ECS40 Fall 2014 2014-12-04

Homework 4 due Thu 2014-12-11 at 23:50

Use the handin directory hw4 to submit your work

RPN calculator

Description

In this assignment (HW4), you will implement a simple Reverse Polish Notation (RPN) calculator. RPN is a notation in which arithmetic expressions are written using operands followed by operators (it is also called postfix notation). For example, the expression `2 + 3` is written in RPN form as

```
2  3  +
```

More complicated expressions can be written in RPN form without the use of parentheses. For example, the expression

```
((2 + 3)*(7 – 4))/5
```

can be written in RPN form as

```
2 3 + 7 4 - * 5 /
```

An RPN calculator can be implemented using a stack to store operands.
The algorithm for evaluating an expression in RPN form can be described as follows. The RPN expression is provided in the form of a string consisting of tokens. Each token is either an operand (a double precision floating point number) or an operator (one of `'+', '-', '*','/'`). The algorithm evaluates the expression using the following steps:

1) Read a token
2) If the token is an operand, push it on the stack
3) If the token is an operator, apply it to the two top-most operands on the stack. Remove the two operands used from the stack. Then store the result of the operation on the top of the stack
4) If no more tokens are available, the final result is the operand at the top of the stack. Otherwise go to 1)

In the example of the expression given above, **`2 3 + 7 4 - * 5 /`** , the stack would have the following contents at each step of the calculation

```
2          // push operand 2
2 3        // push operand 3
5          // apply operator + to 2 and 3, result is 5
5 7        // push operand 7
5 7 4      // push operand 4
5 3        // apply operator - to 7 and 4, result is 3
15         // apply operator * to 5 and 3, result is 15
15 5       // push operand 5
3          // apply operator / to 15 and 5, result is 3
```

The final result is **`3`**

RPN program specification

The RPN calculator program should read the RPN expression as an entire line from **`stdin`**. Input will consist of a single line. After completing the evaluation of the expression, the program should print the contents of the entire stack, starting with the top operand and printing each operand on a separate line. Use the default precision for double precision numbers.

Error conditions

Invalid input

If any operand is invalid, i.e. the token cannot be converted to a **`double`**, or if any operator is not in the set **`{'+','-','*','/'}`**, the program should print

```
invalid input
```

and exit.

Stack underflow

If an operator is read and there are currently fewer than two operands on the stack, the operation cannot be performed. In that case, the program should print

```
stack underflow
```

and exit.

Division by zero

If the operator **`'/'`** is read and the operand currently on the top of the stack is zero, the division cannot be performed. In that case, the program should print

```
division by zero
```

and exit.

The above error messages should be printed on **`stdout`**. Do not use **`stderr`**.

Implementation

The program should use the STL **`stack`** container adapter to store the double precision operands. Use the methods described in Lecture 15 to read entire lines of input and analyze each token to ensure that the input is valid.

HW4 Assignment

You task is to implement the file **`rpn.cpp`**, which contains the entire program. The program should compile without warning using the following command

```
$ g++ -Wall -o rpn rpn.cpp
```

You will use the **`rpn`** executable and the input and output test files provided on the web site http://web.cs.ucdavis.edu/~fgygi/ecs40/homework/hw4 to check the functionality of your program and verify that your program reproduces the test output *exactly*. Use the **`diff`** command to compare your output files with the reference test output files. Note that other test files may also be used when grading your implementation. The **`rpn`** program reads input from **`stdin`** and writes to **`stdout`**. It can be used e.g. as

```
$ ./rpn < test1.in
```

Submission

Create a tar file named **`hw4.tar`** containing the file **`rpn.cpp`** . Submit your project using:

```
$ handin cs40 hw4 hw4.tar
```