STANDARD TEMPLATE LIBRARY STACKS

Problem Solving with Computers-II

freq. AC

C++

```
#include <iostream>
using namespace std;

int main(){
    cout<<"Hola Facebook\n";
    return 0;
}
```

C++STL

- The C++ Standard Template Library is a very handy set of three built-in components:
  - Containers: Data structures → generic data structures
  - Iterators: Standard way to search containers → generic way of parsing/iterating through datastructures
  - Algorithms: These are what we ultimately use to solve problems

C++ STL container classes

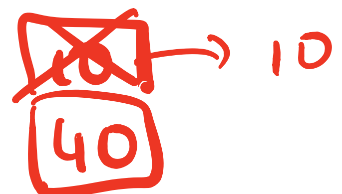
```
array → fixed length array
vector → dynamic
forward_list → single-linked list
list → double-linked list
set → balanced BST
stack }
queue } → today
priority_queue }
multiset (non unique keys) |
deque |
unordered_set |
map | → CS32 and onwards
unordered_map |
multimap |
bitset |
```

Stacks – container class available in the C++ STL

- Container class that uses the Last In First Out (LIFO) principle
- Methods

i. push() → insert to the top of stack
ii. pop() → delete top
iii. top() → get top
iv. empty() → true if empty

push(40)
push(10)
top() // 10
pop()



The datastructure "stack" is different from the "runtime stack" even though both follows the LIFO principle.

Demo reversing a string

Lab05 – part 1: Evaluate a fully parenthesized infix expression

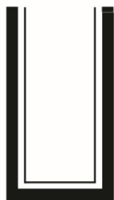
```
( 4 * ( ( 5 + 3.2 ) / 1.5 ) ) // okay

( 4 * ( ( 5 + 3.2 ) / 1.5 ) // unbalanced parens - missing last ')'

( 4 * ( 5 + 3.2 ) / 1.5 ) ) // unbalanced parens - missing one '('

4 * ( ( 5 + 3.2 ) / 1.5 ) // not fully-parenthesized at '*' operation

( 4 * ( 5 + 3.2 ) / 1.5 ) // not fully-parenthesized at '/' operation
```

) 4 + 5 (not balanced

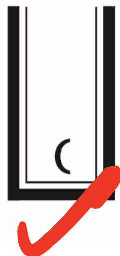
((2 * 2) + (8 + 4))

stack <char> s;
s.push('(');

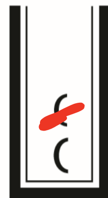
Initial empty stack



Read and push first (

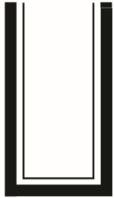


Read and push second (



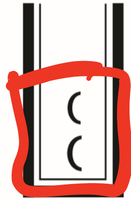
((2 * 2) + (8 + 4))

Initial empty stack



Read and push first (

Read and push second (



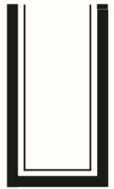
What should **be the next step after the first right parenthesis is encountered**?

A. Push the right parenthesis onto the stack
B. If the stack is not empty pop the next item on the top of the stack
C. Ignore the right parenthesis and continue checking the next character
D. None of the above

```
if ( arr[i] == ')' )
    if(!s.empty())
        s.pop();
    else return false;
```

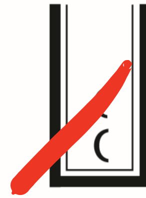
((2 * 2) + (8 + 4))

Initial empty stack

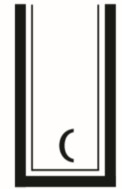


Read and push first (

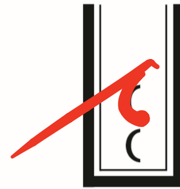
Read and push second (



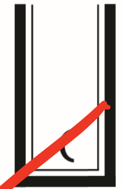
Read first) and pop matching (



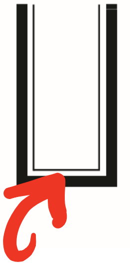
Read and push third (



Read second) and pop matching (



Read third) and pop the last (



Evaluating a fully parenthesized infix expression

(((6 + 9)/3)*(6 - 4))

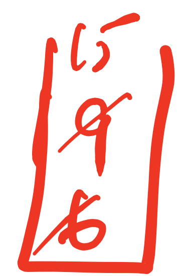
Student's solution demoed in class

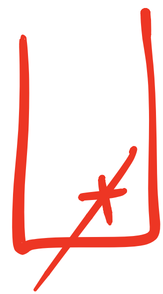
9 + 6

3 / 15

4 - 6

9+6



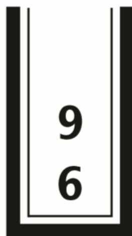


Evaluating a fully parenthesized infix expression

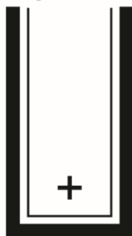
Characters read so far (shaded):

(((6 + 9) / 3) * (6 - 4))

Numbers



Operations



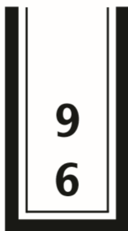
We can evaluate a fully paranthesized expression using two stacks

Evaluating a fully parenthesized infix expression

Characters read so far (shaded):

(((6 + 9) / 3) * (6 - 4))

Numbers Operations



Before computing 6 + 9

6 + 9 is 15

Numbers Operations

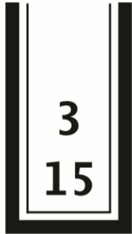
After computing 6 + 9

Evaluating a fully parenthesized infix expression

Characters read so far (shaded):

(((6 + 9) / 3) * (6 - 4))

Numbers



Operations

15 / 3 is 5

Numbers

Operations

Before computing 15/3

After computing 15/3

Notations for evaluating expression

- Infix number operator number
- (Polish) Prefix operators precede the operands
- (Reverse Polish) Postfix operators come after the operands

- 3 * 5
- 4 / 2
- 7 + (3 * 5)
- (7 + (3 * 5)) – (4 / 2)

Prefix (Polish)

* 3 5

/ 4 2

+ 7 * 3 5

- + 7 * 3 5 / 4 2

Postfix (Reverse Polish)

3 5 *

4 2 /

7 3 5 * +

7 3 5 * + 4 2 / -

Small group exercise

Write a ADT called in minStack that provides the following methods

- push() // inserts an element to the “top” of the minStack
- pop() // removes the last element that was pushed on the stack
- top () // returns the last element that was pushed on the stack
- min() // returns the minimum value of the elements stored so far

Summary of operations

Operation	Sorted Array	Binary Search Tree	Linked List
Min			
Max			
Median			
Successor			
Predecessor			
Search			
Insert			
Delete			