**Ch3 Stack與Queue**

Stack的應用:

1. 處理副程式呼叫(Subroutine Call)
2. 處理遞迴式呼叫(Recursive Call)
3. 中序(Infix)轉後序(Postfix)/前序(Prefix)
4. Compiler執行文法剖析(Parsing)
5. 二元樹的中序追蹤(Inorder)、前序追蹤(Preorder)、後序追蹤(Postorder)
6. Stack Computer機器
7. 機器處理(Re-Entrant Routine) = Pure Procedure = Pure Code

可重複進入執行之程式(需Hardware支援)

1. 圖形的深度搜尋(depth-first search)
2. 資料反序輸出
3. 自助餐廳取餐盤之行為
4. 迷宮問題(maze)
5. 迴文判斷(palindrome)

Stack之操作:

1. Create
2. IsFull
3. IsEmpty
4. Push

Array:

If IsFull(S)

then “Stack is Full”

else

top = top + 1;

S[top] = item;

Link List:

New(t);

t -> Data = item;

t -> Link = top;

top = t;

1. Pop

Array:

If IsEmpty(S)

then “Stack is Empty”

else

item = S[top];

top = top – 1;

return item;

Link List:

If (top == Null)

then “Stack is Empty”

else

t = top;

item = top -> Data;

top = top -> Link;

free(t);

return item;

1. Top

Stack Permutation:

3個Data -> 5種合法

4個Data -> 14種合法

5個Data -> 42種合法

N個Data =

Infix: operand1 operator operand2

Postfix: operand1 operand2 operator

Prefix: operator operand1 operand2

中轉前:2個stack支援

中轉後:1個stack支援

中序轉後序:

1. 對中序式加上完整的括號配對
2. 運算子取代最近的右括號
3. 刪左括號其餘依序寫出

中序轉前序:

1. 同上
2. 取代左括號
3. 刪右括號

Operator之優先權(高到低)

括號、負號、冪次方、\*/、+-、>< <= != ==、否定Not ~、AND OR、Assign =

冪次方、Assign:右結合

\*/、+-、關係、And or:左結合

後序/前序轉中序:

移除不必要的括號判斷

Queue之應用

1. OS中各式的Queue
2. Buffer(緩衝區)design
3. Queue Theory
4. 圖形的BFS
5. Binary Tree的Level-order traversal
6. 日常生活之排隊行為

Queue的種類

1. FIFO Queue
2. Priority Queue

製作此Queue最合適的Data Structure: Heap

1. Double-ended Queue
2. Double-ended priority Queue

基本操作:Insert、Delete-min、Delete-Max

製作此Queue最合適的Data Structure: Min-Max Heap、Deap、SMMH