Assignment 2

COMP 1123 - Data Structures I

Yaşar University

April 1, 2022

You are expected to implement an algorithm that checks whether the parentheses in a code are properly ordered. You must use the stack data structure which is coded in this course, for the algorithm. Create a function for this assignment. This function must take a char array as the parameter. The function must return 1 if the parentheses are properly ordered, otherwise 0.

Algorithm

The algorithm takes the code to be checked as a string. The algorithm checks all of the characters of the given string. When the algorithm faces an open parenthesis, it pushes it to the stack. Once the algorithm faces the close parenthesis, the stack is popped. If the popped character is null, then it means the parentheses are not ordered properly. We can terminate the algorithm with 0. If the popped character is the matching parenthesis, then we can continue the algorithm. Otherwise, we can terminate the algorithm with 0. Once we reach the end of the string, if the stack is empty, we can return 1. If the stack is not empty, it means the parentheses are not balanced, so we can return 0.

Notes

- Use only C programming language.
- Assume that source codes which are input, has no comment lines or blocks.
- Assume that there is no parenthesis inside any string in the source code.
- Consider '(', '{' and '[' as open parenthesis, ')', '}' and ']' as close parenthesis.
- Upload only the c source code related to assignment 2, do not upload any code about the stack.
- Do not change anything in the stack code which is already coded in the course.
- Any kind of cheating is strictly prohibited.

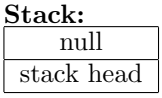
Example

Initial

Input:

```
while(c[i]==pow(a,b)){i++;}
```

Stack:



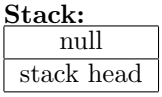
Description:
We start with an initial stack and a string which is input.

Iteration 1

Input:

```
while(c[i]==pow(a,b)){i++;}
↑
```

Stack:



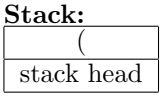
Description:
Because the character is not a parenthesis, we can skip directly.

Iteration 6

Input:

```
while(c[i]==pow(a,b)){i++;}
      ↑
```

Stack:



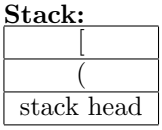
Description:
Because the character is an open parenthesis, we can push it into the stack.

Iteration 8

Input:

```
while(c[i]==pow(a,b)){i++;}
          ↑
```

Stack:



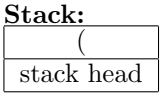
Description:
Because the character is an open parenthesis, we can push it into the stack.

Iteration 10

Input:

```
while(c[i]==pow(a,b)){i++;}
            ↑
```

Stack:



pop: [

Description:
Because the character is a close parenthesis, we can pop from the stack. We can continue because popped parenthesis is the same type of parentheses with the character from the string. If they were not the same type, we would terminate the algorithm with returns zero which means the parentheses are not ordered properly.

Iteration 16

Input:

```
while(c[i]==pow(a,b)){i++;}
                  ↑
```

Stack:

(
(
stack head

Description:
Because the character is an open parenthesis, we can push it into the stack.

Iteration 20

Input:

```
while(c[i]==pow(a,b)){i++;}
                    ↑
```

Stack:

(
stack head

pop: (

Description:
Because the character is a close parenthesis, we can pop from the stack. We can continue because popped parenthesis is the same type of parentheses with the character from the string. If they were not the same type, we would terminate the algorithm with returns zero which means the parentheses are not ordered properly.

Iteration 21

Input:

```
while(c[i]==pow(a,b)){i++;}
                      ↑
```

Stack:

stack head

pop: (

Description:
Because the character is a close parenthesis, we can pop from the stack. We can continue because popped parenthesis is the same type of parentheses with the character from the string. If they were not the same type, we would terminate the algorithm with returns zero which means the parentheses are not ordered properly.

Iteration 16

Input:

```
while(c[i]==pow(a,b)){i++;}
                      ↑
```

Stack:

{
stack head

Description:
Because the character is an open parenthesis, we can push it into the stack.

Iteration 22

Input:

```
while(c[i]==pow(a,b)){i++;}
                       ↑
```

Stack:

stack head

pop: {

Description:
Because the character is a close parenthesis, we can pop from the stack. We can continue because popped parenthesis is the same type of parentheses with the character from the string. If they were not the same type, we would terminate the algorithm with returns zero which means the parentheses are not ordered properly.

Final Iteration

Input:

```
while(c[i]==pow(a,b)){i++;}
                        ↑
```

Stack:

stack head

Description:
We come to the end of the string. We return one as output because the stack is empty.