3.5.3 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 Symbols

Input : (){()}
Success
หรือ
Input :)
Error : No open symbol
หรือ
Input : {(]}
Error : Not match
หรือ
Input : (){{()}
Error : stack is not empty

1. ถ้าเป็นเครื่องหมาย**เปิด** push
2. ถ้าเป็นเครื่องหมาย**ปิด**

2.1 **ถ้า stack ว่าง** error "No open symbol"
2.2 **else**
pop
ถ้าตัวที่ pop ไม่ตรงคู่กับเครื่องหมายเปิด
Error "Not match"

3. **ถ้าข้อมูลหมด** แต่ stack ไม่ว่าง error. "Stack is not empty"

ตัวช่วย Balancing Symbols

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

int main()
{    struct record *p1=NULL, *p2=NULL, *p;
     char num;
     cout << "Insert operator example (){()} : ";
     while(num!='\n')
     {    num=getchar();
          if(num != '\n')
           cout << "Push or pop? : "<< num << endl;
          else
         {  …  }
     }
     cout << endl;
}
```

2) Infix and Postfix

Infix 4*2

Postfix 42*

4*2 = 42*

4*2+3 = 42*3+

4+2*3 =423*+

4+2+3 =42+3+

4 * 2 + 5 + 6 * 3 =

3) Infix to Postfix Conversion

Operator + , * , (,)

Example

- 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

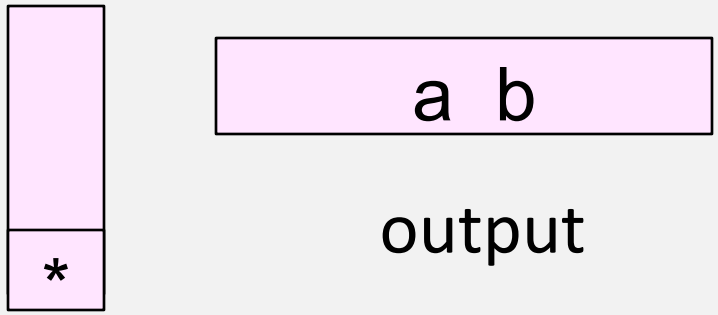
- Stack

a*b-c+d

a/b+c*d

a-b*c/d

a-b*c+d



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

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

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

เงื่อนไข

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 (

4) Postfix Expressions

42* = 8

42*3+ = 11

423*+ = 10

42+3+ = 9

42*

Infix : 4 * 2 + 5 + 6 * 3

Postfix : 4 2 * 5 + 6 3 * +

Postfix expression = 31

Implementation: Stack

Input number : push onto the stack

Input operator : applied to the two numbers that are poped from the stack.

ตัวช่วย infix postfix

```
#include <iostream>
#include <string>
#include <sstream>
using namespace std;
int main()
{    stringstream ss;
     string str="";
     int num;

     cout << "Input example: 123 + 15 + 7 ."<< endl;
     cout << "Enter : ";
```

```
while(str!=".")
{     cin >> str;
      if(str==".")
            break;
      if(str=="+")
            cout << "Operator : "<< str << endl;
      else
      {     ss << str;
            ss >> num;
            cout << "Operand : "<< num << endl;
            ss.clear();      //???? clear
      }
   }
}
```

การบ้าน Postfix Expressions

2. จงเขียนโปรแกรมรับค่าเป็นนิพจน์ postfix ทำการหาผลลัพธ์ โดยมีข้อกำหนดว่า

- Operator ประกอบด้วยเครื่องหมาย + - * /
- Operand จะต้องเป็นตัวเลขจำนวนเต็ม ไม่มีทศนิยม
- แต่ละพจน์จะต้องห่างกัน 1 space และ จบการ input ด้วยเครื่องหมาย .
- ทำการ implement ด้วย stack

Input : 12 7 + .
Output : 17
หรือ
Input : 12 15 2 / + .
Output : 19
หรือ
Input : 12 5 * 200 10 / - .
Output : 60

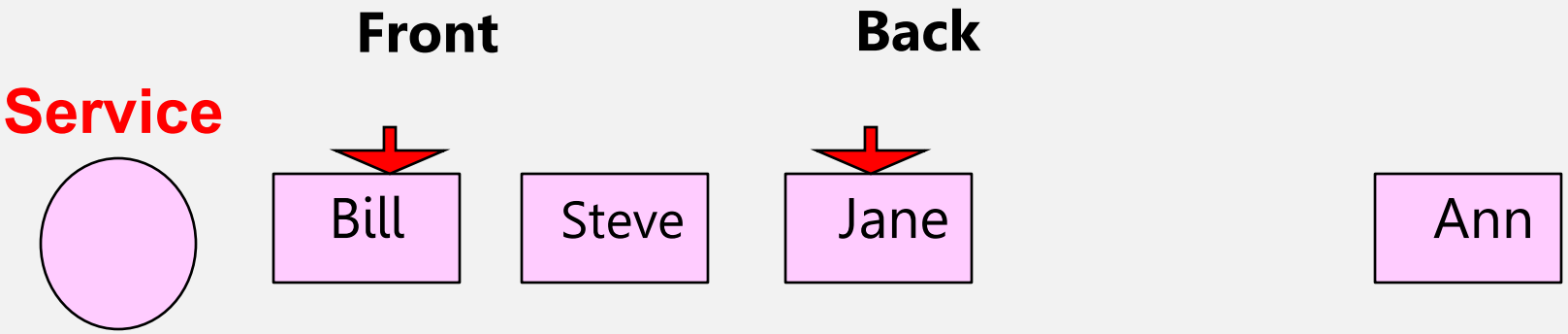
การบ้าน Infix to Postfix

3. จงเขียนโปรแกรมรับค่าเป็นนิพจน์ Infix ให้แปลงเป็น postfix โดยมีข้อกำหนดว่า

- Operator ประกอบด้วยเครื่องหมาย + - * /
- Operand จะต้องเป็นตัวเลขจำนวนเต็ม ไม่มีทศนิยม
- แต่ละพจน์จะต้องห่างกัน 1 space และ จบการ input ด้วยเครื่องหมาย .
- ทำการ implement ด้วย stack

Input : 12 + 7 .
Output : 12 7 + .
หรือ
Input : 12 + 15 / 7 .
Output : 12 15 2 / + .
หรือ
Input : 12 / 5 - 200 / 10 .
Output : 12 5 * 200 10 / - .

3.6 Queue are lists. With a queue, however, insertion is done at one end, whereas deletion is performed at the other end.

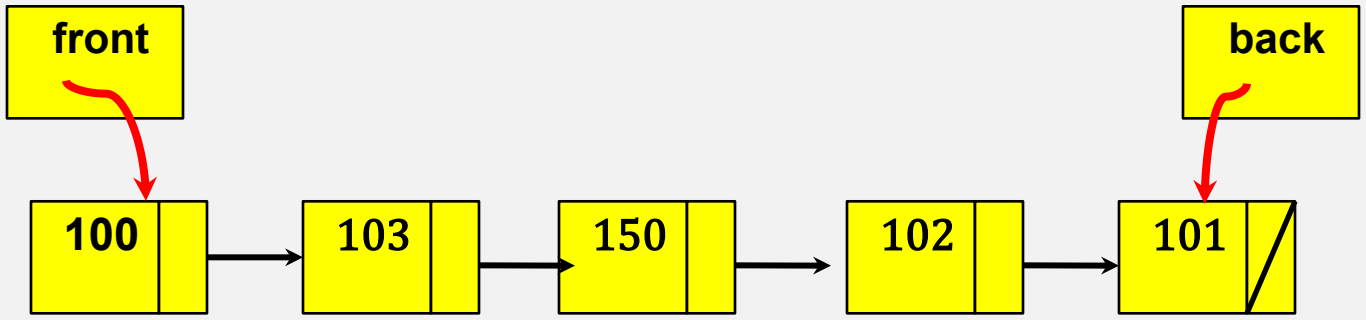


3.6.2 Basic operation

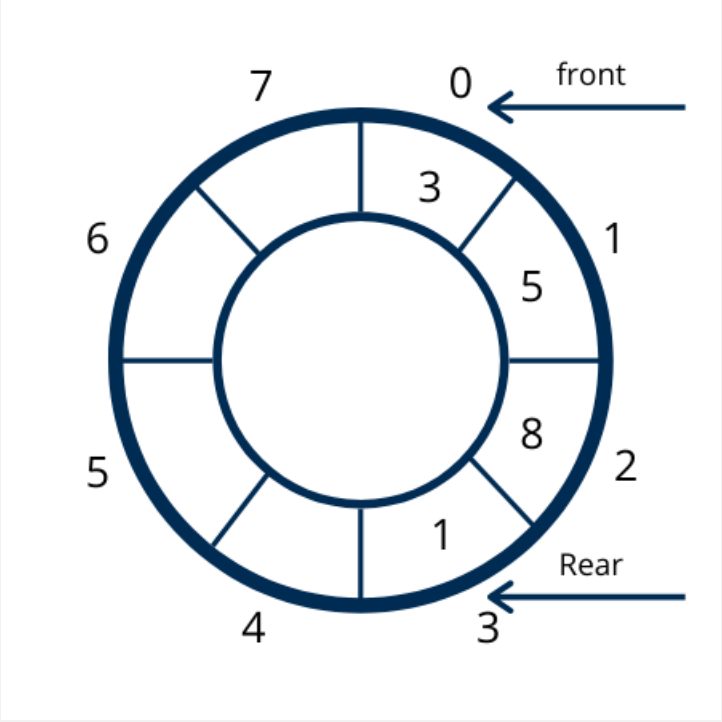
- **Enqueue(x,q)** —Insert item x at the back of queue q.
- **Dequeue(q)** — Return (and remove) the front item from queue q
- **Initialize(q), Full(q), Empty(q)** — Analogous to these operation on stacks.

3.6.3 Implementation of queue.

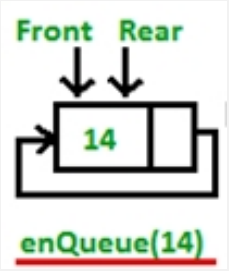
- List (pointer)
- Array

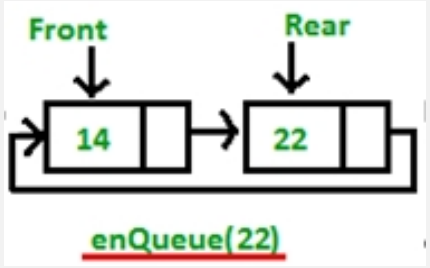


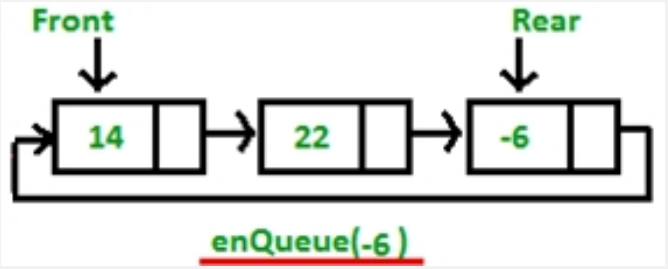
3.6.4 Circular Queue whenever front or back gets to the end of the array. It is wrapped around to the beginning.

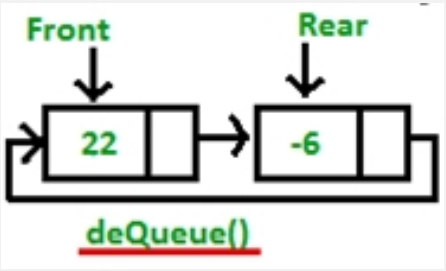


Circular queue : ใช้ Linked list









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,
- players are eliminated in order, and player 5 wins.
- If M=1 and N = 5, the order of elimination is 2,4,1,5.
- Write a program to solve the josephus problem for general values of M and N. Try to make your program efficient as possible. Make sure you dispose of cells.

การบ้าน Josephus problem

4. จงเขียนโปรแกรมเกม Josephus problem รับ input

- จำนวนผู้เล่น ค่า N
- จำนวน pass ค่า M
- แสดงผลลัพธ์เป็นจำนวนผู้เล่นที่จะต้องออกจากเกมและผู้ชนะ
- Implement ด้วย Circular Queue

ตัวอย่าง

Input N = 5
Input M = 1
Eliminate : 2 4 1 3
Winner : 5
หรือ
Input N = 6
Input M = 3
Eliminate : 4 2 1 2 6
Winner : 5