Reduced Triangle Compiler Report

Jack Neilson

November 3, 2017

1 Introduction

The goal of this program is to take a source file written in a reduced form of the triangle langauge and compile it in to a source language. This first submission deals with only the scanner and parser parts of the semantic analyser. The program has been implemented in C# using .NET Core.

2 Implementation

2.1 Scanner

The scanner reads the source file a single character at a time and groups them in to tokens, ignoring white space and comments, with a single class "Scanner.cs". It returns an enumerable containing all tokens to be iterated over by the parser. A regular expression matching any character in the alphabet is used to generate identifier tokens (see example 1). Any other single-character special tokens (numerals, operators, EOF) have their own, seperate logic in a switch statement to generate the appropriate token (see example 2).

2.2 Parser

3 Examples

3.1 Example 1

```
Regex token = new Regex("^[a-zA-Z]+");

if (token.IsMatch(((char) _source.Current).ToString())) {
        while (token.IsMatch(((char) _source.Current).ToString())) {
                TakeIt();
    }
        return TokenKind.Identifier;
}
```

3.2 Example 2

```
case '+':
        TakeIt();
        return TokenKind.Operator;
```

3.3 Example 3