CS1321 Homework 8*

Jeffrey D. Oldham

2000 Apr 06

Due Thursday, 2000 Apr 13, at the beginning of class.

Contents

1 Reading

Read chapter 10. Sections 7.4 and and 10.4 on tree traversals may be useful for this homework.

2 C++ Programming Tip: Conditional Compilation

Frequently, programmers want to use the same C++ files to produce different executable programs. For example, when developing code, it is frequently useful to print debugging messages, but, after development finishes, these debugging messages are a nuisance. Thus, the code producing the debugging messages should be conditionally included when compiling the program.

The tree debugging code uses `#ifdef DEBUG` and `#endif` to delimit the statements that should only be executed when debugging a program. `#ifdef DEBUG` abbreviates `#if defined(DEBUG)`, which yields true only if the preprocessor symbol `DEBUG` is defined. `#endif` marks the end of the conditional section.

To tell the compiler that the preprocessor symbol `DEBUG` should be defined and thus the debugging statements should be included in the executable, compile with the `-DDEBUG` compiler flag. For example,

```
g++ -Wall -pedantic -DDEBUG infix2postfix.cc -g -o infix2postfix
```

`-D` precedes the preprocessor symbol to define. Omitting `-DDEBUG` when compiling omits the statements between the `#ifdef` and `#endif`. The `DEBUG` is not a C++ variable, function, or anything else. It is a preprocessor symbol that can only be manipulated using a `-D` compiler flag.

3 Problem Statement

You and possibly a CS1321 classmate are to convert an infix arithmetic expression to postfix notation using a tree as an intermediary data structure. You may solve the problem in any way you desire as long as it conforms to the input and output specifications except that you may not use a stack.

3.1 Input and Output Specifications

Your program should read an infix arithmetic expression from the standard input and print the equivalent postfix arithmetic expression to the standard output.

Infix arithmetic expressions are recursively defined. The simplest arithmetic expression consists of a variable name or a number. More complicated infix expressions have the form

`(` *expression operator expression* `)`

Whitespace must separate each of these five pieces; e.g., whitespace must appear before and after the parentheses. A variable, number, or operator is a whitespace-delimited word not equal to "(".

Postfix arithmetic expressions are similarly defined. The simplest expression consists of a variable name or a number. More complicated postfix expressions have the form

expression expression operator

No parentheses are permitted.

3.2 A Strategy for Solving the Problem

Infix arithmetic expressions, postfix arithmetic expressions, trees, in-order tree traversal, and postorder tree traversal are all recursively defined. Thus, most functions involving them are recursively defined.

Here is a possible sequence of steps to solve the problem:

1. Write a recursively defined function that reads an infix arithmetic expression from an istream, storing it in a tree.
2. Separately, write a recursive function that takes a tree and produces a postfix arithmetic expression. Test this function in a test program by manually constructing trees to see if the results are correct.
3. Combine the two functions together.

4 What Files Do I Need?

We have provided a recursively defined `Tree` class and a skeleton file. Add code to the latter file to solve the problem. For more information about how to use the Tree class, see the instructions for its use and this example of its use.

We have also provided a file containing functions to print a tree. To use this code,

1. Make sure a copy of the file is in your directory.
2. Add `#include "tree-debug.h"` at the top of the file using one of the debugging functions.
3. When compiling, be sure to add `-DDEBUG` as a compiler flag.

5 Rules

5.1 Programming Rules

As for previous homeworks, working with other people during your planning phase is encouraged. For this homework, you are permitted to write the code, i.e., program, with one other person in CS1321. To learn the material, both of you should be actively involved in the programming. Failure to do so will almost certainly hurt your comprehension of the material in the rest of the course.

5.2 Submission Rules

Each one- or two-person team of programmers should submit only its completed implementation, consisting of the file `infix2postfix.cc`. You do not need to send any other files. Please send only text documents, do *not* send Microsoft Word documents, PDF documents, HTML documents, etc. **Please include both your names and email addresses at the top of your program.**

We will test your program using our own input. Please be sure your code compiles without warning when using `g++ -Wall -pedantic`.

See the submission details for information how to email the programs. Note that the CS computer configuration was changed over spring break. If you want to receive an automated reply acknowledging your submission, please read the submission WWW page. If a team of two are in different sections, submit exactly once to one of the two permissible email addresses.