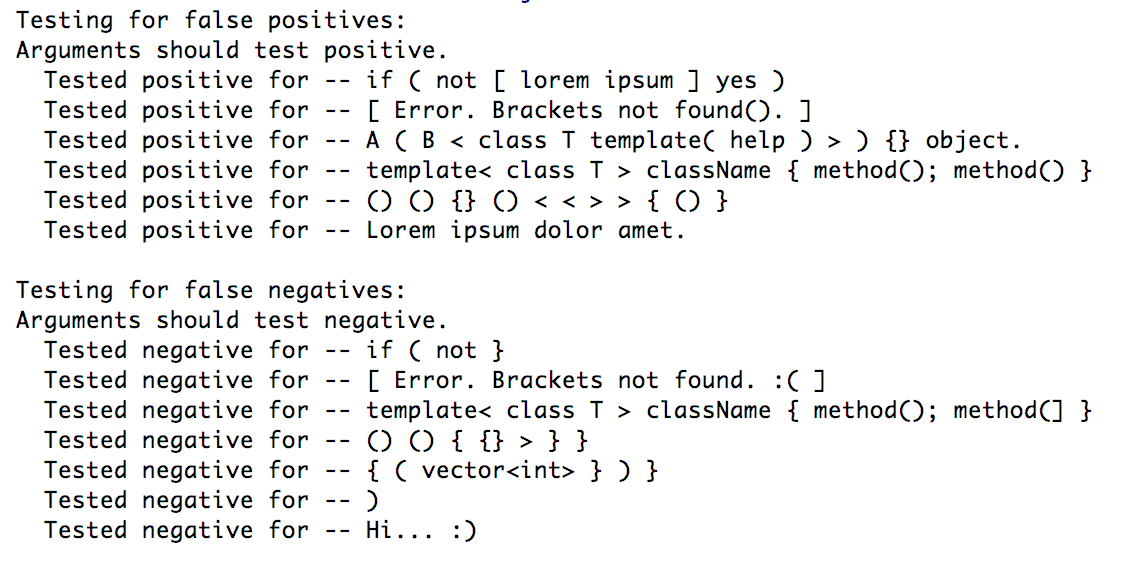
CSC 122 001 Computer Science I

Julius Ranoa

Chapter 18 Programming Challenge 11 Balanced Delimiters

Write a program that tests for the pairing of delimiters (e.g. parenthesis, square brackets, curly brackets).

Screenshot of Runtime:



Files included: (1) DelimiterTester.h, (2) DelimiterTester.cpp, (3) main.cpp

**DelimiterTester.h**

#ifndef **CH18\_PR10\_BALANCED\_PARENTHESIS\_DELIMITERTESTER\_H**#define **CH18\_PR10\_BALANCED\_PARENTHESIS\_DELIMITERTESTER\_H**#include **<string>**#include **<vector>**#include **<stack>  
  
class** DelimiterTester {  
  
**private**:  
 **static unsigned const** nBracketTypes;  
 **static char** startingBrackets[];  
 **static char** endingBrackets[];  
  
 **int** findStartBracket(**char**);  
 **int** findEndBracket(**char**);  
  
**public**:  
 DelimiterTester() {}  
 **bool** testString(std::string);  
};  
  
  
#endif *//CH18\_PR10\_BALANCED\_PARENTHESIS\_DELIMITERTESTER\_H*

**DelimiterTester.cpp**

#include **"DelimiterTester.h"***// Controls***unsigned const** DelimiterTester::nBracketTypes = 4;  
**char** DelimiterTester::startingBrackets[DelimiterTester::nBracketTypes] = {  
 **'('**, **'{'**, **'['**, **'<'**};  
**char** DelimiterTester::endingBrackets[DelimiterTester::nBracketTypes] = {  
 **')'**, **'}'**, **']'**, **'>'**};  
  
*// Private  
  
// This method returns the index of the starting bracket.  
// Returns -1 if not found.***int** DelimiterTester::findStartBracket(**char** c) {  
 **int** index = -1;  
 **for** (**int** i = 0; i < nBracketTypes; i++) {  
 **if** (c == startingBrackets[i]) {  
 index = i;  
 **break**;  
 }  
 }  
 **return** index;  
}  
  
*// This method returns the index of the ending bracket.  
// Returns -1 if not found.***int** DelimiterTester::findEndBracket(**char** c) {  
 **int** index = -1;  
 **for** (**int** i = 0; i < nBracketTypes; i++) {  
 **if** (c == endingBrackets[i]) {  
 index = i;  
 **break**;  
 }  
 }  
 **return** index;  
}  
  
*// Public Methods***bool** DelimiterTester::testString(std::string text) {  
 std::stack<**int**> bracketIndexStack;  
 **char** lastBracket, testEndBracket;  
 **bool** isGood = **false**;  
  
 **for** (**char** s : text) {  
 *// Find character in Starting Brackets* **int** idx = findStartBracket(s);  
 **if** (idx != -1) { *// If first character is a starting bracket.* bracketIndexStack.push(idx);  
 lastBracket = startingBrackets[idx];  
 testEndBracket = endingBrackets[idx];  
 } **else if** (!bracketIndexStack.empty() && s == testEndBracket) {  
 *// If character is not a starting bracket and  
 // the stack is not empty...* bracketIndexStack.pop();  
 **if** (!bracketIndexStack.empty()) {  
 lastBracket = startingBrackets[bracketIndexStack.top()];  
 testEndBracket = endingBrackets[bracketIndexStack.top()];  
 }  
 } **else if** (findEndBracket(s) != -1) {  
 *// If the stack is empty ...  
 // ... and you've found an ending bracket.* bracketIndexStack.push(-1);  
 **break**;  
 }  
 }  
  
 **if** (bracketIndexStack.empty()) {  
 **return true**;  
 } **else** {  
 **return false**;  
 }  
}

**main.cpp**

#include **<iostream>**#include **"DelimiterTester.h"  
  
int** main() {  
 DelimiterTester dl;  
  
 *// FALSE POSITIVE TESTS.* std::string positiveTests[] = {  
 **"if ( not [ lorem ipsum ] yes )"**,  
 **"[ Error. Brackets not found(). ]"**,  
 **"A ( B < class T template( help ) > ) {} object."**,  
 **"template< class T > className { method(); method() }"**,  
 **"() () {} () < < > > { () }"**,  
 **"Lorem ipsum dolor amet."** };  
  
 std::cout << **"Testing for false positives: \n"**;  
 std::cout << **"Arguments should test positive. \n"**;  
 **for** (std::string s : positiveTests) {  
 **if** (dl.testString(s)) {  
 std::cout << **" Tested positive for -- "**;  
 } **else** {  
 std::cout << **" Tested negative for -- "**;  
 }  
 std::cout << s << **"\n"**;  
 }  
 std::cout << **"\n"**;  
  
 *// FALSE NEGATIVE TESTS.  
 // The following functions should return negative.* std::string negativeTests[] = {  
 **"if ( not }"**,  
 **"[ Error. Brackets not found. :( ]"**,  
 **"template< class T > className { method(); method(] }"**,  
 **"() () { {} > } }"**,  
 **"{ ( vector<int> } ) }"**,  
 **")"**,  
 **"Hi... :)"** };  
  
 std::cout << **"Testing for false negatives: \n"**;  
 std::cout << **"Arguments should test negative. \n"**;  
 **for** (std::string s : negativeTests) {  
 **if** (!dl.testString(s)) {  
 std::cout << **" Tested negative for -- "**;  
 } **else** {  
 std::cout << **" Tested positive for -- "**;  
 }  
 std::cout << s << **"\n"**;  
 }  
 std::cout << **"\n"**;  
  
 **return** 0;  
}