Miniature Pascal Compiler

John Dawkins

Overview

This paper gives a description of a coding project I created for Augsburg University Professor Erik Steinmetz' Compilers II computer science class. This project is centered on a computer program that is a compiler and takes as input Pascal code and then outputs MIPS code that will perform the instructions specified in the input code upon execution by a MIPS simulator. The compiler was written in java and is made up of the following java packages:

- scanner
- variableType
- syntaxTree
- symbolTable
- parser
- compiler

These modules are described briefly now and in depth later on. The scanner module dispenses the input stream tokens. The variableType class stores a variable type. The syntaxTree package contains a class for each type of node created by the parser tree. These nodes generate the MIPS code. The symbol table is used by the parser to store and get info about variables. The parser reads the input stream and builds the syntax tree. The compiler package implements the final program.

scanner

The Token class, found in the scanner package, represents one token from the input stream. A token could be an identifier(variable name, function name, etc.), a binary operation(+, *, and, etc.) or another lexical component of a pascal program. The MyScanner class is initialized with a pascal program and then provides a function called "nextToken()" that returns the next token from the input stream of the pascal code beginning with the first.

variableType

The variableType class represents a variable type. This could be a single integer, real number or an array of either integers or real numbers.

syntaxTree

The syntaxTree package contains the following classes:

- ProgramNode: It is the root of the syntax tree. It stores one DeclarationsNode, one SubProgramDeclarationsNode and one CompoundStatementNode.
- DeclarationsNode: It stores an array of VariableNodes.
- VariableNode: It stores a variables name and type.
- StatementNode: It is the abstract base class for all different StatementNodes. These are: CompoundStatementNode, IfThenStatementNode, ReadStatementNode, ReturnStatementNode, SubProgramStatementNode, VariableAssignmentStatementNode, WhileDoStatementNode and WriteStatementNode.
- ExpressionNode: It is the abstract base class for all different ExpressionNodes. These are: NotValueExpressionNode, OperationExpressionNode, SubProgramValueExpressionNode and VariableValueExpressionNode.
- SubProgramDeclarationsNode: It stores an array of SubProgramNodes, each of which stores a DeclarationsNode and a CompoundStatementNode.

Any instance of any of these classes can be tested using the indentedToString() method. Each class supplies the toMips() function which returns the final pascal code.

symbolTable

The symbolTable class stores identifiers for the parser. It provides functions for storing identifiers and what they describe and then retrieving this information.

parser

The most mathematically complicated class, Parser, builds a syntax tree given an initialized MyScanner object. The parser uses recursive descent and, therefore, its code resembles the grammar(of the miniature pascal language).

compiler

Provides a main function that takes as input a pascal program on the command line and returns the generated MIPS code. It does so by initializing a MyScanner, passing this to a Parser object which returns a ProgramNode object and then returning the output of the ProgramNode's toMips() function.