CSCI 4160 Project5

Due: see class calendar

Goal:

This assignment serves several purposes:

- to be familiar with Visual Studio, the official IDE for this class;
- to understand class inheritance and benefits of polymorphism
- to be familiar with symbol table, and abstract syntax tree

Description:

In this assignment, you are required to write an interpreter for a programming language (Don't be scared, you don't need to know the syntax and you never need to parse the program). The input to the interpreter is an abstract syntax tree of a program, which will be provided by the instructor. The output of the interpreter is the output of the program (i.e. interpreter it).

The language can have statements and expressions. Statements have no return value, while expressions have associated integer values.

The statements can be

- compound statements: a sequence of statements separated by semicolon.
- print statement: a statement to print the values of a sequence of expressions.
- assignment statement: assign an integer value to a variable

The expressions can be

- id expression: it is purely a variable
- integer literals like 10, 34.
- arithmetic expressions using operators +, -, *, /.
- Sequence expressions: a statement followed by an expression

We defined abstract base classes *Stm* and *Exp* to represent statements and expressions. Each type of statements/expressions is defined as an individual class derived from the abstract base class *Stm*/*Exp*. We also defined an abstract base class *ExpList* and two derived classes *PairExpList* and *LastExpList* to represent a list of expressions. All classes are defined in ***slp.h*** and you should implement all member functions in **slp.cpp**. What you need to do is to provide implementation to the following member function within each non-abstract base class:

1. void/int interp (SymbolTable&) that "interprets" a statement, an expression, or a list of expressions.

The symbol table is a hash map from string to integer. The symbol table is used to store variables and their values. So you should understand when to push a variable to the symbol table and when to lookup the symbol table. In this assignment, you never delete a variable from the symbol table..

The instructor provides the driver to test your interpreter on four different test cases which can be found in prog.h.

HOW TO GET STARTED

1. The project contains the following skeleton files:
    - description1.pdf: this document
    - slp.h: definition of all classes
    - slp.cpp: skeleton file
    - prog.h: all four test cases
    - main.cpp: the driver
    - rubric5.doc: the rubric to grade your project.

   All your implementation should be placed in the file **slp.cpp**.
2. Once you have finished, submit the following file only to D2L dropbox:
    - slp.cpp