

CS 181 – Fall 2002
OUTLINE OF PROJECT #2
Due Nov 21, 11:59pm

Implement basic interpreter for SimpleSem with simple GUI

Implement a C++ program **simplesem.c** which reads SimpleSem code from a file called **code.txt** and data from a file **data.txt**. and outputs the resulting (modified) data segment to the file **data.txt**. Initially the data file may or may not exist.

The **format of code.txt** is:

| | | | | | |
|---|----------------------|--------------|---|------|---------|
| 0 | statement (or empty) | For example: | 0 | jump | 2 |
| 1 | statement (or empty) | | 1 | | |
| 2 | statement (or empty) | | 2 | set | 0, D[9] |

The **format of data.txt** is:

| | | | | |
|---|--------------------|--------------|---|-----|
| 0 | integer (or empty) | For example: | 0 | 67 |
| 1 | integer (or empty) | | 1 | |
| 2 | integer (or empty) | | 2 | -99 |

You are free to use **additional spaces** whenever needed.

SYNTAX OF BASIC VARIABLES:

Basic → integer
Basic → D[Basic]
Basic → (Basic + Basic)
Basic → (Basic - Basic)
Basic → (Basic * Basic)
Basic → (Basic / Basic)

For the sake of this project, in all SimpleSem **statements all variables must be basic**. You are free to use **additional spaces** whenever needed.

SYNTAX OF STATEMENTS:

Statement → set Basic, Basic
Statement → set write, Basic
Statement → set Basic, read
Statement → jump Basic
Statement → jumpt Basic, Basic
(Jump occurs only if the conditional variable evaluates to a **positive** integer value)
Statement → halt

GUI

The program should open a panel (window) with two scroll boxes: one displays code segment and one displays data segment. The code segment box allows you to select exactly one line of code, which always corresponds to the current instruction pointer (initially line 0). This way you can manually change the current ip. The data segment only allows you to scroll through the text (no line selection).

The program should also have one input, one output and one error text box. Input box is used to input data using “set read”. The input text is accepted by pressing carriage return. The output box is used to output data using “set write”. Both are single-line text boxes. The error box is a multi-line scroll box used to log all error messages.

The panel should also have several buttons:

Trace: Executes exactly one instruction (at the current ip), moves ip to the new position (selects the corresponding line in the code segment) and displays updated data segment.

Run: Executes the entire program (until halt, end of file or empty line) and displays the resulting data segment.

Exit: Exits the program.

Error Checking

The code should gracefully handle **all possible input errors** and output suitable error messages:

- **Before processing any given line of code (at the current ip)** the program should **check its syntax**. If any errors are found, the ip **should not move**, no other processing should take place, informative errors should be output to the error box and the execution should be interrupted.
- Before reading any line from the data segment, the program should **check the syntax** of that particular line **and the value of the data**. If any errors are found, informative errors should be output to the error box and the execution should be interrupted at the point at which the error was found. This includes negative memory references or uninitialized variables.
- **Termination of the execution** means that no more lines of code are interpreted. It does not mean exiting from the program!!!!
- Error messages should be added to the error log file so that it contains all previous error messages