



Prince Sultan University
Department of Computer & Information Sciences

CS320: Programming Languages: Concepts and Paradigms

Lexical Analysis Project

Prepared by:

Tala Alhazmi - 220410073

Instructor:

Dr. Siwar Rekik

Lexical Analysis

Lexical analyzers, known as the front-end of the parser, are pattern matchers that search for a substring of a given string of characters that matches a certain pattern. Lexical analyzers perform syntax analysis at the lowest level of program structure by reading the input set of characters and producing a corresponding sequence of tokens as output. A token is a lexical category of a group of characters with a collective memory. Lexical analyzers identify lexemes (a particular instance of a token) which matches a certain pattern.

This project aims to improve the lexical analyzer (provided in section 4.2 (page 190) in Concepts of Programming Languages (Sebesta)) using C language. The improved lexical analyzer will be able to:

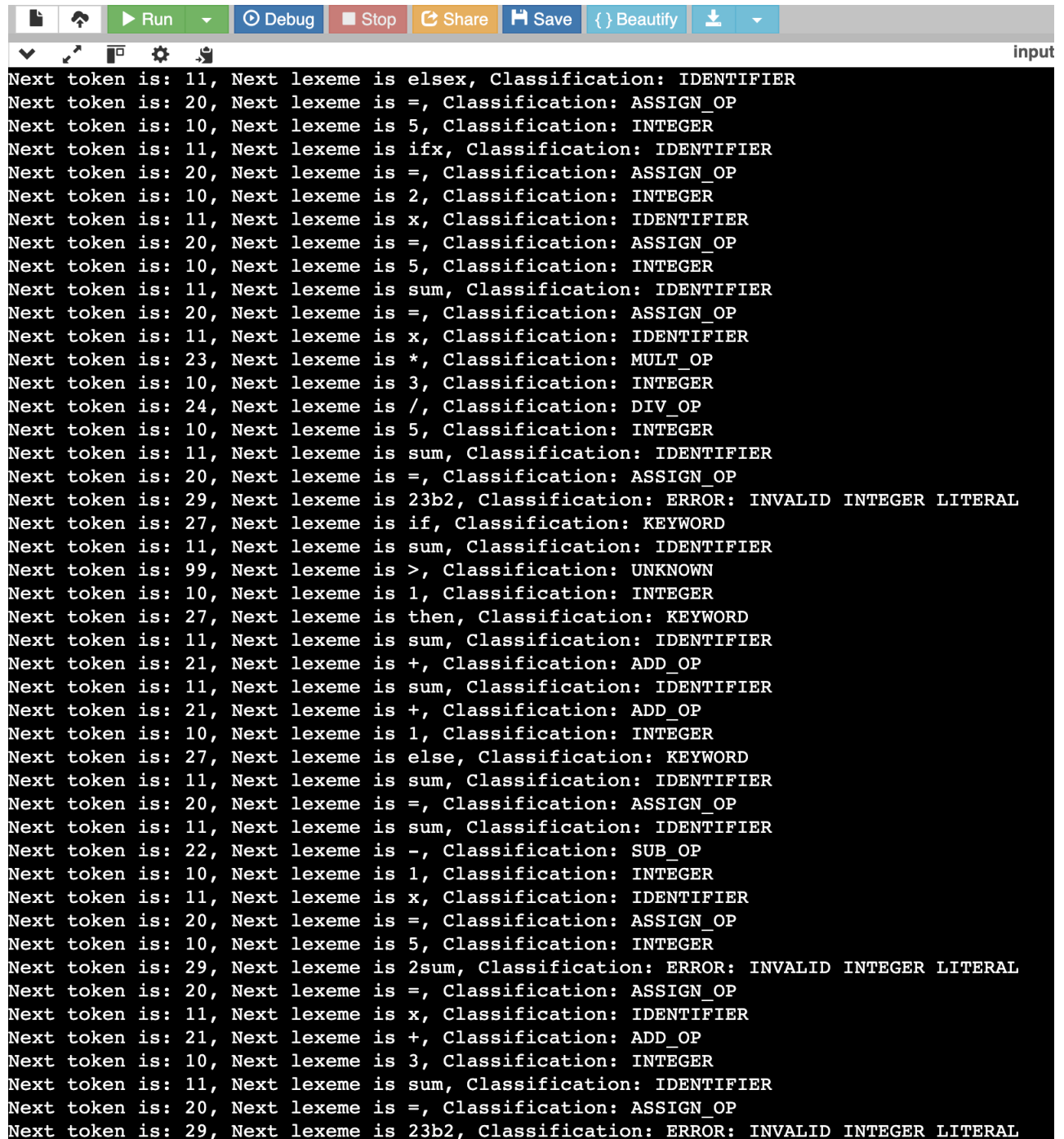
- Report and identify errors in identifiers and integer literals.
- Recognize certain keywords and assignment operators.

The input will be given via an external file called input.txt. [View input file.](#)

Improved Lexical Analyzer:

Coding was done via OnlineGDB, an online compiler and debugger for c/c++.

Output:



The screenshot shows a code editor with a toolbar at the top containing icons for file operations, a 'Run' button, a 'Debug' button, a 'Stop' button, a 'Share' button, a 'Save' button, a 'Beautify' button, and a download icon. The editor area displays a list of tokens and their classifications, with the word 'input' in the top right corner. The list consists of 48 lines, each starting with 'Next token is:' followed by a token number, 'Next lexeme is' followed by the lexeme, and 'Classification:' followed by the classification. The classifications include IDENTIFIER, ASSIGN_OP, INTEGER, MULT_OP, DIV_OP, KEYWORD, ADD_OP, SUB_OP, and ERROR: INVALID INTEGER LITERAL.

```
Next token is: 11, Next lexeme is elsex, Classification: IDENTIFIER
Next token is: 20, Next lexeme is =, Classification: ASSIGN_OP
Next token is: 10, Next lexeme is 5, Classification: INTEGER
Next token is: 11, Next lexeme is ifx, Classification: IDENTIFIER
Next token is: 20, Next lexeme is =, Classification: ASSIGN_OP
Next token is: 10, Next lexeme is 2, Classification: INTEGER
Next token is: 11, Next lexeme is x, Classification: IDENTIFIER
Next token is: 20, Next lexeme is =, Classification: ASSIGN_OP
Next token is: 10, Next lexeme is 5, Classification: INTEGER
Next token is: 11, Next lexeme is sum, Classification: IDENTIFIER
Next token is: 20, Next lexeme is =, Classification: ASSIGN_OP
Next token is: 11, Next lexeme is x, Classification: IDENTIFIER
Next token is: 23, Next lexeme is *, Classification: MULT_OP
Next token is: 10, Next lexeme is 3, Classification: INTEGER
Next token is: 24, Next lexeme is /, Classification: DIV_OP
Next token is: 10, Next lexeme is 5, Classification: INTEGER
Next token is: 11, Next lexeme is sum, Classification: IDENTIFIER
Next token is: 20, Next lexeme is =, Classification: ASSIGN_OP
Next token is: 29, Next lexeme is 23b2, Classification: ERROR: INVALID INTEGER LITERAL
Next token is: 27, Next lexeme is if, Classification: KEYWORD
Next token is: 11, Next lexeme is sum, Classification: IDENTIFIER
Next token is: 99, Next lexeme is >, Classification: UNKNOWN
Next token is: 10, Next lexeme is 1, Classification: INTEGER
Next token is: 27, Next lexeme is then, Classification: KEYWORD
Next token is: 11, Next lexeme is sum, Classification: IDENTIFIER
Next token is: 21, Next lexeme is +, Classification: ADD_OP
Next token is: 11, Next lexeme is sum, Classification: IDENTIFIER
Next token is: 21, Next lexeme is +, Classification: ADD_OP
Next token is: 10, Next lexeme is 1, Classification: INTEGER
Next token is: 27, Next lexeme is else, Classification: KEYWORD
Next token is: 11, Next lexeme is sum, Classification: IDENTIFIER
Next token is: 20, Next lexeme is =, Classification: ASSIGN_OP
Next token is: 11, Next lexeme is sum, Classification: IDENTIFIER
Next token is: 22, Next lexeme is -, Classification: SUB_OP
Next token is: 10, Next lexeme is 1, Classification: INTEGER
Next token is: 11, Next lexeme is x, Classification: IDENTIFIER
Next token is: 20, Next lexeme is =, Classification: ASSIGN_OP
Next token is: 10, Next lexeme is 5, Classification: INTEGER
Next token is: 29, Next lexeme is 2sum, Classification: ERROR: INVALID INTEGER LITERAL
Next token is: 20, Next lexeme is =, Classification: ASSIGN_OP
Next token is: 11, Next lexeme is x, Classification: IDENTIFIER
Next token is: 21, Next lexeme is +, Classification: ADD_OP
Next token is: 10, Next lexeme is 3, Classification: INTEGER
Next token is: 11, Next lexeme is sum, Classification: IDENTIFIER
Next token is: 20, Next lexeme is =, Classification: ASSIGN_OP
Next token is: 29, Next lexeme is 23b2, Classification: ERROR: INVALID INTEGER LITERAL
```

```
Run Debug Stop Share Save {} Beautify ing
Next token is: 99, Next lexeme is >, Classification: UNKNOWN
Next token is: 10, Next lexeme is 1, Classification: INTEGER
Next token is: 27, Next lexeme is then, Classification: KEYWORD
Next token is: 11, Next lexeme is sum, Classification: IDENTIFIER
Next token is: 21, Next lexeme is +, Classification: ADD_OP
Next token is: 11, Next lexeme is sum, Classification: IDENTIFIER
Next token is: 21, Next lexeme is +, Classification: ADD_OP
Next token is: 10, Next lexeme is 1, Classification: INTEGER
Next token is: 27, Next lexeme is else, Classification: KEYWORD
Next token is: 11, Next lexeme is sum, Classification: IDENTIFIER
Next token is: 20, Next lexeme is =, Classification: ASSIGN_OP
Next token is: 11, Next lexeme is sum, Classification: IDENTIFIER
Next token is: 22, Next lexeme is -, Classification: SUB_OP
Next token is: 10, Next lexeme is 1, Classification: INTEGER
Next token is: 11, Next lexeme is x, Classification: IDENTIFIER
Next token is: 20, Next lexeme is =, Classification: ASSIGN_OP
Next token is: 10, Next lexeme is 5, Classification: INTEGER
Next token is: 29, Next lexeme is 2sum, Classification: ERROR: INVALID INTEGER LITERAL
Next token is: 20, Next lexeme is =, Classification: ASSIGN_OP
Next token is: 11, Next lexeme is x, Classification: IDENTIFIER
Next token is: 21, Next lexeme is +, Classification: ADD_OP
Next token is: 10, Next lexeme is 3, Classification: INTEGER
Next token is: 11, Next lexeme is sum, Classification: IDENTIFIER
Next token is: 20, Next lexeme is =, Classification: ASSIGN_OP
Next token is: 29, Next lexeme is 23b2, Classification: ERROR: INVALID INTEGER LITERAL
Next token is: 27, Next lexeme is if, Classification: KEYWORD
Next token is: 11, Next lexeme is sum, Classification: IDENTIFIER
Next token is: 99, Next lexeme is >, Classification: UNKNOWN
Next token is: 10, Next lexeme is 1, Classification: INTEGER
Next token is: 27, Next lexeme is then, Classification: KEYWORD
Next token is: 11, Next lexeme is sum, Classification: IDENTIFIER
Next token is: 21, Next lexeme is +, Classification: ADD_OP
Next token is: 11, Next lexeme is sum, Classification: IDENTIFIER
Next token is: 21, Next lexeme is +, Classification: ADD_OP
Next token is: 10, Next lexeme is 1, Classification: INTEGER
Next token is: 27, Next lexeme is else, Classification: KEYWORD
Next token is: 11, Next lexeme is sum, Classification: IDENTIFIER
Next token is: 21, Next lexeme is +, Classification: ADD_OP
Next token is: 11, Next lexeme is sum, Classification: IDENTIFIER
Next token is: 22, Next lexeme is -, Classification: SUB_OP
Next token is: 10, Next lexeme is 1, Classification: INTEGER
Next token is: -1, Next lexeme is EOF, Classification: EOF

...Program finished with exit code 0
Press ENTER to exit console.
```

References and Resources:

1. **OnlineGDB:** Online C Compiler. https://www.onlinegdb.com/online_c_compiler#
2. Sebesta, R. W. (2019). *Concepts of Programming Languages* (12th ed.). Pearson.