Project 2, TI-83 Calculator

Due: Please check due date on BlackBoard

Objectives

The objective of this programming assignment is to have you practice implementing a new data structure and also to gain some experience using a data structure from the C++ STLs.

Introduction

In this project, using the Calculator Priority Table shown below, your program will solve a one line equation given by a data file. After the equation is broken down into tokens, the priority table helps solve and produce a value. Many examples (videos) will be available for your viewing.

Application Setup

Several stacks are used to evaluate expressions involving numbers as in an equation like:

3 \* 2 + 4 = 10

3 \* (2 + 4) = 18

4.5 \* 3 = 13.5

2.5 \* 4 = 10

A “Priority Table” is used to determine what values to calculate in which order given the rules of precedence such as ^ is calculated before \*, etc…

|  |  |  |  |
| --- | --- | --- | --- |
| **Priority Table** | | | |
| symbol | INPUT  priority | STACK  priority | Comment |
| +, - | 1 | 1 |  |
| \*, /, % | 2 | 2 |  |
| ^ | 5 | 4 | changes association from left to right |
| ( | 6 | 0 |  |
| ) | - | - | pop & process until ‘(’, then pop that too |
| $ | - | -1 |  |
| eoe | - | - | pop & process until ‘$’ |

eoe = end of equation “-“ MEANS NONE, NOT 0!!!!

|  |  |  |  |
| --- | --- | --- | --- |
| (char) | (int) |  | (???) |
|  |  |  |  |
|  |  |  |  |
|  |  | **red = default** |  |
|  |  | **(optional for your program)** |  |
| **$** | **-1** |  | **$** |
| Operator | Priority |  | Value |
| Operator Stack | |  | Digit Stack |

The “Operator” Stack is two separate, BUT related stacks. The operator is a char stack that holds ALL chars symbols in equation like: (, ), \*, ^, /, etc… while the priority stack holds the priority value of that symbol given in the priority table above.

The digit stack holds ALL numeric values which can be floats or integers, found in the equation given in the file, but notice it will be a String stack since it has a $ as its delimiter (or end marker). How you handle this detail in the stack is up to you. But the result should be an integer if a whole number and float otherwise. That is, an expression that evaluates to 3.00 should return the integer value 3.

Here is a link to [examples](https://www.youtube.com/playlist?list=PLC7fNkE1QplYiKF0MEiDegdtAgDsyPx7R) worked out and how the overall application works.

Using a C++ STL Map

One of your requirements will be to use a map to store the values of the afore- mentioned Priority Table. While a Map may not be covered in class, they are easy to use and set up. But to help you along, here are two good references:

<http://www.cplusplus.com/reference/map/map/>

<http://www.cprogramming.com/tutorial/stl/stlmap.html>

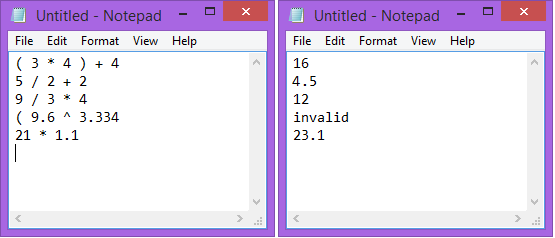
Data File and Printing results

The date file will have a few guarantees.

1. The file will only contain numbers and characters that are found on the priority table.
2. The data will be separated by a single space
3. No two numbers will not be in a adjacent (i.e. 23 45). Numbers will be separated by some operation.

After accepting the filename as a parameter in your “make run”, you should validate that the file is present, if not, state that it is missing and exit.

Results will always be written to a file named “results.txt”. The results should be printed on separate lines. The results line up with the input, such that side by side it would look like:



Notice 3(4 \* 4) will return an “invalid” statement. Yes, we know the Ti-83 would handle this correctly.

Coding and Implementation Requirements

*Note: Running time is one of the most important considerations in the implementation of a data structure. Programs that produce the desired output but exceed the required running times are considered wrong implementations and will receive substantial deductions during grading.*

1. You will use C++’s STL Stack class to hold the various calculator values and priorities. The stacks are to be named *op (for operator), priority and value*. One component of grading will evaluate how elegantly you implement this data structure. (Yes, you are being graded on aesthetics!)
2. You will use C++’s STL Map class to hold the priority table. The priority table does not make the calculations, but grabbing values from the table with ease using the Map will be helpful.
   1. as ***a suggestion***, key would be the operator, value would be priority value
   2. overall setup is up to you
3. Equations given in a file will ultimately be broken up and stored in their respective stacks.
4. The run time for evaluating EACH equation should not be anything more than O(n)
5. The main base class will be named Calculate.h. It **must** contain:
   1. **class** Calculate
   2. any other class additions for encapsulation, inheritance, etc… is up to you
   3. constructor that will accept the filename
   4. the function readFile()
      1. validates the file
   5. the function writeResults()
      1. writes results to a file
      2. depending on your implementation, this may append to the file “results.txt” or wait until all calculations are completed before writing to the results file.
      3. this is a grading and debugging tool
      4. parameters to this function are up to you
   6. the function push()
      1. can ONLY push a symbol on the stack if it’s INPUT priority is **greater** to what priority is currently at top of the stack
      2. otherwise must pop and process to make priority value lower
      3. could use this function in conjunction with Stack’s push function
   7. the function popAndProcess()
      1. where actual calculation of equation is done
      2. procedure
         1. pop 2 values from digit stack, A and B respectively
         2. pop 1 value from Operator Stack
         3. pop it’s priority too, and discard
         4. solve for *x* = B z A
            1. z (just the operator)
         5. push the computed value *x* onto the value stack
         6. pop “)“ as well if there
6. The driver will be named Driver.cpp. It **must** contain:
   1. greeting function named printGreeting(), that will print YOUR NAME AND SECTION
   2. an instance of “Calculate”, **in which the instance will be given a filename to read**
7. Specifications other than what is present in project documents are up to you (some flexibility).

Running and Compiling Requirements

Please compile and complete your work on your own directory on GL. I have had several students that are compiling on mine (or slupoli/pub/cs341...). This takes up a lot of space and my disk quota. My directory is only to be used a repository for your completed files. If this is abused, your directory will be closed automatically by GL.

## What to Submit

Read the [course project submission procedures](http://www.csee.umbc.edu/courses/undergraduate/341/fall13/projects/submission.shtml). *Submission closes by script immediately after 9pm.* Submit well before the 8:59pm deadline, because 9:00:01 might already be late (the script takes only a few seconds to run).

You should copy over all of your code under the src directory. You must also supply an appropriate makefile. Do NOT submit test data files. **Any unnecessary files submitted will be considered for a deduction.**

Make sure that your code is in the ~/cs341proj/proj2/src directory and not in any other subdirectory of ~/cs341proj/proj2/. In particular, the following Unix commands should work.

cd ~/cs341proj/proj2/src

make

make run **FILE=**data.txt (data.txt could be ANY filename)

make clean