CS6771 Assignment One 2017

Stack Based Calculator

Due Date: 11:59pm Sunday 13 August 2017

Extended to: 11:59pm Tuesday 15 August 2017

Value: 10%

Aims:

- Familiarity with C++ Containers
- File and string processing

Write a stack-based calculator program in C++. The calculator reads tokens (numbers and commands) from a specified input file. In the input file each token is separated by a space or a new line. Each number is placed on a stack and each command consumes one or more values from the stack. A command may result in a modification of the stack, an additional number added to the stack, and/or output printed to the console (stdout / cout). The calculator should support int and double values and only promote an int to a double when necessary. Your program should be written in pure C++ (not C) and should make use of the C++ STL. When printing output your double values should be formatted to three decimal places. Your calculator should act on the following tokens:

Token	Description	Console Output
x.y	A double (where x and y are numbers). All doubles will have decimal places. e.g., 2.1	
x	An integer (where x is a number).	
add	Removes two numbers from the stack and calculates their sum. The sum is pushed back onto the stack.	x + y = z
sub	Removes two numbers from the stack and subtracts the second number removed from the first. The result is pushed back onto the stack.	x - y = z
mult	Removes two numbers from the stack and multiplies their product. The result is pushed back onto the stack.	x * y = z
div	Removes two numbers from the stack and divides the first by the second. The result is pushed back onto the stack.	x / y = z
sqrt	Removes the top of the stack and calculates the sqrt of this number. Pushes the result back onto the stack.	sqrt y = z
pop	Removes the top of the stack.	
reverse	Removes the top of the stack. The ordering of the next n (the number that was at the top of the stack) elements are reversed on the stack.	
repeat	Removes the top of the stack. Numbers and commands continue to be read from the file but not acted on until an endrepeat command.	
endrepeat	Processes the numbers and commands that were read from the input file but stored from the start of the repeat command. These numbers and commands are repeated n times (where n is the number that was at the top of the stack when the repeat command was given).	

Sample input:

```
10 20 sub
4 3 add
2 mult
16.0
2 repeat
sqrt
endrepeat
pop
2.0
mult
3 repeat
2
2 reverse
div
3 mult
endrepeat
```

Sample output:

```
20 - 10 = 10
3 + 4 = 7
2 * 7 = 14
sqrt 16.000 = 4.000
sqrt 4.000 = 2.000
2.000 * 14 = 28.000
28.000 / 2 = 14.000
3 * 14.000 = 42.000
42.000 / 2 = 21.000
3 * 21.000 = 63.000
63.000 / 2 = 31.500
3 * 31.500 = 94.500
```

Getting started:

The following program will allow you to read a file from the command line and sets up the formatting for printing doubles to three decimal places.

```
#include <iostream>
#include <fstream>
#include <iomanip>

int main(int argc, char* argv[]) {

        // setup the print out format for the precision required.
        std::cout.setf(std::ios::fixed,std::ios::floatfield);
        std::cout.precision(3);

        // open the file for reading
        std::ifstream in;
        in.open(argv[1]);

        // string to be read into
        std::string s;

        // read the file while we have input.
        while (in >> s) {

        }
        in.close();
}
```

Tips:

- Use C++! Do not use C-style printfs, unions, macros or arrays.
- You shouldn't need any pointers or dynamic memory to complete this assignment (but you may want to use references and const).
- The lecture slides and tutorials have many code snippets that you may find helpful.
- You may need multiple containers (stack, queue, vector, etc).
- Use the C++ STL - you shouldn't build your own container classes or use other libraries (e.g., boost).
- You shouldn't need to create any classes (but you may if you like).
- To detect if a string is a double look for a '.' in the token, e.g.,

  ```
  s.find('.') != std::string::npos
  ```

- To detect if a string is a number look for a number in the first position of the token, e.g.,

  ```
  isdigit(s[0])
  ```

- To convert a string to an int or a double use

  ```
  std::stoi(s) or std::stod(s).
  ```

- To calculate a square root you may need to

  ```
  #include <cmath>
  ```

- Carefully consider when you need to promote an int to a double. Any command that only uses ints should return an int result, only commands that mix ints and doubles should result in a double.
- The reference solution is 250 lines including comments.
- All testing input will be valid and computable (e.g., no '-1 sqrt') however you should be able to handle or ignore bad input.
- This assignment should be relatively easy, mixing ints and doubles is the only difficult bit.
- All your code should be placed in one C++ source file, you don't need header or makefiles.
- Make sure your code reads a file specified in argv[1] and outputs using std::cout

Testing

Place all your code in a file called calculator.cpp

Ensure it compiles on the CSE machines under the following command:

```
g++ -std=c++14 -Wall -Werror -O2 -o calculator calculator.cpp
```

To run your code type:

```
./calculator input.txt
```

You should create your own input files to test the full functionality of your code against the specifications.

Marking

Your submission will be given a mark out of 10 with 7 an automarked performance component for output correctness and 3 manually marked component for code style and quality.

As this is a third-year course we expect that your code will be well formatted, documented and structured. We also expect that you will use standard formatting and naming conventions.

Note, if you write in C or use C types (e.g. union) or #define macros you will lose performance marks as

well as style marks.

A number of test cases will be used to mark your solution. To pass a test case, your solution must produce exactly the same output as the reference solution. The results from both will be compared by using the linux tool diff.

Submission Instructions

Copy your code to your CSE account and make sure it compiles without any errors or warnings. Then run your test cases. If all is well then submit using the command:

```
give cs6771 ass1 calculator.cpp
```

If you submit and later decide to submit an even better version, go ahead and submit a second (or third, or seventh) time; we'll only mark the last one. Be sure to give yourself more than enough time before the deadline to submit.

Late submissions will be penalised unless you have legitimate reasons for an extension which is arranged before the due date. Any submission after the due date will attract a reduction of 20% per day to your individual mark. A day is defined as a 24-hour day and includes weekends and holidays. No submissions will be accepted more than three days after the due date.

Plagiarism constitutes serious academic misconduct and will not be tolerated. CSE implements its own plagiarism addendum to the UNSW plagiarism policy. You can find it here: http://www.cse.unsw.edu.au/~chak/plagiarism/plagiarism-guide.html.

Further details about lateness and plagiarism can be found in the Course Introduction.

Acknowledgement
This assignment is based on a 2007 Massey University assignment written by Professor Ken Hawick and has been modified with permission.