2.6.3 Application

1) Balancing Symbols

```
int main()
{
        cout << “hello”;
}
```

*Home Work

( ) { " " }

1. ถ้า input เครื่องหมายเปิด push *
2. ถ้า input เครื่องหมายปิด *
   - ถ้า stack ว่าง **error**
   - ถ้าไม่ว่าง ให้ pop จนเจอคู่ และ pop คู่ทิ้ง
   - ถ้า pop แล้วไม่เจอคู่ **error**
3. ถ้า input หมดแล้ว แต่ stack ไม่ว่าง ให้ report **error**

2.6.3 Application

1) Balancing Symbols

```
int main()
{
        cout << “hello”;
}
```

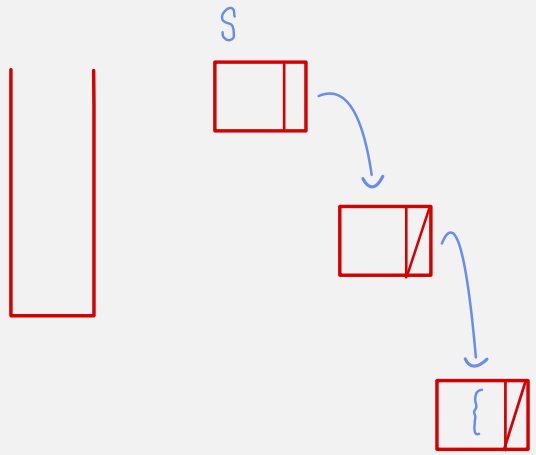
1. Make an empty stack.
2. Read characters until end of file.
3. If the character is an opening symbol, push it onto the stack.
4. If it is a - closing symbol,
   - then if the stack is empty report an **error**.
   - Otherwise, pop the stack.
5. If the symbol popped is not the corresponding opening symbol, the report an error.
6. At end of file, if the stack not empty report an error.

การบ้าน

1) Balancing Symbol

{ } [ ] ( )

* เว้นช่องว่าง 1 space



1) Push

choose > 1 Enter
int char

Fix :

เมื่อรับเลขเสร็จ Enter

```
cin >> choose
getchar();
```

2) Infix and Postfix

* วิธีคำนวณ

Infix 4*2

Postfix 42*

Operter อยู่กลาง

อยู่ท้าย

4 * 2 + 3

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

4*2	= 42*
4*2+3	= 42*3+
4+2*3	=423*+
4+2+3	=42+3+

4 * 2 + 5 + 6 * 3 =

(4 + (2 * 7)) 427*+

* +

4 + 2 * 7 427*+

5 - 3 * 2 + 1 532*-1+

[5 - (3 * 2)] + 1

3 * 2 + 1 - (15 - 7 * 4) / 9

321*+ 1574*-9/-

ลำดับการทำงาน [(3 * 2) + 1] - [(15 - (7 * 4)) / 9]

③ ⑤ ⑥ ② ① ④

*-

32*1+1574*-9/-

3) Infix to Postfix Conversion

Operator + , * , (,)

Example Home Work

- parentheses

a+ b *c + (d * e + f) *g = a b c * + d e * f + g * +

a*b-c+d

a/b+c*d

a-b*c/d

a-b*c+d

Example

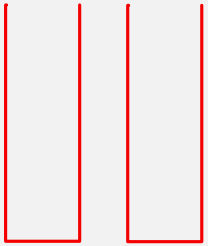
- Stack

a*b-c+d

a/b+c*d

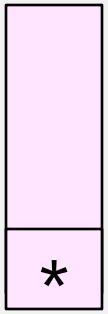
a-b*c/d

a-b*c+d

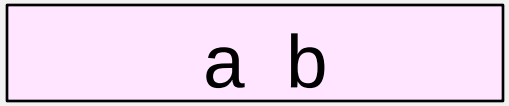


a + b *c + (d * e + f) *g

* เป็น เครื่องหมาย



stack



output

15 - 2 * 71 .

a + b * c + (d * e + f) * g

a b c * + d e * f + g * +

```
#include <iostream>
#include <string>
#include <sstream>
using namespace std;
int main()
{   stringstream ss;
    string str="";
    int num;
    while(str!=".")
    {           cin >> str;
                if(str==".")
                        break;
                if(str=="+")
                        cout << "Push or Pop"<< endl;
                else
```

* แปลง string → int

```
        else    //ตัวเลข
        {       ss << str;
                ss >> num;          // num ไปใช้งานได้แล้ว
                ss.clear();         //ต้อง clear
                cout << ++num << endl;
        }
    }

}
```

เงื่อนไข

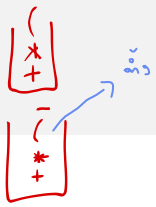
Home Work

1. ถ้า input เป็น operand ให้ print ที่จอภาพ
2. ถ้า input เป็น operator
   2.1 ถ้าเป็น operator ให้เปรียบเทียบ operator ใหม่กับค่าที่อยู่ top ของ stack
   - ถ้าค่าใหม่มี precedence มากกว่า ให้ push ข้อมูลลงใน stack ได้เลย
   - ถ้าค่าใหม่มี precedence **น้อยกว่าหรือเท่ากับ** ให้ pop ข้อมูลมาพิมพ์จนกว่า precedence จะน้อยกว่าค่าใหม่จะน้อยกว่าค่าใน stack หรือ stack empty แล้ว push ค่าใหม่ลงใน stack
   - ถ้าค่าใหม่เป็นวงเล็บเปิด (ให้ push ลง stack ได้เลย และถือว่า precedence มีค่าน้อยที่สุด
   - ถ้าค่าใหม่เป็น วงเล็บปิด) ให้ pop ข้อมูลขึ้นมาพิมพ์จนกว่าจะเจอเครื่องหมาย (

3 2 5 4 2 1 + * - 1 -

3 - 2 * (5 + 4 / 2) - 1

10 + 2 * 5 (5 - 7 * 8) + 4



10 2 5 7 - 8 - * + 4 +

4) Postfix Expressions

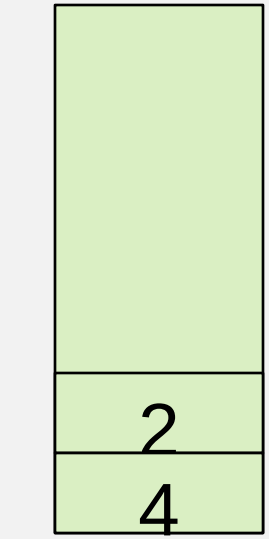
Implementation: Stack
Input number : push onto the stack
Input operator : applied to the two numbers that are poped from the stack.

Home Work

42* = 8
42*3+ = 11
423*+ = 10
42+3+ = 9

* Pop ไม่ย้ายตำแหน่งค่า ถ้าอยากแสดงต้อง Top ก่อน

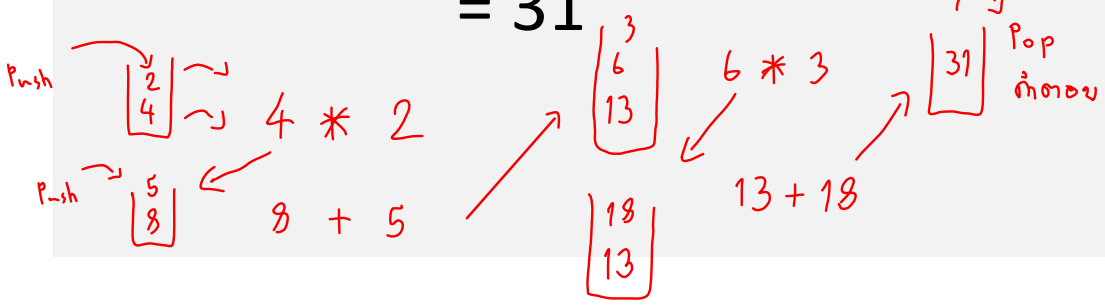
Infix : 4 * 2 + 5 + 6 * 3
Postfix : 4 2 * 5 + 6 3 * +
= 31



42*

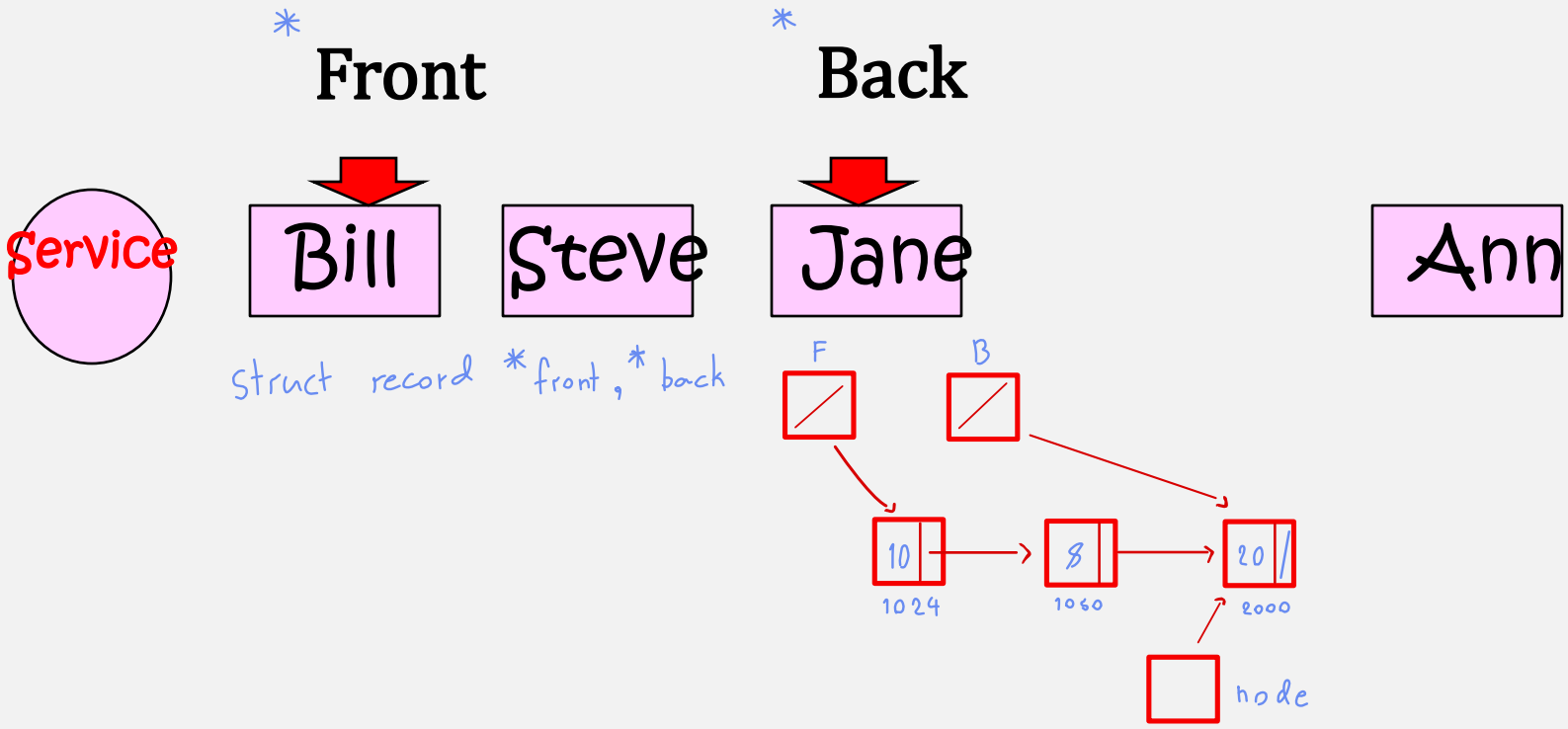
x 4 * y 2

8



ชุดข้อมูลที่มีลำดับ

2.7 Queue are lists. With a queue, however, insertion is done at one end, whereas deletion is performed at the other end. big O = O(1)

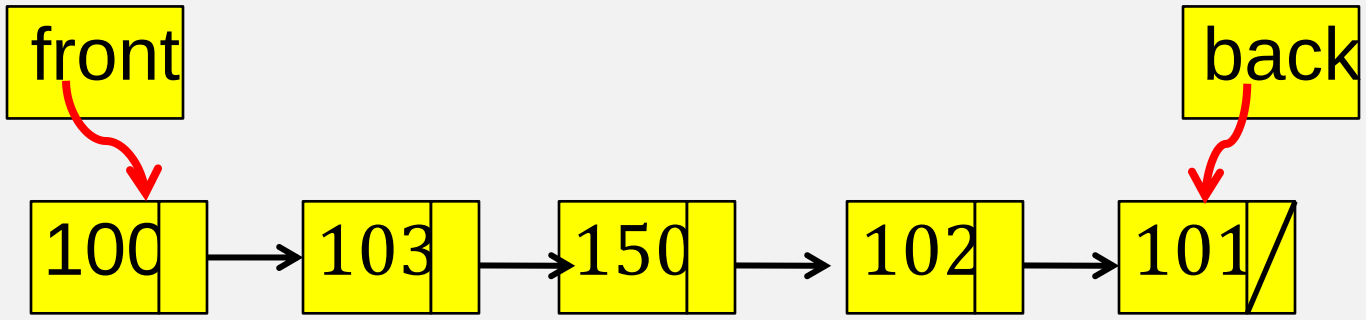


2.7.2 Basic operation

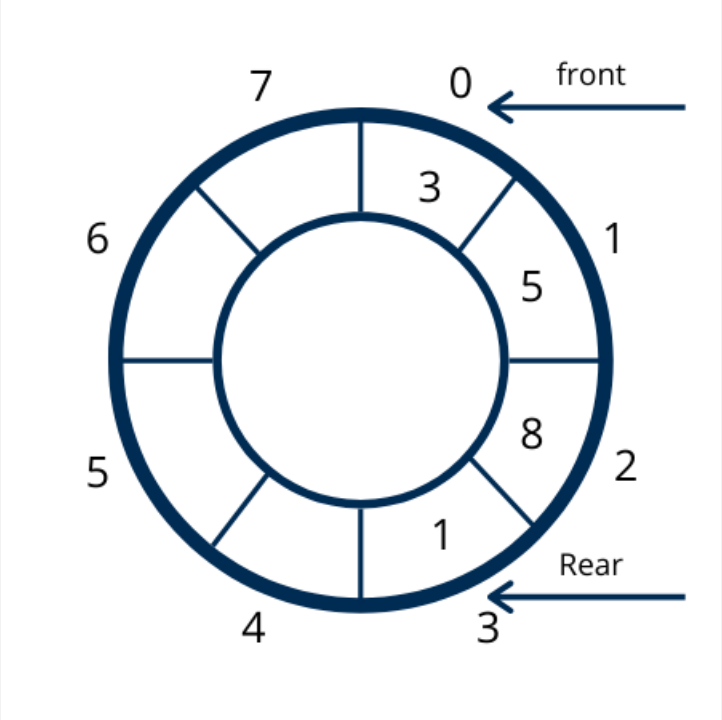
- **Enqueue(x,q)** —Insert item x at the back of queue q.
- **Dequeue(q)** — Return (and remove) the front item from queue q
- **Initialize(q), Full(q), Empty(q)** — Analogous to these operation on stacks

2.7.2 Implementation of queue.

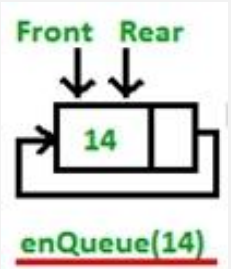
- List (pointer) * ง่าย
- Array * ยาก

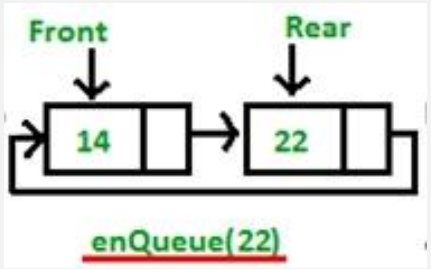


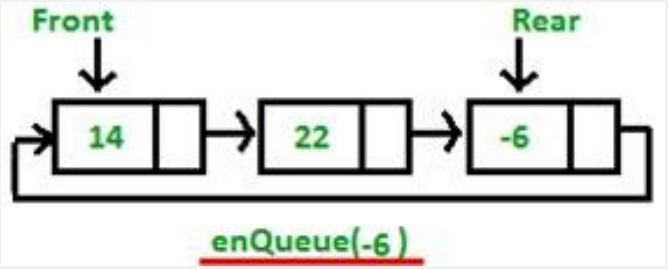
2.7.3 Circular Queue whenever front or back gets to the end of the array. It is wrapped around to the beginning. Array

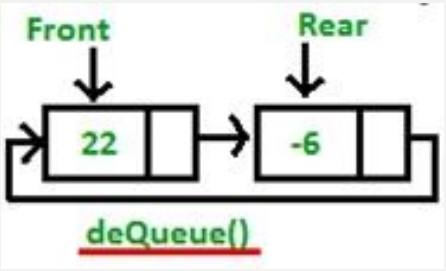


Circular queue : ใช้ Linked list









The josephus problem is the following game:

- N people, numbered 1 to N, are sitting in a circle.
- Starting at person 1, a hot potato is passed.
- After m passed, the person holding the hot potato is eliminated,
- the circle closes ranks,

- and the game continues with the person who was sitting after the eliminate person picking up the hot potato.
  - The last remaining person wins.
  - Thus, if M = 0 and N = 5,