โครงสร้างข้อมูลพื้นฐาน

Interger จำนวนเต็ม int a,b,c=5;

float จำนวนมีทศนิยม float a,b,c=2.5;

boolean ตรรกะ int flag; 0=เท็จ, อื่นๆ=จริง

character ตัวอักษร 1 ตัว char name;

แถวลำดับ (Array)

โครงสร้างข้อมูลแบบแถวลำดับ หรือเรียกว่าตัวแปรชุด คือ การสร้างตัวแปรเพื่อใช้งานเป็นชุด หลายๆตัว เช่น

int a[10]; สร้างตัวแปร a ชนิด interget จำนวน 10 ตัว ประกอบด้วย a[0],a[1] ,a[2] ,a[3] ,a[4] ,a[5] ,a[6] ,a[7] ,a[8] และ a[9]

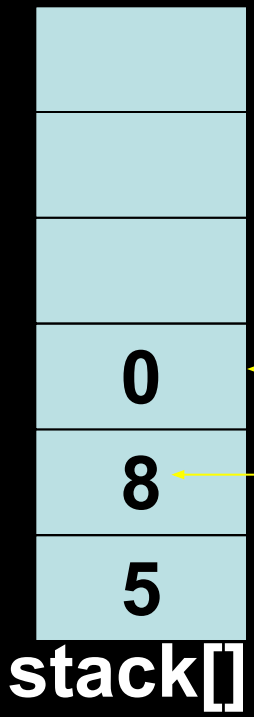
แถวลำดับ (Array)

char name[20]; สร้างตัวแปร name ชนิด character จำนวน 20 ตัว ประกอบด้วย name[0], name[2], name[3], name[4] ... name[18] และ name[19]

Array ของ char มีชื่อเรียกเฉพาะว่า string

โครงสร้างข้อมูล Stack

เป็นโครงสร้างข้อมูลที่เข้าหรือออกได้ทางเดียวคือส่วนบนของ Stack หรือเรียกว่า Top Of Stack ซึ่งมีคุณสมบัติเป็น LIFO (Last In First Out) การเพิ่มข้อมูลเรียกว่า Pushing การนำข้อมูลออกเรียกว่า Poping



max=6

top=3

สมาชิกในสแตก stack[1]

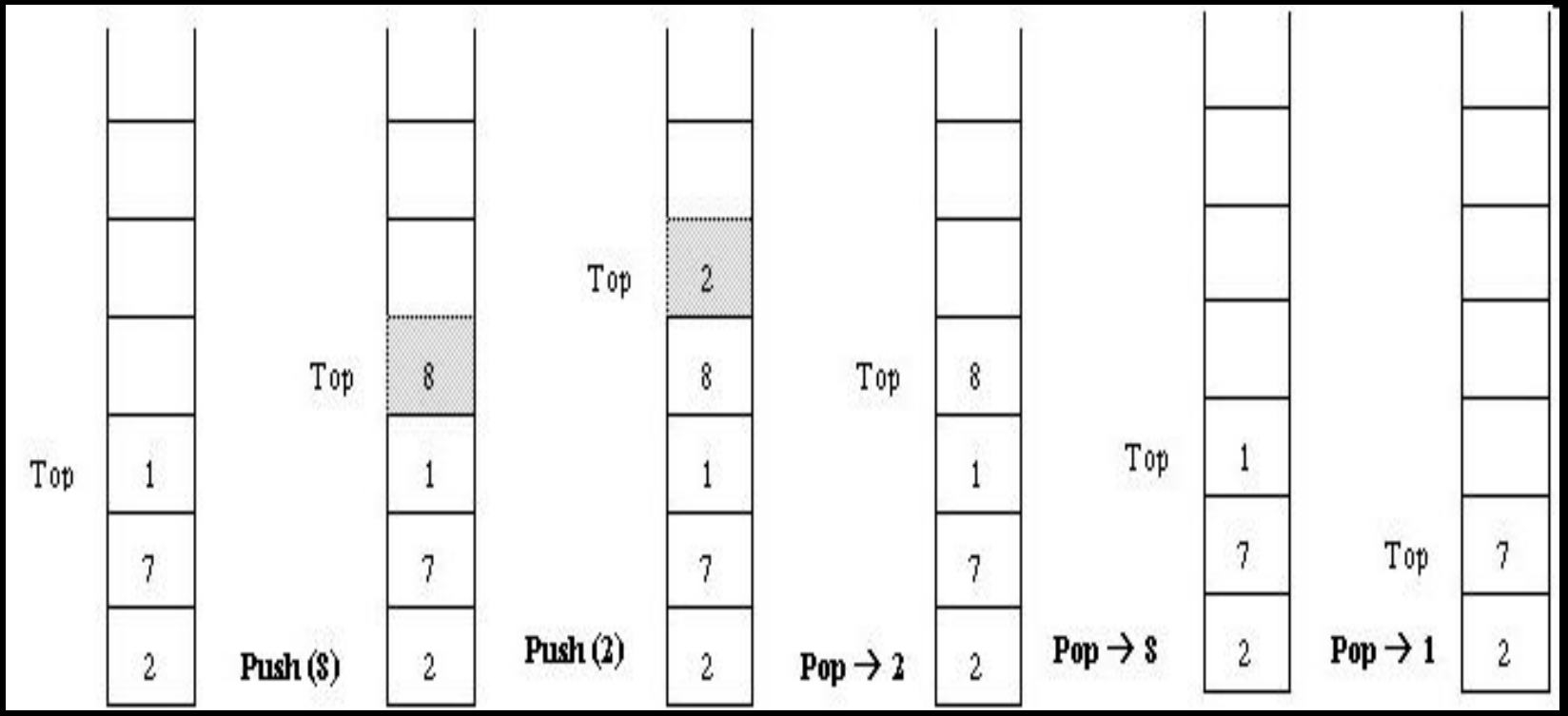
push() ฟังก์ชันสำหรับเพิ่มข้อมูล

pop() ฟังก์ชันสำหรับนำข้อมูลออก

empty() ฟังก์ชันตรวจสอบสแตกว่าง

full() ฟังก์ชันตรวจสอบสแตกเต็ม

Stack


Top
1
7
2
Push (8)
Top
8
1
7
2
Push (2)
Top
2
8
1
7
2
Pop → 2
Top
8
1
7
2
Pop → 8
Top
1
7
2
Pop → 1
Top
7
2

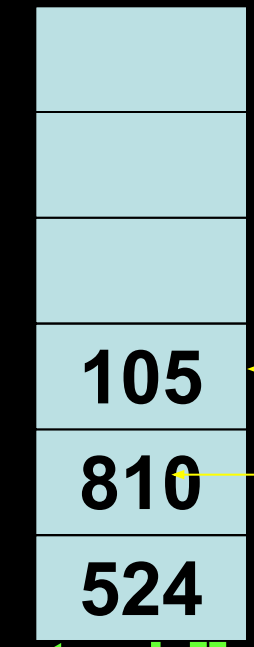
แบบฝึกหัด stack-1

โจทย์ จงเขียนโปรแกรมสร้างโครงสร้างสแตก เก็บเลขจำนวนเต็ม เพิ่มข้อมูลให้สแตก 3 จำนวน และนำข้อมูลออก 3 จำนวน

ข้อมูลนำเข้า เป็นเลขจำนวนเต็ม 3 จำนวน บรรทัดละ 1 จำนวน

ข้อมูลส่งออก บรรทัดเดียว เป็นข้อมูลที่นำออกจากสแตก 3 จำนวน คั่นด้วยช่องว่าง

แบบฝึกหัด



max=10

top=3

สมาชิกใน
สแตก
stack[1]

push() ฟังก์ชันสำหรับเพิ่มข้อมูล
pop() ฟังก์ชันสำหรับนำข้อมูลออก
empty() ฟังก์ชันตรวจสอบสแตกว่าง
full() ฟังก์ชันตรวจสอบสแตกเต็ม

ตัวแปร stack[10] เก็บเลขจำนวนเต็ม

max จำนวนสูงสุดที่สแตกรับได้

top ตัวชี้ข้อมูลที่อยู่บนสุด

push(), pop(),empty(),full()

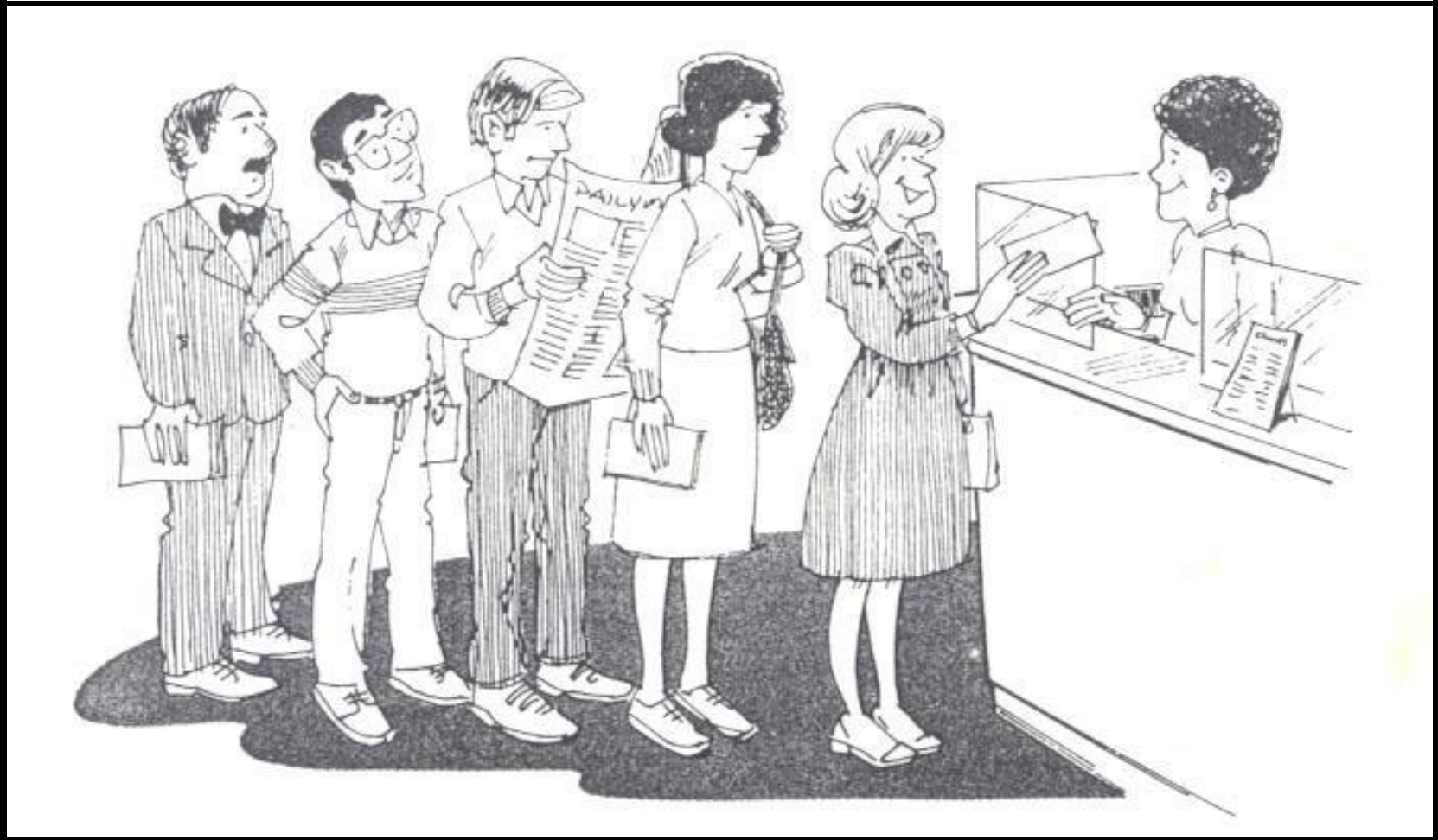
แบบฝึกหัด stack-4

โจทย์ จงเขียนโปรแกรมสลับตัวเลข

ข้อมูลนำเข้า เป็นตัวเลข 5 ตัว บรรทัดละ 1 ตัว

ข้อมูลส่งออก บรรทัดเดียว เป็นตัวเลข

ที่นำออกจากสแตก 5 ตัว คั่นด้วยช่องว่าง



ลักษณะของโครงสร้างข้อมูลแบบ Queue

- ข้อมูลที่เก็บใน Queue จะเก็บในลักษณะเรียงต่อกันไปเช่นเดียวกับการเข้าคิวทั่วๆ ไป การนำข้อมูลออกจาก Queue จะเริ่มจากข้อมูลที่เก็บไว้ใน queue ตัวแรกไปตัวสุดท้าย ตามลำดับ การทำงานลักษณะนี้เรียกว่า FIFO (first-in-first-out)

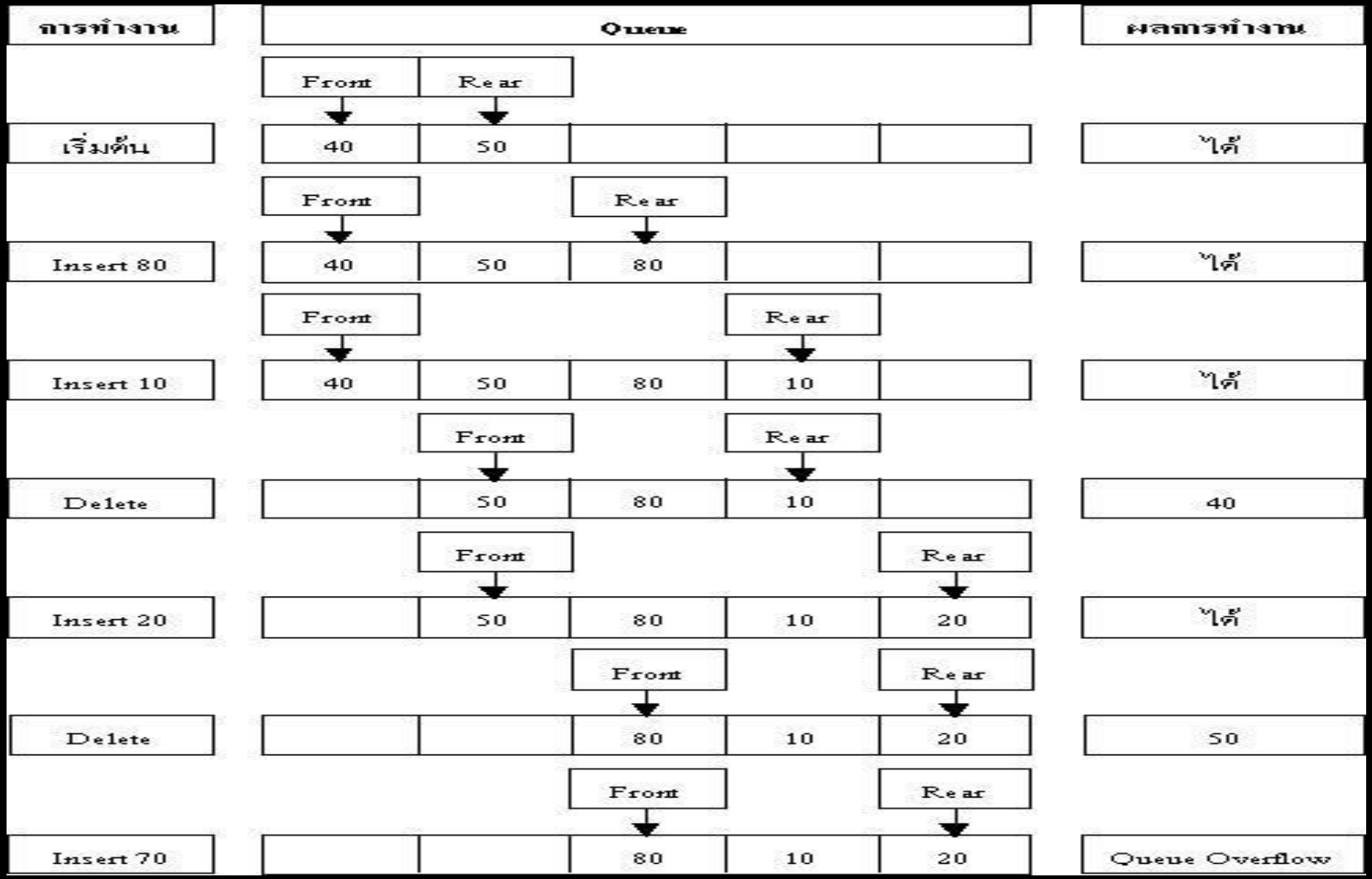
Operation ของ Queue

1. Insert เป็น operation สำหรับนำข้อมูลเก็บไว้ใน Queue
2. Delete เป็น operation สำหรับนำข้อมูลออกจาก Queue

ในการ Insert และ Delete จะมีตัวแปร 2 ตัวแปร คือ

1. ตัวแปรสำหรับเก็บตำแหน่งของข้อมูลที่อยู่ต้น Queue (Front)
2. ตัวแปรที่เก็บตำแหน่งของข้อมูลที่อยู่ท้าย Queue (Rear)

ตัวอย่างการทำงานของ Operation Insert และ Delete (กำหนดให้ข้อมูลใน Queue มี 5 จำนวน)



โครงสร้างของคิว (Queue) ประกอบด้วย

1 อะเรย์ 3 ตัวแปร 2 ฟังก์ชัน

```
int queue[5]; int front=-1,rear=-1,max=4;
int ins(int data) {}              int del(){}
```

โครงสร้างของสแตก (Stack) ประกอบด้วย

1 อะเรย์ 2 ตัวแปร 4 ฟังก์ชัน

```
int stack[5]; int top=-1,max=4;
int full() {}              int push(int data) {}
int empty(){}          int pop(){}
```

Implementation Queue

- การ Implement Queue ทำได้ 2วิธีคือ
  1. Array Implementation
  2. Linked List Implementation

Insert *Algorithm*

- **Array Implementation** **Insert** ***Algorithm***

1. ตรวจสอบว่า Queue เต็ม ? (Rear = N)

 - ถ้า Queue เต็ม ให้แสดงข้อความว่า "Queue Overflow" แล้วเลิกงาน

 - ถ้า Queue ไม่เต็ม ให้ทำงานข้อที่ 2 และ 3

2. ให้เพิ่มค่าของ Rear อีก 1

3. ใส่ข้อมูลลงใน Queue ในตำแหน่งของตัวแปร Rear

ตัวอย่างโปรแกรมนี้เป็น Queue ของ Integer

- ***Program*** C,C++

```
Insert ( int X )
{
   if (Rear = = N)
   printf ( " Queue Overflow " );
   else
   {
   Rear = Rear + 1 ;
   Queue [ Rear ] = X ;
   }
}
```

Delete *Algorithm*

- **Array Implementation** **Delete** ***Algorithm***

  1. ตรวจสอบว่า Queue ว่าง ? (โดยการตรวจสอบว่า Front = Rear)
     - ถ้า Queue ว่าง ให้แสดงข้อความว่า "Queue Empty" แล้วเลิกงาน
     - ถ้า Queue ยังมีข้อมูล ให้ทำงานข้อที่ 2 และ 3
  2. ให้นำข้อมูลในตำแหน่งที่ Front ออกจาก Queue
  3. เพิ่มค่าของตัวแปร Front

ตัวอย่างโปรแกรมนี้เป็น Queue ของ Integer

- ***Program*** C,C++

```
int Delete ( )
{    int x;
     if (Front = = Rear)
     printf ( " Queue Empty " );
     else
     {
     Front = Front + 1 ;
     X = Queue [ Front ] ;
     }
      return x;
}
```

ตัวอย่างโปรแกรม Queue

```
#include <stdio.h>
int queue[7],front=-1,rear=-1,data,n=6;
int insertq(int x)
{    if (rear==n)
     {   printf("Queue overflow!");  }
     else
     {   rear=rear+1;
         queue[rear]=x;  }
}
int deleteq()
{   int x;
    if (front==rear)
    {    printf("Queue empty!");  }
    else
    {    front=front+1;
        x=queue[front];     }
    return x;
}
int main()
{
    insertq(5);
    insertq(8);
    printf("%d/n",deleteq());
    return 0;
}
```

แบบฝึกหัด

- จงเขียนโปรแกรมกำหนดโครงสร้างคิวตัวเลข ขนาด 20 ช่อง และประมวลผลดังนี้
- ข้อมูลนำเข้า รับข้อมูลเข้า บรรทัดละ 1 จำนวน จนกว่าจะพบเลข 999 เก็บในคิว
- ข้อมูลส่งออก ลบข้อมูลออกจากคิวทีละตัวจนหมด แสดงข้อมูลที่ลบคั่นด้วยช่องว่าง

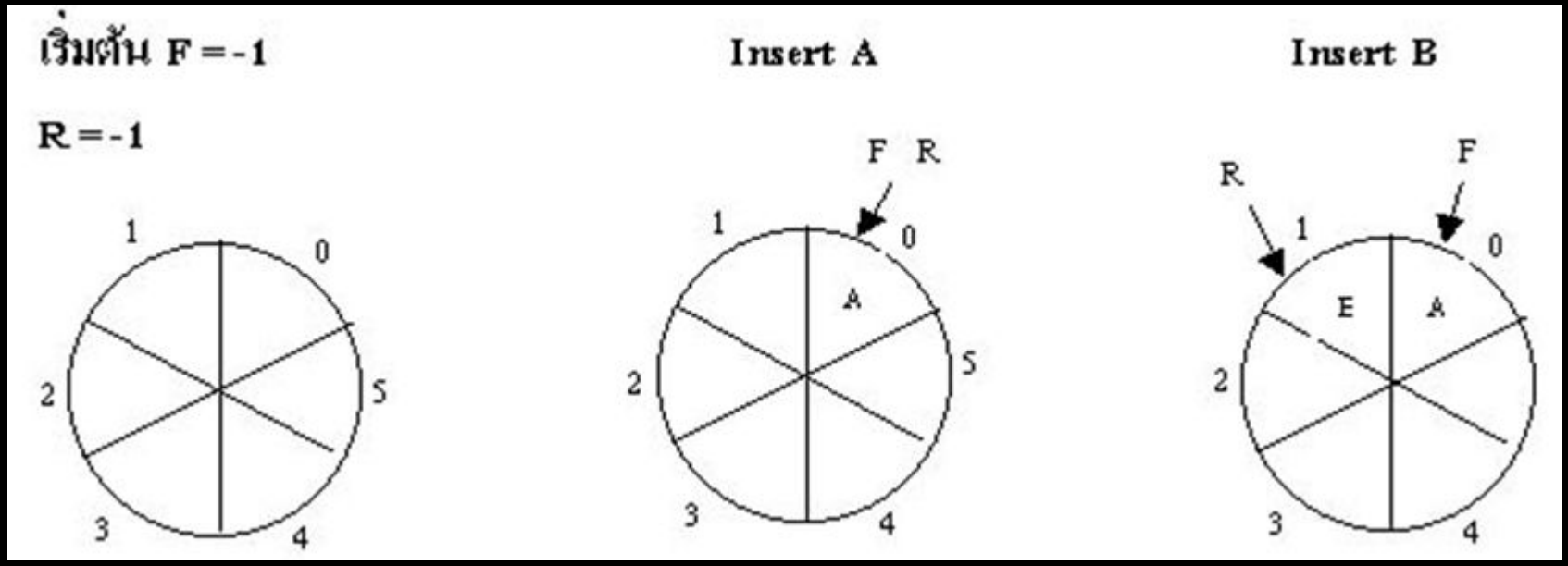
การแก้ปัญหา Queue overflow

- Circular Queue
- Link List

Circular Queue

- Circular Queue หรือ คิววงกลม ใช้เพิ่มประสิทธิภาพในการทำงานของ Queue ซึ่ง มีข้อจำกัดคือ ปัญหา Queue Overflow เมื่อพื้นที่ในการทำงานถูกใช้หมดแล้ว แนวคิดของคิววงกลมคือนำด้านหน้าคิวและด้านท้ายคิวมาต่อกัน ลักษณะเป็นวงกลม

Circular Queue



Circular Queue

- การทำงานของ Circular Queue มีลักษณะเป็น FIFO (First In First Out) เข้าก่อนออกก่อน
- เมื่อคิวมีช่องว่าง Insert ข้อมูลได้
- เมื่อคิวมีข้อมูล (ไม่ว่าง) Delete ข้อมูลได้
- การเลื่อนลำดับจะเพิ่มทีละ 1 ยกเว้นเมื่ออยู่ที่จุดสุดท้ายจะวนมายังตำแหน่งแรก

Circular Queue

- Algorithm Insert

-ตรวจสอบว่า Queue เต็มหรือไม่ (front=rear+1)

-ถ้าเต็ม Queue Overflow

-ถ้าไม่เต็ม

- เลื่อน Rear

- นำข้อมูลใส่ใน Queue

-ถ้า front = -1 ให้ front = 0 (กรณีข้อมูลตัวแรก)

Circular Queue

- Algorithm Delete

-ตรวจสอบว่า Queue ว่างหรือไม่ (front=-1)

-ถ้าว่าง Queue Empty

-ถ้าไม่ว่าง

- นำข้อมูลออกจาก Queue
- เป็นข้อมูลตัวสุดท้ายหรือไม่ (front==rear)

-ถ้าใช่ กำหนด front = -1 , rear = -1

-ถ้าไม่ใช่ เลื่อน front

ตัวอย่าง

- โปรแกรมภาษาซี กำหนดโครงสร้างคิววงกลม ขนาด 12 ช่อง ในฟังก์ชัน main ไม่ต้องมีคำสั่งใดๆ

circular queue

```
 #include <stdio.h>
int cirqdata[12];
int front=-1,rear=-1,max=11;
int ins(int data)
{   if(front==rear+1) { printf("Queue overflow"); }
    else {  rear++;
            cirqdata[rear]=data; }
    if(front==-1) { front=0; }
}
int del()
{   int data=999;
    if(front==-1) { printf("Queue empty"); }
    else {  data=cirqdata[front];
            if(front==rear) {front=-1; rear=-1;}
            else { front++; }
    }
    return data;
}
int main()
{
    return 0;
}
```

แบบฝึกหัด

- จงเขียนโปรแกรมกำหนดโครงสร้างคิววงกลมเก็บตัวเลขขนาด 4 ช่อง และประมวลผลดังนี้
- ข้อมูลนำเข้า รับข้อมูลเข้า บรรทัดละ 1 จำนวน จนกว่าจะพบเลข 999 เก็บในคิว
- ข้อมูลส่งออก ลบข้อมูลออกจากคิวทีละตัวจนหมด แสดงข้อมูลที่ลบคั่นด้วยช่องว่าง

แบบฝึกหัด

- จงเขียนโปรแกรมกำหนดโครงสร้างคิววงกลมเก็บตัวเลข ขนาด 12 ช่อง และประมวลผลดังนี้
- ข้อมูลนำเข้า แบ่งเป็น 3 กรณี
  - –a <ตัวเลข> รับข้อมูลเข้าคิว
  - –d ลบข้อมูลออกจากคิว
  - –x จบการทำงาน
- ข้อมูลส่งออก หลังจบการทำงานให้ลบข้อมูลออกจากคิวทีละตัวจนหมด แสดงข้อมูลที่ลบคั่นด้วยช่องว่าง