CS 4308 – Concepts of Programming Languages

Course Project

The CS4308 project consists of developing a complete (but small) interpreter for a subset of the SCL language. The specification of the SCL grammar is given below. The interpreter can be implemented in any of the following **programming languages: C , C + + , Java, Python, or Ada.**

Interpreter Project

The interpreter will process an SCL program and build some intermediate form of the program. This intermediate form will then be interpreted to run the program.
All tokens in this language are separated by white space. The parsing algorithm should detect any syntactical or semantic error. The first such error discovered should cause an appropriate error message to be printed, and then the interpreter should terminate. Run-time errors should also be detected with appropriate error messages being printed.

Project Deliverables (see course schedule for due dates):

1. Module_3 – 1st Deliverable

Develop a complete scanner for SCL. You must show the execution of this program by using a relevant source file as input, the program must show a list of the tokens recognized. Write a short report describing the work performed and include the input file and output data produced.

2. Module_5 – 2nd Deliverable

Develop a complete parser that executes with the scanner. You must show the execution of this program by using a relevant source file as input, the program must show the corresponding statements recognized. Write a short report describing the work performed and include the input file and output data produced.

3. Module_7 – 3rd Deliverable

Develop a complete interpreter that executes with the scanner and parser. You must show the execution of this program by using a relevant source file as input, the program must show the results after executing the statements. Write a short report describing the work performed and include the input file and output data produced. .

Grammar for the (subset of SCL) language

Syntax Rules, see 'scl_subset.yy'

Lexical Rules

identifier → letter char_list

char_list → letter_digit char_list | letter_digit

literal_integer → digit literal_integer | digit

assignment_operator → =

le_operator → <=

lt_operator → <

ge_operator → >=

gt_operator → >

eq_operator → = =

ne_operator → ~=

add_operator → +

sub_operator → -

mul_operator → *

div_operator → /