Data Structures & Algorithms

Teacher: Dr. Adeel Muzaffar Syed

DATE: 23-09-2024

Assignment # 1

Name #	Enrollment #
Muhammad Sami Shahid	01-131232-066
Mudassir Raiz	01-131232-046

Department of Software Engineering

Muhammad Sami Shahid
01-131232-066

Mudassir Raiz
01-131232-046

Asgn 1
Deadline Tuesday Sep 24, 2024.
Group size: 2

Read a text file containing at least 10 different lines having expressions in it. Ask the user which line to check. Whichever line is selected by the user, the program should tell if the expression is valid or not. Use stacks to perform this check. All conventions are to be followed.

Git-hub link:

Code:

Stack.h:

```
#include <vector>
#include <stdexcept>

template <typename T>

class Stack {
private:
	std::vector<T> elements;

public:
	void push(const T& element) {
		elements.push_back(element);
	}

	void pop() {
		if (!isEmpty()) {
			elements.pop_back();
		}
	}

	T top() const {
		if (!isEmpty()) {
			return elements.back();
		}
		throw std::out_of_range("Stack is empty");
	}

	bool isEmpty() const {
		return elements.empty();
	}
};

bool isValidExpression(const std::string& expression);
void readExpressionsFromFile(const std::string& filename, std::vector<std::string>&
expressions);
```

Main.cpp

```cpp
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include "Stack.h"

using namespace std;

bool isValidExpression(const string& expression) {
	Stack<char> stack;

	for (char ch : expression) {
		if (ch == '(' || ch == '{' || ch == '[') {
			stack.push(ch);
		}
		else if (ch == ')' || ch == '}' || ch == ']') {
			if (stack.isEmpty()) return false;

			char top = stack.top();
			if ((ch == ')' && top != '(') ||
				(ch == '}' && top != '{') ||
				(ch == ']' && top != '[')) {
				return false;
			}
			stack.pop();
		}
	}

	return stack.isEmpty();
}

void readExpressionsFromFile(const string& filename, vector<string>& expressions) {
	ifstream file(filename);
	string line;

	if (file.is_open()) {
		while (getline(file, line)) {
			expressions.push_back(line);
		}
		file.close();
	}
	else {
		cerr << "Unable to open file: " << filename << endl;
	}
}

int main() {
	vector<string> expressions;
	readExpressionsFromFile("expressions.txt", expressions);

	if (expressions.empty()) {
		cout << "No expressions found in the file." << endl;
		return 1;
	}

	cout << "Available expressions:" << endl;
	for (size_t i = 0; i < expressions.size(); ++i) {
		cout << i + 1 << ": " << expressions[i] << endl;
	}
```

```
int choice;
	cout << "Enter the line number to check (1 to " << expressions.size() << "): ";
	cin >> choice;

	if (choice < 1 || choice > expressions.size()) {
		cout << "Invalid choice." << endl;
		return 1;
	}

	string selectedExpression = expressions[choice - 1];
	if (isValidExpression(selectedExpression)) {
		cout << "The expression \"" << selectedExpression << "\" is valid." <<
endl;
	}
	else {
		cout << "The expression \"" << selectedExpression << "\" is invalid." <<
endl;
	}

	return 0;

}
```

expressions.txt:

- (2 + 3) * (4 - 5)
- {5 * [3 + (2 - 1)]}
- (3 + (2 * 4) / {2 - [1 + (2 * 3)]}
- (10 / [5 + (3 - 2)]) + {4 * (1 + 2)}
- (5 + 2) * {[3 - 1) * 2]}
- [(x + y) * z{(a + b) * c} - d(2 * {3 + (4 - 5)[(a + b) * (c - d)]
- {[(x * y) + (z / a)] * b)
- (1 + 2) * (3 + 4)
- {[5 + (6 - 7)] * (8 + 9)}
- (4 + 5] * (6 - 7)

Code Display:

For expression 1 which is valid:

```
Microsoft Visual Studio Debug Console
Available expressions:
1: (2 + 3) * (4 - 5)
2: {5 * [3 + (2 - 1)]}
3: (3 + (2 * 4) / {2 - [1 + (2 * 3)]}
4: (10 / [5 + (3 - 2)]) + {4 * (1 + 2)}
5: (5 + 2) * {[3 - 1) * 2]}
6: [(x + y) * z{(a + b) * c} - d(2 * {3 + (4 - 5)[(a + b) * (c - d)]
7: {[(x * y) + (z / a)] * b)
8: (1 + 2) * (3 + 4)
9: {[5 + (6 - 7)] * (8 + 9)}
10: (4 + 5] * (6 - 7)
Enter the line number to check (1 to 10):
1
The expression "(2 + 3) * (4 - 5)" is valid.

C:\Users\HAMMAD\source\repos\DSA Assign 1\x64\Debug\DSA Assign 1.exe (process 10276) exited with code 0.
Press any key to close this window . . .
```

For expression 5 which is invalid:

```
Microsoft Visual Studio Debug Console
Available expressions:
1: (2 + 3) * (4 - 5)
2: {5 * [3 + (2 - 1)]}
3: (3 + (2 * 4) / {2 - [1 + (2 * 3)]}
4: (10 / [5 + (3 - 2)]) + {4 * (1 + 2)}
5: (5 + 2) * {[3 - 1) * 2]}
6: [(x + y) * z{(a + b) * c} - d(2 * {3 + (4 - 5)[(a + b) * (c - d)]
7: {[(x * y) + (z / a)] * b)
8: (1 + 2) * (3 + 4)
9: {[5 + (6 - 7)] * (8 + 9)}
10: (4 + 5] * (6 - 7)
Enter the line number to check (1 to 10): 5
The expression "(5 + 2) * {[3 - 1) * 2]}" is invalid.

C:\Users\HAMMAD\source\repos\DSA Assign 1\x64\Debug\DSA Assign 1.exe (process 13932) exited with code 0.
Press any key to close this window . . .
```

***********The End***********