# Chapter 1: Stack & Quene

## Stack

1. First in last out(FILO)
2. Basic operations
   * 1. Pop():Take the data out from the top of the stack
     2. Push():Put the data to the top of the stack
     3. IsEmpty():Check the stack is empty or not

DIsFull():Check the stack is full or not(especially if the stack is made by array)

1. application(Pre,In,Postfix)
2. Infix

Format:operand1 operator operand2(ex:a+b)

Disadvantage: Calculating with infix notation in the compiler is quite challenging. This is because one has to account for(考慮) the priority and associativity(關聯性) of operators, which may necessitate(需要) multiple scans to compute the final result.

1. Postfix

Format:operand1 operand2 operator(ab+)

Advantage: The compiler only needs to scan from left to right once to produce the result, eliminating the need to worry about operator priority and associativity.

1. Prefix

Format:operator operand1 operand2(+ab)

Advantage: Same as postfix.

1. Infix to postfix conversion algorithm
2. Scan the expression from left to right.
3. If an operand is encountered, print it out.
4. If an operator is encountered, consider the following situation

If the stack is empty.Push()

If the stack contains other operators, compare their priorities with the current operator. If the priority of the operator at the top of the stack is lower, push() the current operator. Otherwise, pop() the stack until the priority at the top of the stack is lower than the current operator's. Do this until the stack is empty or the priority conditions are met.

If a “)” is encountered, pop() the current data in stack, until encounterint ”(“.

## Quene

1. First in first out(FIFO)
2. Basic operations
   1. Dequeue():
3. Stack & Quene