoid it also being used as a compiler: this will result in an exponential explosion in sensor data. Depending on the computer being tested on, the testing and coverage analysis may take a minute or two.
4. TestSimple.bat: performs unit testing and coverage analysis with TestExamples/Simple.pie. It should result in 100% code coverage.
5. TestTwoThirds.bat: performs unit testing and coverage analysis with TestExamples/TwoThirds.pie. It should result in 66% code coverage.
6. CompileFibonacci.bat: compiles TestExamples/Fibonacci.pie into an executable (Fibonacci.exe) that allows the user to determine a Fibonacci sequence number. This is a brute force algorithm, so don’t try numbers over 40 or so!
7. TestFibonacci.bat: performs unit testing and coverage analysis with TestExamples/Fibonacci.pie. It should result in 75% code coverage.
8. Irony.dll: this is a dependency of the Pie compiler: the Irony .NET parsing library.