August 21, 2018

Tecnologico y de Estudios Superiores de Monterrey
Campus Estado de Mexico
School of Engineering and Sciences
Computer Science Department

Programming Languages and Translators
Course project
Part 1

Instructor Edgar E. Vallejo
Due: September 18, 2018, 7:00 PM.

1 Introduction

In this first part of the course project, you will use `Flex` and `Bison` to implement an scanner and a parser for the `C--` programming language.

The program must be capable to received source programs written in `C--` as input and to print the list of identifiers from the symbol table as output, providing that the source program has no lexical and syntactical errors.

The parser will command the process using the method of syntax directed translation. The scanner will be invoked by the parser and the scanner report the recognized token categories and their corresponding lexemes in the source program. In addition, the scanner will enter the names of the recognized identifiers into the symbol table, along with the number of occurrences of each identifier in the source program.

2 Scanner implementation

The scanner must implement the following functionality:

1. Ignore blank spaces,
2. Ignore comments,
3. Recognize keywords and return their token category
4. Recognize delimiters and one-character operators and return their `ASCII` code as their token category.
5. Recognize two-character operators and return their token category
6. Recognize integer, double, and boolean constants, and return their token category and their value
7. Recognize identifiers and return their token category. In addition, if the identifier has not been identified, the identifier must be entered into the symbol table; if the identifier has been already identified, their number of occurrences must be incremented.
8. Print a generic error message for illegal tokens, indicating the line number where the error occurs

3 Implementation of the symbol table

A hash table is recommended to implement the symbol table. Other alternatives such as dictionaries are allowed. The symbol table will be used in later stages of the compilation process.

4 Parser implementation

The provided grammar definition can not be entered directly into Bison, but the least possible modifications to the original grammar are encouraged. Most grammatical ambigüities can be solved by enforcing the appropriate precedence of operators.

The parser must be validated exhaustively with both valid and invalid source programs. During testing, the use of programs that include most of the grammatical categories of the language and potential grammatical errors, is encouraged.

A syntactically valid program must produce no output. However, the parser should report a generic error message when a syntactic error is found.

5 Required documents:

Submit the following in a compressed file on Blackboard:

- Source and executable programs for the scanner and the parser.
- Instructions required to compile and run the programs
- Sample runs (including both valid and invalid programs)
- One-page report describing the contribution from each group member