Programming Assignment #6

Match Brackets using a Stack

CS 2308.256 Spring 2017
Instructor: Jill Seaman

Due: Tuesday, 4/18/2017: upload electronic copy by 9:00am.

Given a text file, your program will determine if all the parentheses, braces, and square brackets match, and are nested appropriately. Your program should work for mathematical formulas and most computer programs.

Your program should read in the characters from the file, but ignore all characters except for the following: `{ } ( ) [ ]`

The general algorithm is to use a stack to store the opening unmatched brackets. When a closing bracket is encountered, check it against the one on top of the stack--make sure it matches, and pop it off. When you are finished there should be no unmatched brackets left on the stack.

Your program should first implement a char stack. **CharStack.h** is provided on the website. You must supply the **CharStack.cpp** file that includes the implementations of the functions in the class declaration. Note that the stack elements will be stored in a string, and no variable named top is necessary. You can complete this assignment using these string functions: [n], size(), +=, and erase(int,int). (mystring.erase(x,n) erases n characters from mystring beginning at position x).
You may also use these functions: push_back, pop_back, back.
You can use the provided **CharStackTester.cpp** to test your implementation.

Input/Output:

Your driver program must prompt the user to enter a filename. It should then try to open the file and then check it to ensure the brackets all match appropriately. If they all match, the program should output a message indicating the number of bracket pairs that were matched. If not, the program should output an appropriate error message.

There are three types of errors that can happen (and with any kind of bracket):

missing } : if you reach the end of the file, and there is an opening { that was never matched, like: `int main () { x[size]=10;`
expected } but found) : this is a wrong closing bracket, like: `{x[i]=10;)...`
unmatched } : this occurs if there is a closing bracket but not an opening bracket (not even one of the wrong kind), like: `int main () { x[i]=10; } }...`

This is an example of appropriate output when there is no error:

```
Enter the name of a file to check: CharStack.h
7 bracket pairs were matched
```

NOTES:

- This program must be done in a **Linux or Unix** environment, using a command line compiler like g++. Do not use codeblocks, eclipse, or Xcode to compile.

- Your CharStack.cpp (and driver) files **must compile** with the (unchanged) provided files CharStack.h and CharStackTester.cpp, otherwise you may receive a score of 0.

- Beware of stack underflow! Do NOT try to peek or pop an empty stack! Your peek and pop functions should include a check that will abort the program if the driver calls them on an empty stack. However, when I run your driver program, it should **never** abort/crash (because it will check if the stack is empty before calling pop or peek).

- As soon as your driver encounters an error, your program should stop and print out the appropriate error message. **Do NOT try to keep going and find more errors!**

- It might be easier to store the expected closing character on the stack when an opening bracket is encountered. This simplifies the matching when a closing bracket is encountered.

Logistics:

For this assignment you need to submit two files: CharStack.cpp and your driver. Please zip these into one file, **assign7_xxxxxx.zip** (where xxxxx is your NetID). Then submit the zip file

Submit an electronic copy using the Assignments tool in TRACS before the deadline.

For this assignment, no printout is required! The feedback cover sheet will be posted to the assignments tool in TRACS (or emailed to you) after the assignments are graded (and the grade will be posted to the TRACS gradebook2 tool).