Kennesaw State University

Department of Computer Science

CS 4308

Concepts of Programming Languages

Section 01

Project Deliverable 1

Charles Wildner: [**cwildner@students.kennesaw.edu**](mailto:cwildner@students.kennesaw.edu)

Nick Tanner: [**ntanner9@students.kennesaw.edu**](mailto:ntanner9@students.kennesaw.edu)

Rachel Anderson: [**rande147@students.kennesaw.edu**](mailto:rande147@students.kennesaw.edu)

Selam Kelil: [**skelil@students.kennesaw.edu**](mailto:skelil@students.kennesaw.edu)

Zachary Carruthers: [**zcarruth@students.kennesaw.edu**](mailto:zcarruth@students.kennesaw.edu)

**Date:** 2/15/2024

1. **Initial Problem Statement**
   1. The initial problem for Deliverable 1 is to develop the scanner component of an interpreter for a specific subset of the SCL (Structured Control Language) programming language. This involves creating a scanner capable of reading SCL files, tokenizing the source code, and defining the grammar for the specific subset of SCL. This is a crucial component for the subsequent stages of the project, as it forms the foundation for the interpreter’s ability to process SCL programs accurately.
2. **Summary/Purpose**
   1. The purpose of this report is to document the development process and outcomes of deliverable 1, focusing on the implementation of the scanner for the SCL interpreter project. The primary objectives for deliverable 1 are:
      1. Implementing the Scanner: Develop a scanner to tokenize SCL source code, separating it into meaningful elements based on the specified syntax rules.
      2. Creating lists of tokens: Use arrays to store the necessary keywords, identifies (variables), operators, constants, and special characters used in the SCL subset, enabling the scanner to categorize tokens accurately.
      3. Defining the grammar: Define the grammar of the SCL subset in BNF or EBNF form, serving as a formal specification of the language syntax and guiding the development of the scanner and subsequent interpreter components. We are defining the grammar in a slightly modified BNF notation conforming to the following rules: The “ : “ is the same as the right arrow in BNF, every rule ends with a semicolon, and the non-terminals are written in upper-case.
   2. By completing deliverable 1, we aim to establish a robust foundation for the SCL interpreter project, enabling accurate parsing and tokenization of SCL programs. This report will detail the design decisions, implementation strategies, and testing procedures employed during the development of the scanner.
3. **Solution**
   1. Scanner:
      1. The first step was to create a scanner that can read the source code and then create a list of tokens. The scanner will read the entire file into string.
      2. These strings will then be split into tokens based on:
         1. All words with letters only
         2. All words with numbers only
         3. All punctuation/operators (,.!@#$%^&\*()+=\_[]{}/?’”;:<>) either in singleton form or repeated
      3. These tokens are then stored in a list, which is iterated over as list items are compared against a dictionary.
      4. After comparison with the dictionary, the entries produce the full form of the token which is included in the JSON file.
   2. Dictionary:
      1. The dictionary consists of multiple arrays that are formed from the chosen subset of the SCL language. In this case, we are using the attached “welcome.scl” file as the subset of the SCL language. Only the tokens obtained from scanning the welcome.scl file are defined within this document, refer to the attached “scl\_grammar.txt” file for the full grammar.
         1. Keywords

‘imports’ ‘implementations’ ‘function’ ‘main’ ‘return’ ‘type’ ‘integer’ ‘is’ ‘variables’ ‘define’ ‘of’ ‘double’ ‘begin’ ‘display’ ‘set’ ‘exit’ ‘endfun’

* + - 1. Identifiers(variables)

‘x’

* + - 1. Operators

‘=’

* + - 1. Special Symbols

‘,’ ‘\n’

* 1. Defined Grammar:

Please see attached scl\_grammar.txt file.

1. **Data/Results**
   1. Input file:

A screenshot of a computer program

Description automatically generated

* 1. Output result:

A screen shot of a computer

Description automatically generated

1. **Conclusion**

In conclusion, we have created a python program that successfully scans and tokenizes a SCL file, using a subset of the overall SCL grammar. It separates these tokens into arrays based upon their status as either keywords, identifiers, operators, or special symbols. Furthermore, this program accepts command-line input.

1. **References**

[1] SCL Fundamentals, PubTeX output, 1999, October 12. Supplied with assignment materials.