# STL (2)

CSE2013-17F

# STL Stack, Queue

- Member functions

| Stack | Queue | definition |
|---|---|---|
| empty | | Test whether container is empty |
| size | | Return size |
| top | front | Access next(last) element |
| | back | |
| push | | Insert element |
| pop | | Remove next element |
| emplace (C++11) | | Construct and insert element |
| swap (C++11) | | Swap contents |

```
#include <stack>
#include <queue>

stack<int> int_stack;
queue<char> char_queue;
```

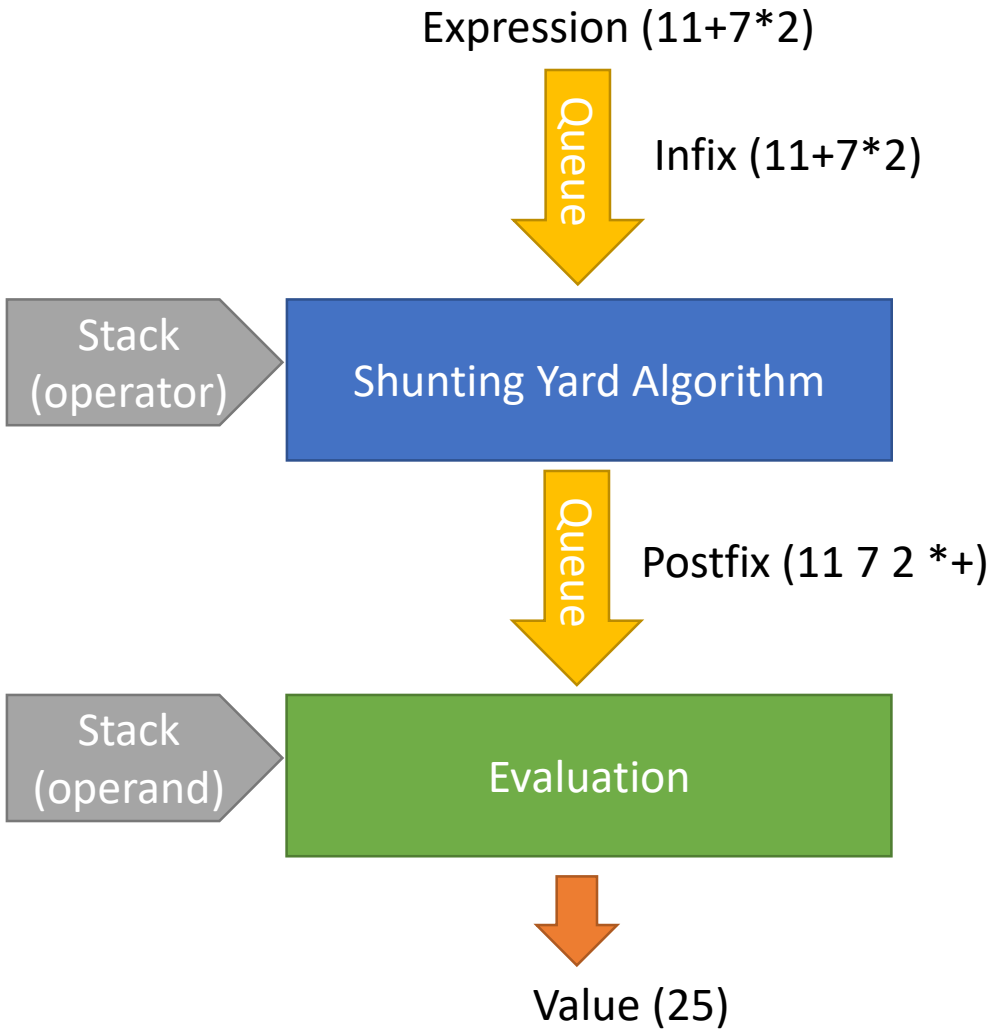
# Lab - Calculator

- Make Calculator
  - Using STL stack & queue
  - Operations { + , - , × , ÷ }

  - Convert infix notation to postfix notation
  - Calculate in order of operator priority
  - Operands are int data type

**11+7*2**
Postfix: 11 7 2 * +
11+7*2 = 25

# Lab - Calculator



# Lab - Calculator

- Shunting Yard Algorithm

Converting infix to postfix :

1. Set postfix string to empty string
2. Create an empty operator stack
3. Repeat
   - 3.1 Get the next token in the infix string
   - 3.2 If next token is an operand (begins with digit), append it to postfix string
   - 3.3 If next token is an operator
     - 3.3.1 Process the next operator

   Until at the end of the string

3.3.1 Process the next operator (next op)

repeat
  If operator stack is empty, push next op onto stack.
  Else if precedence(next op) > precedence(top operator)
    Push next op onto the stack (ensures higher precedence operators evaluated first)
  Else
    Pop the operator stack and append operator to postfix string
Until next op is pushed onto the stack.

operator precedence : * , / 2
+ , - 1

example :
x – y * a + b / c becomes x y a * - b c / +

# Lab - Calculator

```
#include <iostream>
#include <string>
#include <stack>
#include <queue>
using namespace std;

class Calculator {
        string infix;
        stack<int> operand;
        stack<char> operation;
        queue<char> postfix;
public:
        Calculator(string s) {
                infix = s;
        }
        int operPriority(char);         //set operator priority
        void setPostfix();              // converting infix to postfix
        void calculate();
        void printResult();
};
```

```
int Calculator::operPriority(char c) {
        return 0;
}
void Calculator::setPostfix() {

}
void Calculator::calculate() {

}
void Calculator::printResult() {
        cout << infix << " = " << operand.top() << endl;
        operand.pop();
}

int main() {
        string input;
        cin >> input;
        Calculator calc(input);
        calc.setPostfix();
        calc.calculate();
        calc.printResult();

        return 0;
}
```