Data Structures and Algorithms

Pierre Collet/ Manal ElZant-Karray

Practical work n°3 : Stack and Queue

Exercise 1: Write a c program that ask user to enter an expression and use stack to verify if it has a balanced parentheses. (Implement stack with array!)

Example:

Input : (()){}[()] Output : Balanced
Input : ()[]{(}[] Output : Not Balanced

Walk Through:

Ask user to enter the expression then,
Traverse the expression string:

a. If the current character is a starting bracket (**'(' or '{' or '['**) then push it to stack.
b. If the current character is a closing bracket (**')' or '}' or ']'**) then pop from stack and if the popped character is the matching starting bracket then fine else brackets are "not balanced".
c. After complete traversal, if there is some starting bracket left in stack then "not balanced"

Exercise 1: Given a stack of integers, write a c program to sort the stack using a temporary stack. (Implement stack with singly linked list!)

Example: Given Stack: [67, 91, 101, 25]

Sorted Stack: [25, 67, 91, 101]

Approach:

Use another stack called ***temporary stack***.

While given original is not empty:

Pop the element from the original stack, let's call it **tmp**.

While the *temporary stack is not empty* and *top of the temporary stack is greater than the popped element tmp* => pop the element from the temporary stack and push it back to the original stack.

At this point *either temporary stack is empty* or *top of the temporary stack is <= tmp* , so push *tmp* in the temporary stack.

Return the temporary stack, it is sorted.

Walk Through:

Original Stack: [67, 91, 101, 25]

Popped Element from the original stack: 25
Push 25 in the temporary stack
Original Stack: [67, 91, 101]
Temporary Stack: [25]

Popped Element from the original stack: 101
Push 101 in the temporary stack
Original Stack: [67, 91]
Temporary Stack: [25, 101]

Popped Element from the original stack: 91
top of temporary stack (=101) is greater than popped element (= 91)
pop 101 from the temporary stack and push it to the original stack.
Original Stack: [67, 101]
Temporary Stack: [25]
Push 91 in the temporary stack
Original Stack: [67, 101]
Temporary Stack: [25, 91]

Popped Element from the original stack: 101
Push 101 in the temporary stack
Original Stack: [67]
Temporary Stack: [25, 91, 101]

Popped Element from the original stack: 67
top of temporary stack (=101) is greater than popped element (=67)
pop 101 from the temporary stack and push it to the original stack.
Original Stack: [101]
Temporary Stack: [25, 91]
top of temporary stack (=91) is greater than popped element (=67)
pop 91 from the temporary stack and push it to the original stack.
Original Stack: [101, 91]
Temporary Stack: [25]
Push 67 in the temporary stack
Original Stack: [101, 91]
Temporary Stack: [25, 67]

Popped Element from the original stack: 91
Push 91 in the temporary stack
Original Stack: [101]
Temporary Stack: [25, 67, 91]

Popped Element from the original stack: 101
Push 101 in the temporary stack
Original Stack: []
Temporary Stack: [25, 67, 91, 101]
Sorted Stack is: [25, 67, 91, 101]

Exercise 3: Write a program to implement the following operations with the help of a dynamic queue (FIFO):

1. Insert the element.
2. Delete the element.
3. Display.
4. Exit.