บทที่ 8 Queue

บทเรียนย่อย

8.1 Queue Operations and Concept

8.2 Queue with Array Component

8.3 Queue with Array Implementation

8.4 Queue with Pointer Component

8.5 Queue with Pointer Implementation

วัตถุประสงค์

- นิสิตมีความรู้ และความเข้าใจเกี่ยวกับแนวคิด และองค์ประกอบสำคัญต่าง ๆ ในการจัดการโครงสร้างข้อมูลในรูปแบบของ Queue
- นิสิตสามารถเขียนโปรแกรมเพื่อดำเนินการตามแนวคิดของ Queue
- นิสิตสามารถนำแนวคิดของ Queue มาประยุกต์ใช้งานในการพัฒนาโปรแกรม

บทที่ 8 Queue

บทเรียนย่อย

8.1 Queue Operations and Concept

8.2 Queue with Array Component

8.3 Queue with Array Implementation

8.4 Queue with Pointer Component

8.5 Queue with Pointer Implementation

8.1 Queue Operations and Concept

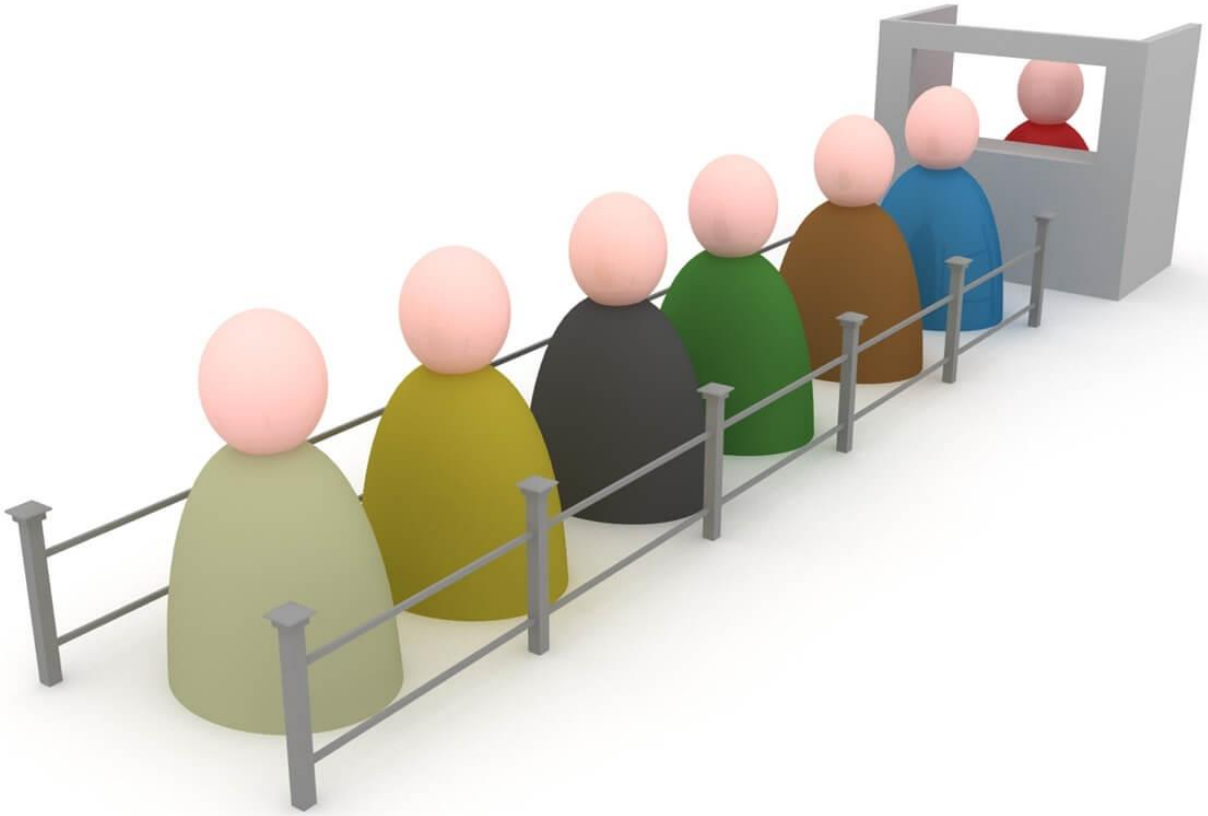
Queue (คิว) คือ โครงสร้างข้อมูลชนิดหนึ่งที่ออกแบบมาให้มีลักษณะทั้งแบบ Linear Structure (เชิงเส้น) และแบบ Non Linear Structure (แบบไม่เชิงเส้น) ซึ่งนำไปใช้กับการเขียนโปรแกรมได้ ทั้งการใช้ Array (อาร์เรย์) และ Pointer (พอยเตอร์) โดยการนำข้อมูลเข้าและออกจากคิวจะมีลำดับการทำงานแบบ “เข้าก่อนออกก่อน” (First In First Out) หรือเรียกสั้น ๆ ว่า FIFO โดยจะเหมือนกับการต่อแถว โดยที่ข้อมูลลำดับแรกของแถวจะได้ออกก่อน และข้อมูลที่เข้ามาต่อแถวจะอยู่หลังสุด

ลักษณะของ Queue

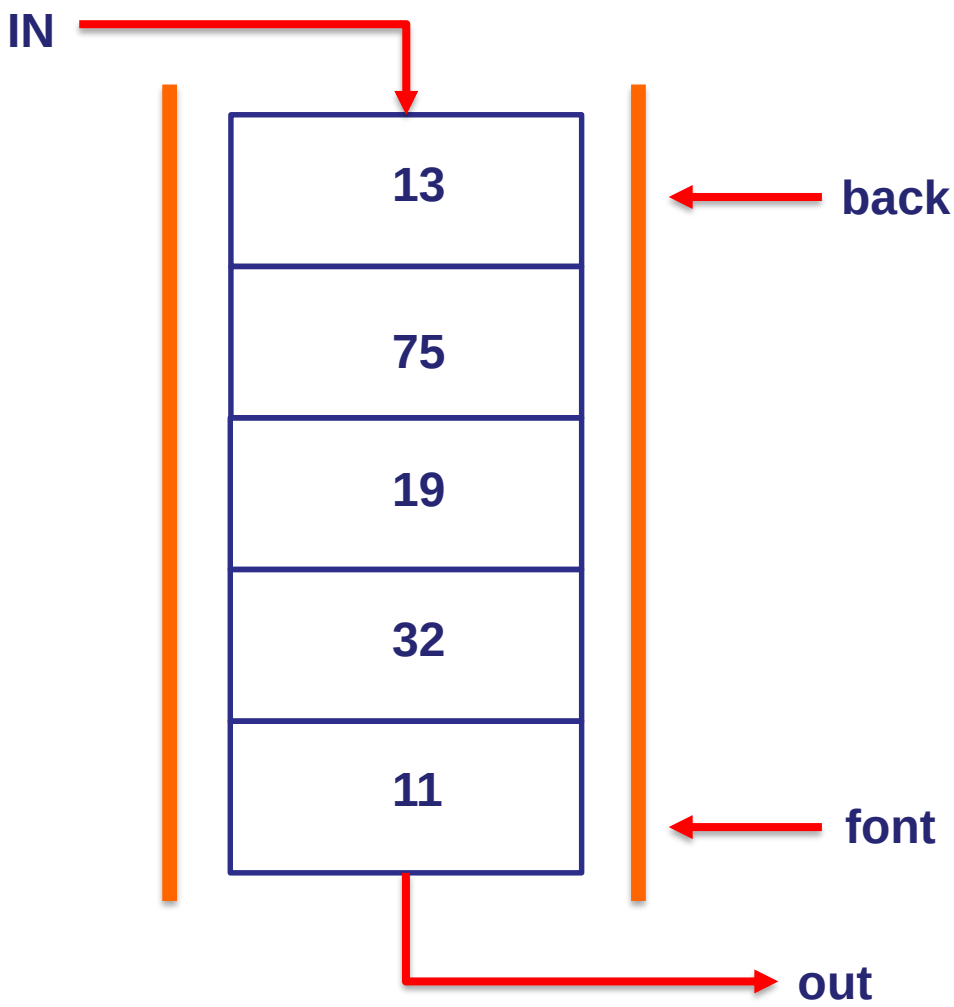
ลักษณะสำคัญของ Queue มีดังนี้

- เป็นโครงสร้างข้อมูลทั้ง 2 ชนิด คือ Linear Structure (เชิงเส้น) และแบบ Non Linear Structure (แบบไม่เชิงเส้น)
- มีทางเข้าข้อมูลอยู่ด้านท้ายและออกของข้อมูลอยู่ด้านหน้า
- มีการทำงานแบบตามลำดับ
- สามารถนำข้อมูลเข้าและนำข้อมูลออกสลับกันได้
- มีลำดับการทำงานแบบเข้าก่อนออกก่อน (FIFO)

ลักษณะของ Queue [2]



ลักษณะของ Queue [3]

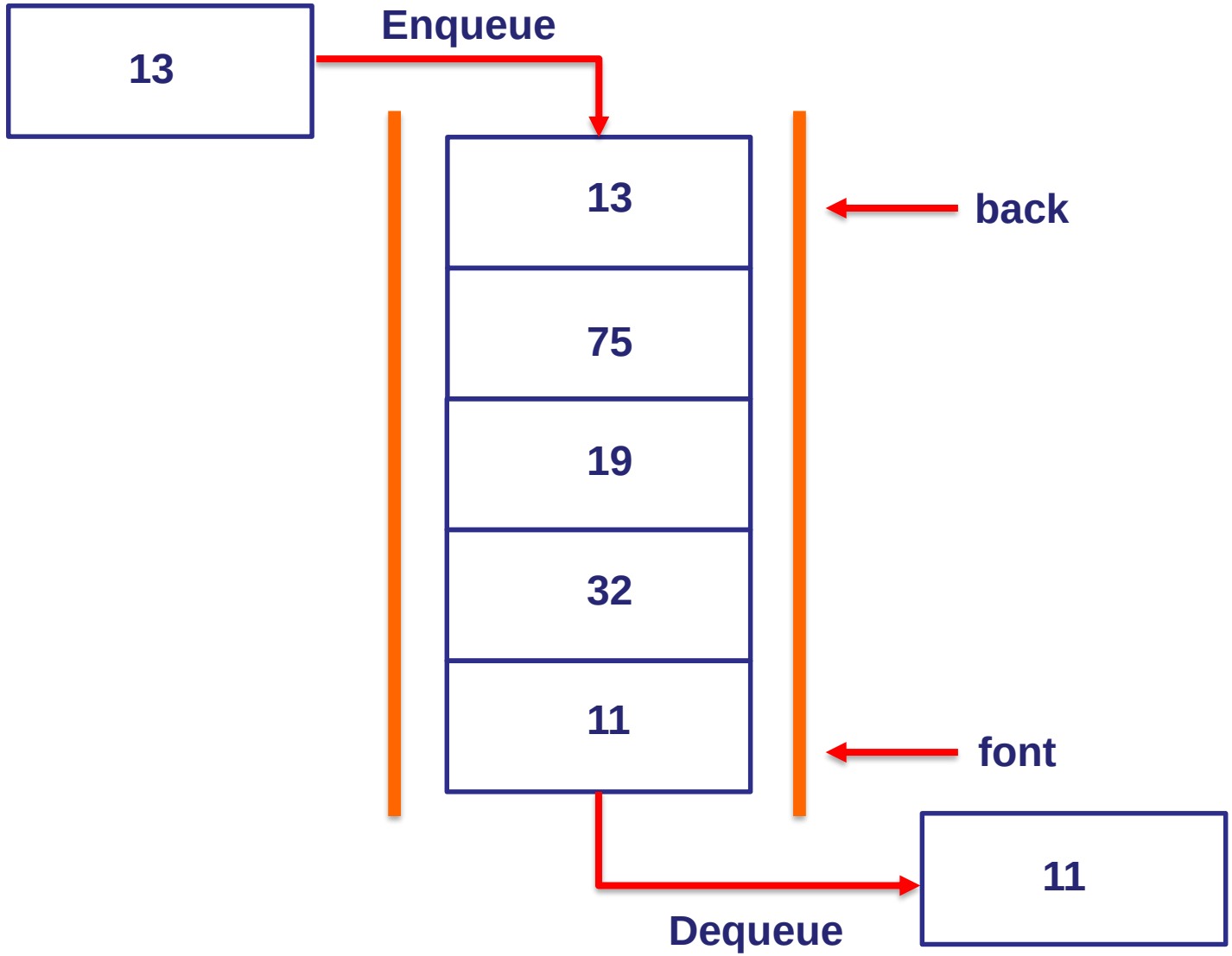


การดำเนินการของ Queue

การดำเนินการของ Queue มี 2 ส่วน ดังนี้

- การนำข้อมูลเข้าไปเก็บในคิว ซึ่งเรียกว่า Enqueue (การเอ็นคิว) คือ การนำข้อมูลไปเก็บแบบเรียงลำดับ (ต่อท้าย) ในคิว
- การนำข้อมูลออกจากคิว ซึ่งเรียกว่า Dequeue (การดีคิว) คือ การนำข้อมูลตัวแรกของคิวออกไปใช้งาน

การดำเนินการของ Queue [2]



บทที่ 8 Queue

บทเรียนย่อย

8.1 Queue Operations and Concept

8.2 Queue with Array Component

8.3 Queue with Array Implementation

8.4 Queue with Pointer Component

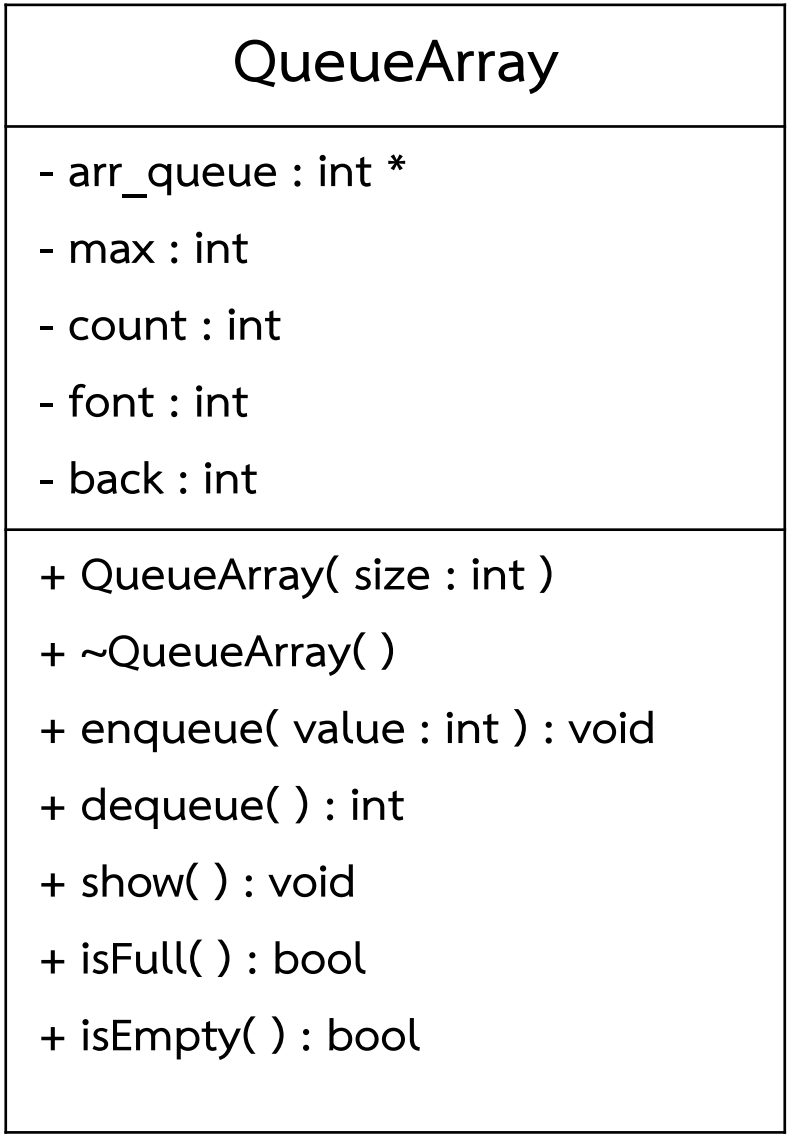
8.5 Queue with Pointer Implementation

7.2 Queue with Array Component

องค์ประกอบของการสร้างคิวด้วยอาร์เรย์ จะประกอบด้วย คุณสมบัติ (Property) และกระบวนการทำงาน (Method) เนื่องจากมีการสร้างขึ้นให้อยู่ในรูปแบบของคลาส (Class) ซึ่งมีรายละเอียดดังนี้

คุณสมบัติ (Property)	กระบวนการทำงาน (method)
arr_queue	enqueue
max	dequeue
count	show
font	isFull
back	isEmpty

Queue with Array Class



รายละเอียดคุณสมบัติของ Queue with Array

คุณสมบัติ (Property)	รายละเอียด
arr_queue	ตัวแปรอาร์เรย์สำหรับเก็บข้อมูลในรูปแบบคิว
max	ตัวแปรสำหรับเก็บจำนวนพื้นที่สูงสุดที่สามารถจัดเก็บข้อมูลได้
count	ตัวแปรสำหรับการใช้นับจำนวนข้อมูลที่เก็บไว้ทั้งหมด
font	ตัวแปรสำหรับเก็บตำแหน่งของข้อมูลตัวแรกในคิว
back	ตัวแปรสำหรับเก็บตำแหน่งของข้อมูลตัวสุดท้ายในคิว

รายละเอียดกระบวนการทำงานของ Queue with Array

กระบวนการทำงาน (method)	รายละเอียดการทำงาน
QueueArray(int size)	Constructor สำหรับสร้าง Array โดยระบุขนาดสูงสุด ตามค่าของพารามิเตอร์ที่ส่งเข้ามา
~QueueArray( )	Deconstructor สำหรับลบข้อมูลที่กำหนดขึ้นออกจากหน่วยความจำ
enqueue(int value)	เพิ่มข้อมูลโดยนำไปเก็บไว้ที่ช่องว่างลำดับหลังสุดของคิว
int dequeue( )	นำข้อมูลตัวที่เก็บตัวแรกของคิวออกไปใช้งาน

รายละเอียดกระบวนการทำงานของ Queue with Array [2]

กระบวนการทำงาน (method)	รายละเอียดการทำงาน
show( )	แสดงผลข้อมูลที่มีในคิวทั้งหมด ผ่านทางหน้าจอ
bool isFull( )	ตรวจสอบข้อมูลในคิว ว่าเต็มหรือไม่ ถ้าเต็มคืนค่า TRUE หากไม่จะคืนค่า FALSE
bool isEmpty( )	ตรวจสอบข้อมูลในคิว โดยถ้าไม่มีข้อมูลจะคืนค่า TRUE หากมีข้อมูลจะคืนค่า FALSE

บทที่ 8 Queue

บทเรียนย่อย

8.1 Queue Operations and Concept

8.2 Queue with Array Component

8.3 Queue with Array Implementation

8.4 Queue with Pointer Component

8.5 Queue with Pointer Implementation

7.3 Queue with Array Implementation

การสร้างคลาส Queue ด้วยภาษา C++

```
class QueueArray {
    private :
        int * arr_queue;
        int max;
        int count;
        int font;
        int back;
     public :
        QueueArray( int size );
        ~QueueArray( );
                ...
};
```

7.3 Queue with Array Implementation [2]

การสร้างคลาส Queue ด้วยภาษา C++

```
class QueueArray {
        ...
    enqueue( int  value );
    int dequeue( );
    show( );
    bool isFull( );
    bool isEmpty( );
};
```

การดำเนินการใน Constructor และ Deconstructor

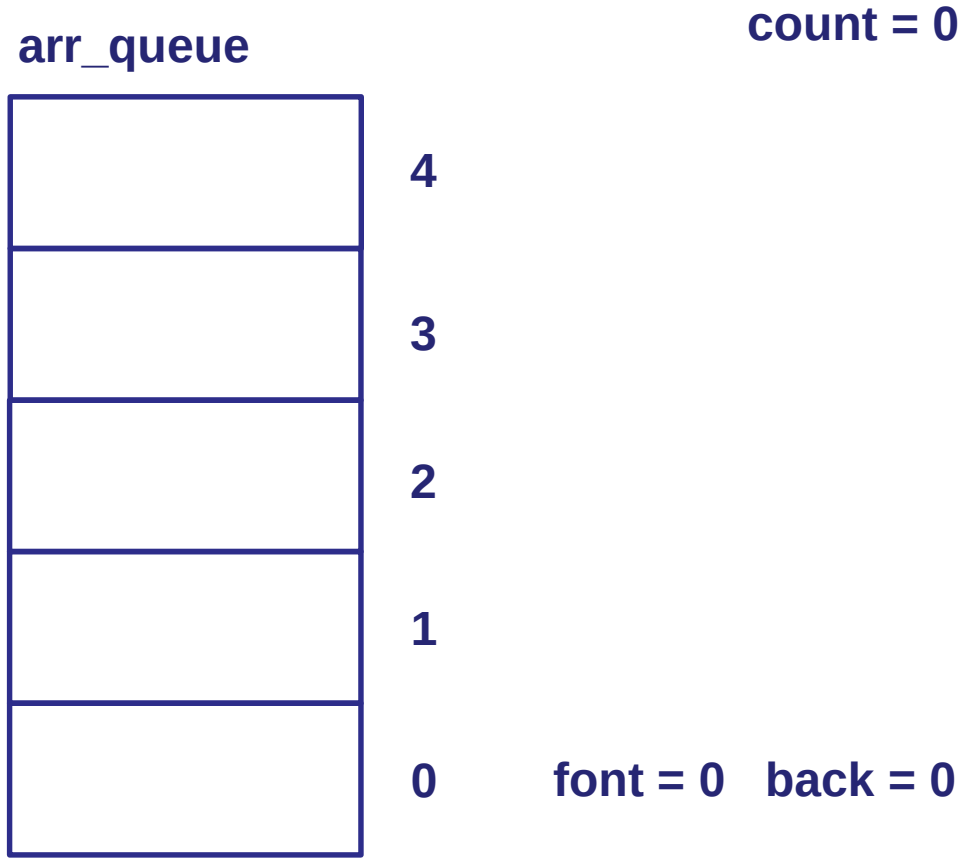
```
QueueArray :: QueueArray( int size ){
        arr_queue = new int[ size ];
        max = size;
        count = 0;
        font = 0;
        back = 0;
}

QueueArray :: ~ QueueArray( ){
      delete [] arr_queue;
}
```

การเรียกใช้งานคลาส QueueArray

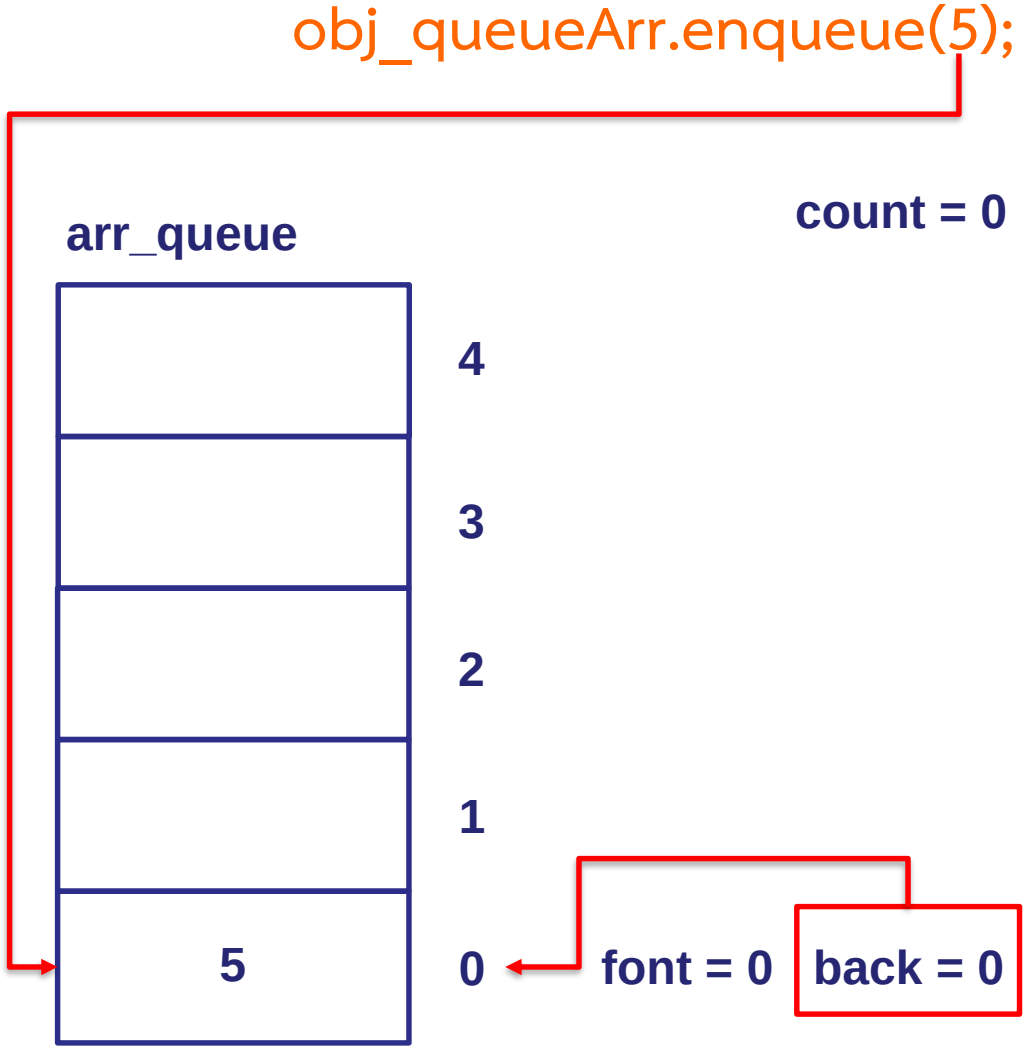
```
int main(void){
    QueueArray * obj_queueArr = new QueueArray(5);
    obj_queueArr->enqueue(5);
    obj_queueArr->enqueue(10);
    obj_queueArr->show();

    QueueArray obj_queueArr(5);
    obj_queueArr.enqueue(5);
    obj_queueArr.enqueue(10);
    obj_queueArr.show();
}
```

แนวคิดการกำหนดค่าเริ่มต้นของคิว

QueueArray obj_queueArr(5);



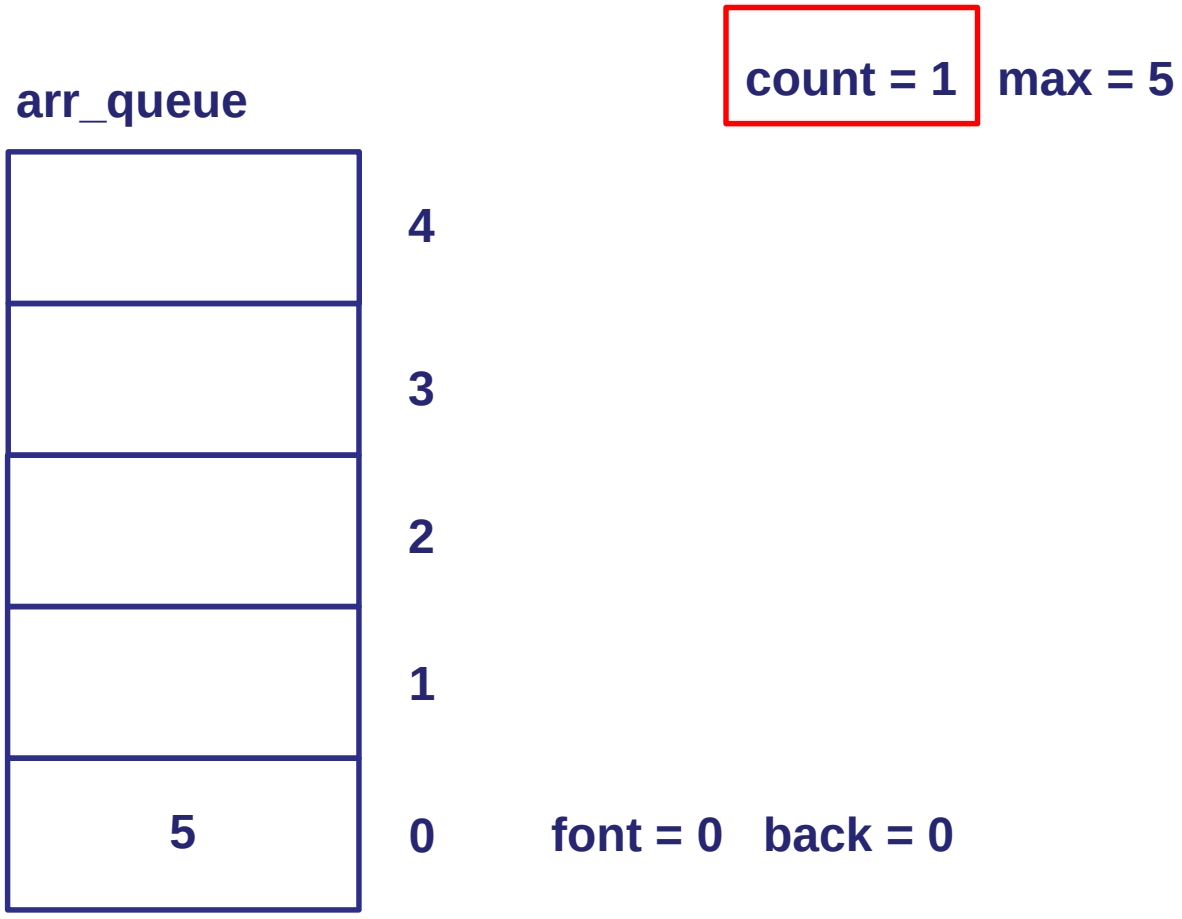
แนวคิดการนำข้อมูลเข้าคิว



* กรณีที่ในคิวยังไม่มีข้อมูล

แนวคิดการนำข้อมูลเข้าคิว [2]

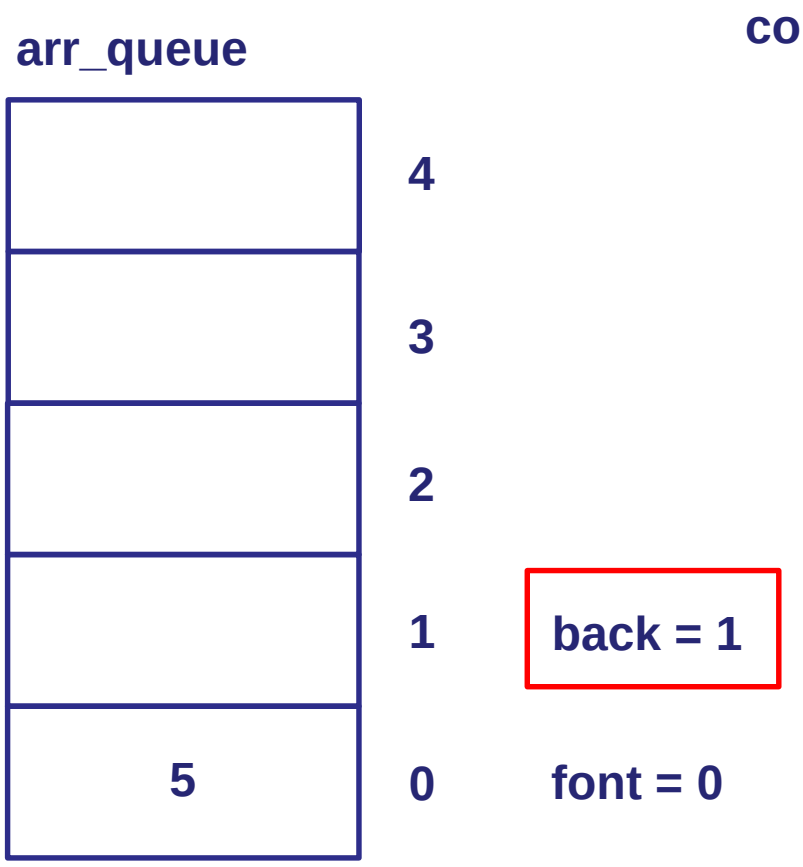
obj_queueArr.enqueue(5);



* กรณีที่ในคิวยังไม่มีข้อมูล

แนวคิดการนำข้อมูลเข้าคิว [3]

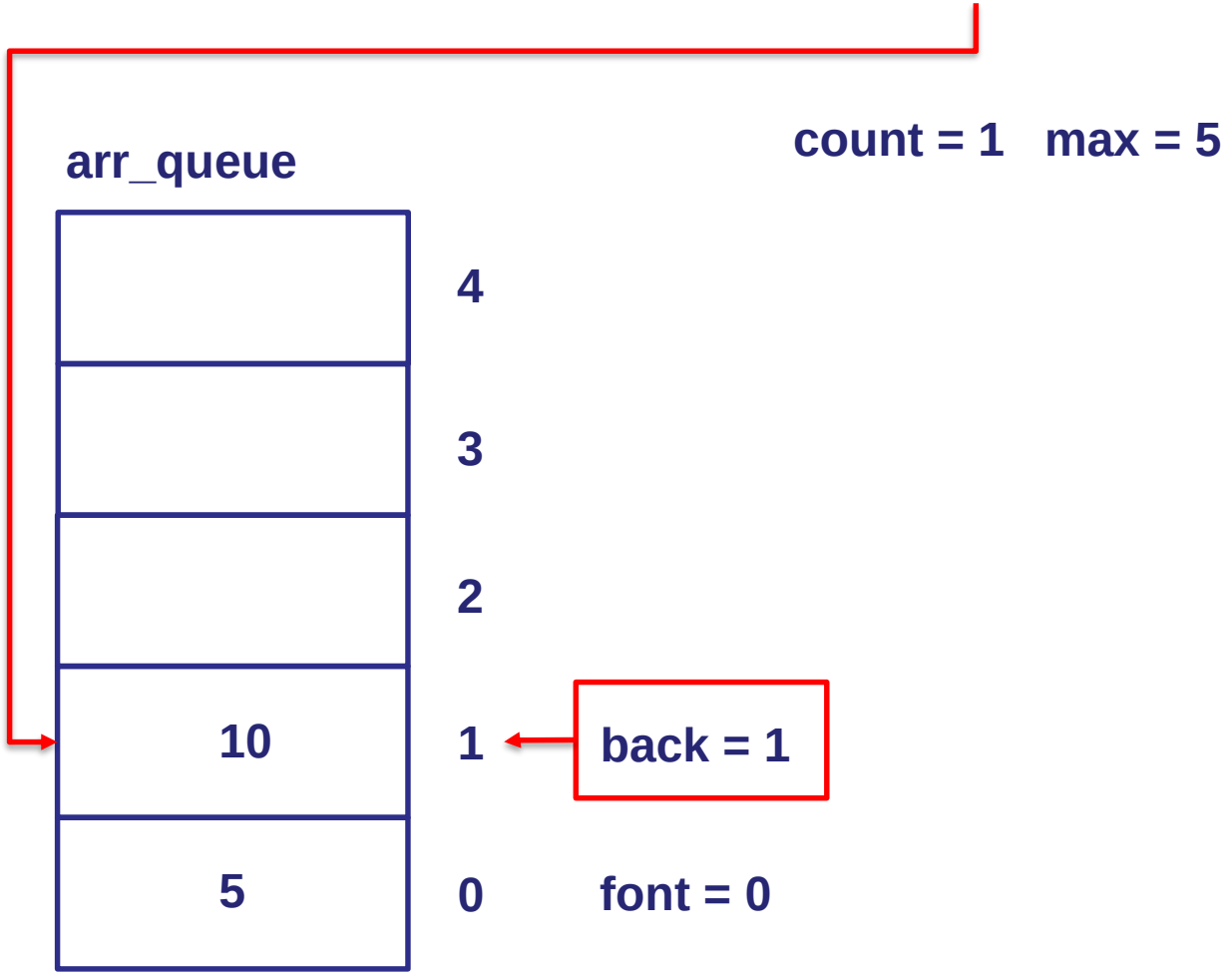
obj_queueArr.enqueue(10);



* กรณีที่ในคิวมีข้อมูล

แนวคิดการนำข้อมูลเข้าคิว [4]

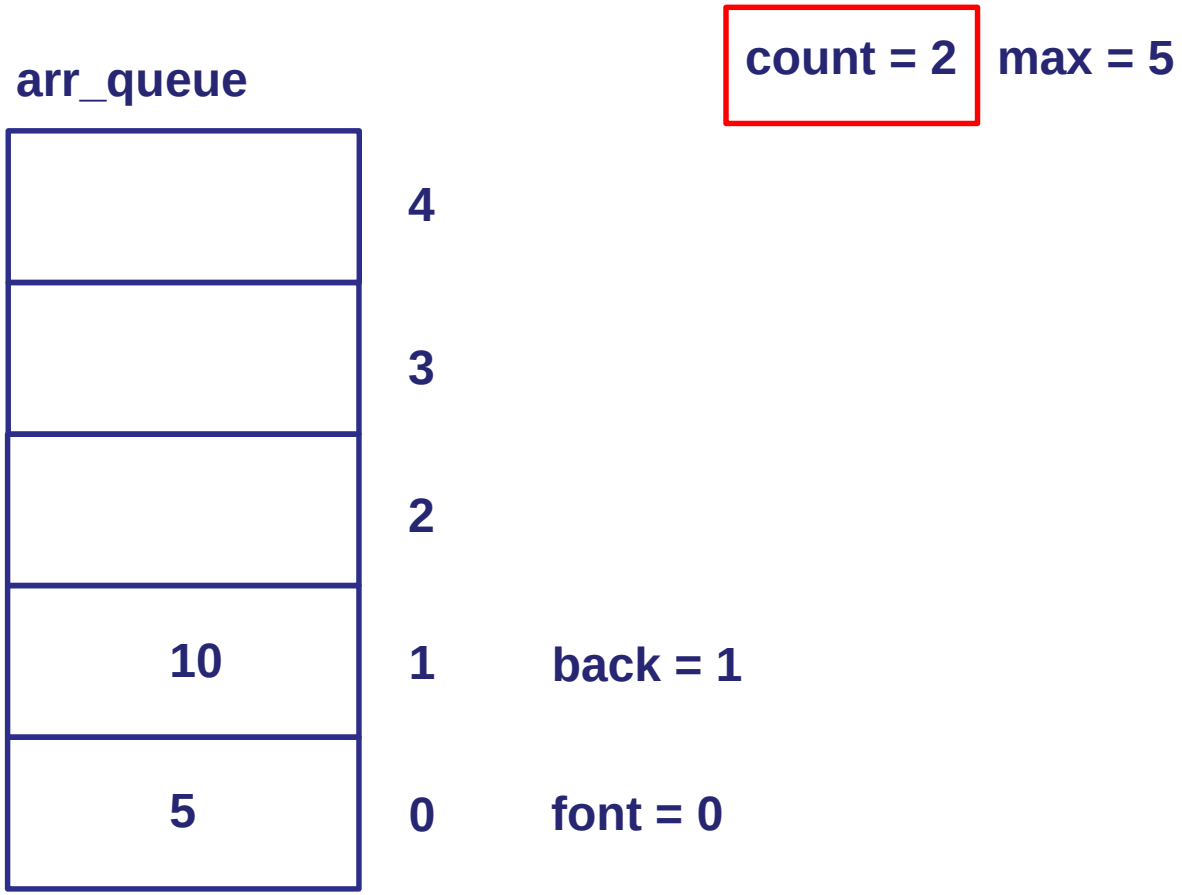
obj_queueArr.enqueue(10);



* กรณีที่ในคิวมีข้อมูล

แนวคิดการนำข้อมูลเข้าคิว [4]

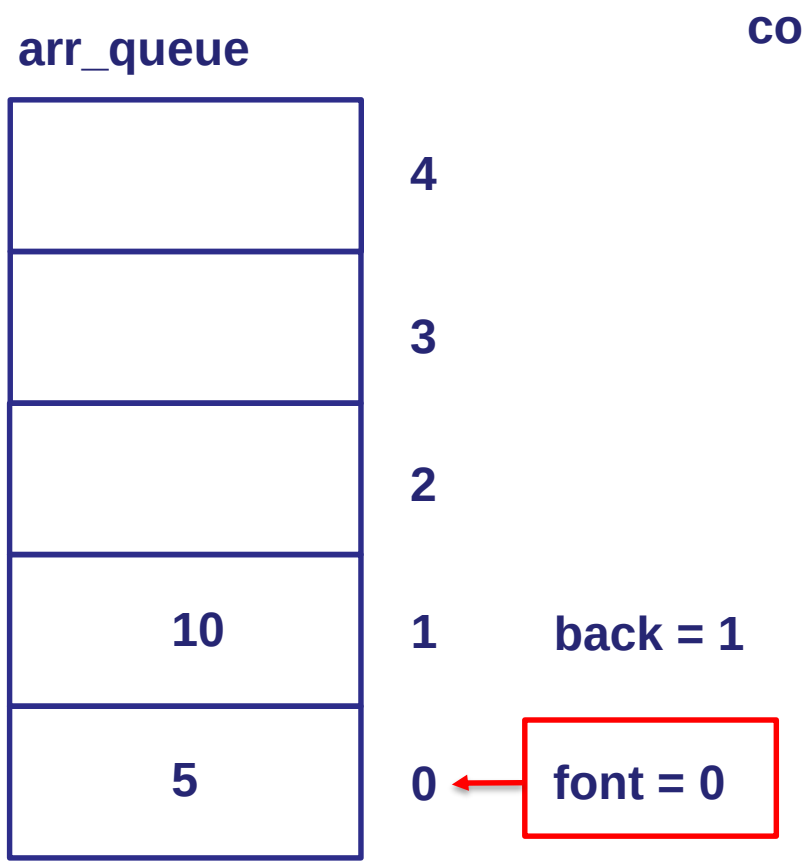
obj_queueArr.enqueue(10);



* กรณีที่ในคิวมีข้อมูล

แนวคิดการนำข้อมูลออกจากคิว

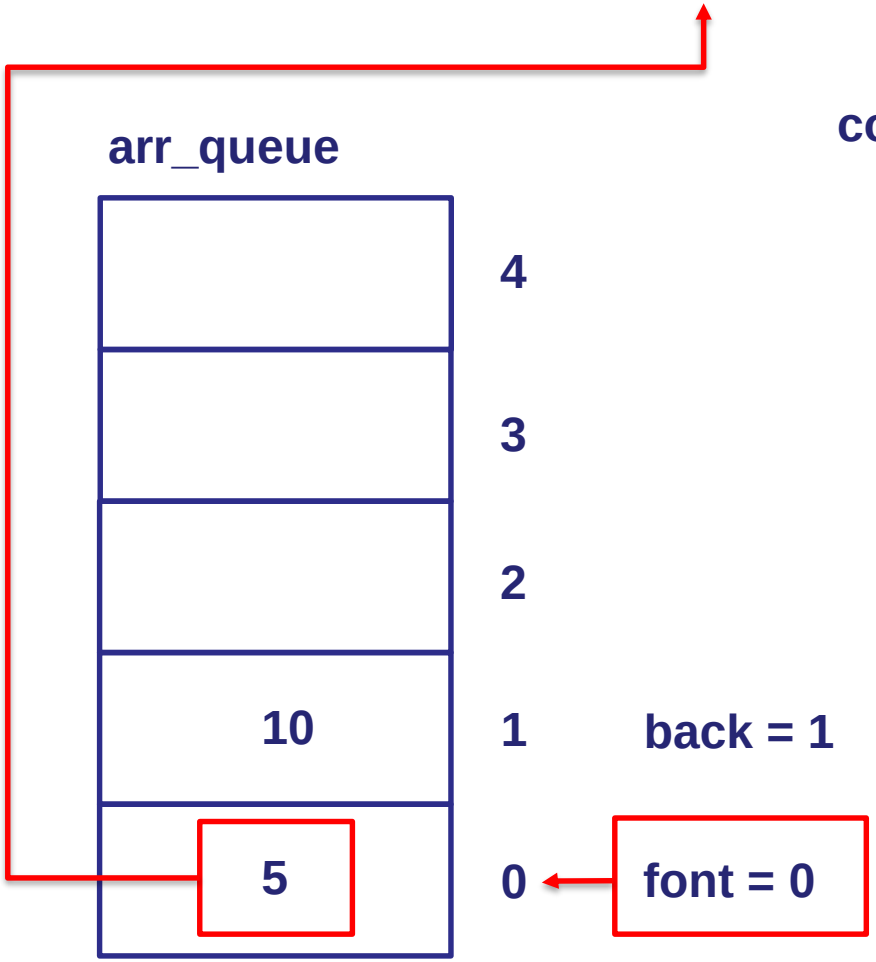
obj_queueArr.dequeue( );



* กรณีที่ในคิวมีข้อมูลมากกว่า 1 ตัว

แนวคิดการนำข้อมูลออกจากคิว

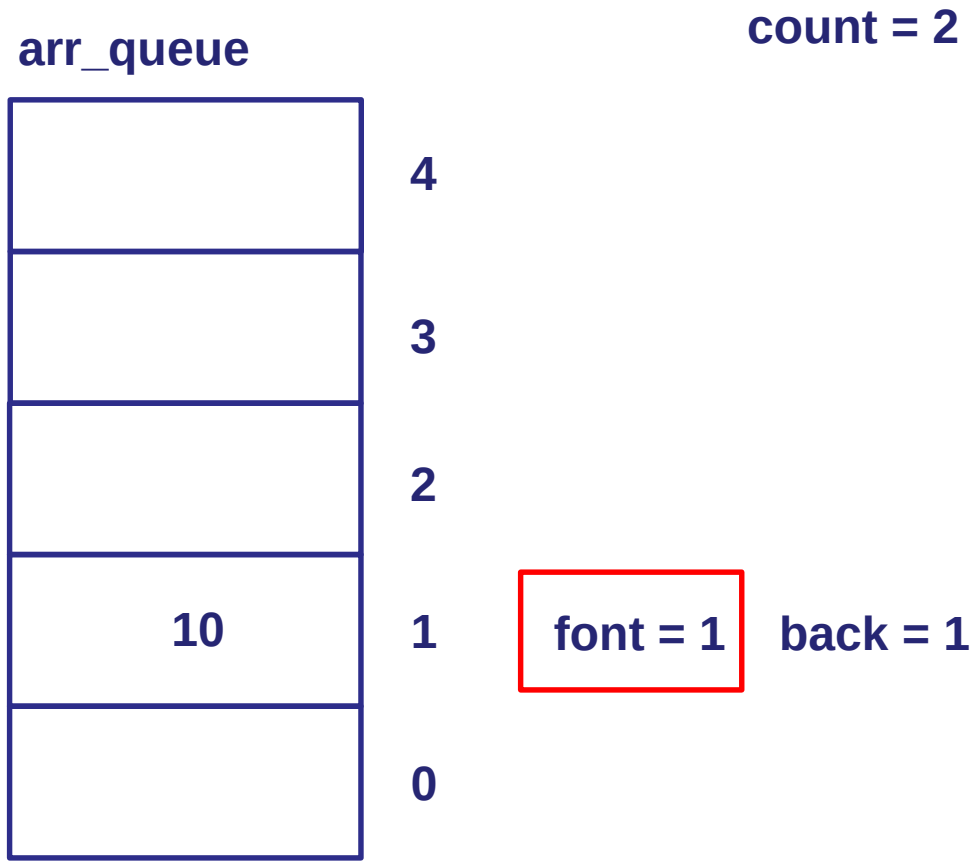
obj_queueArr.dequeue( );



* กรณีที่ในคิวมีข้อมูลมากกว่า 1 ตัว

แนวคิดการนำข้อมูลออกจากคิว [2]

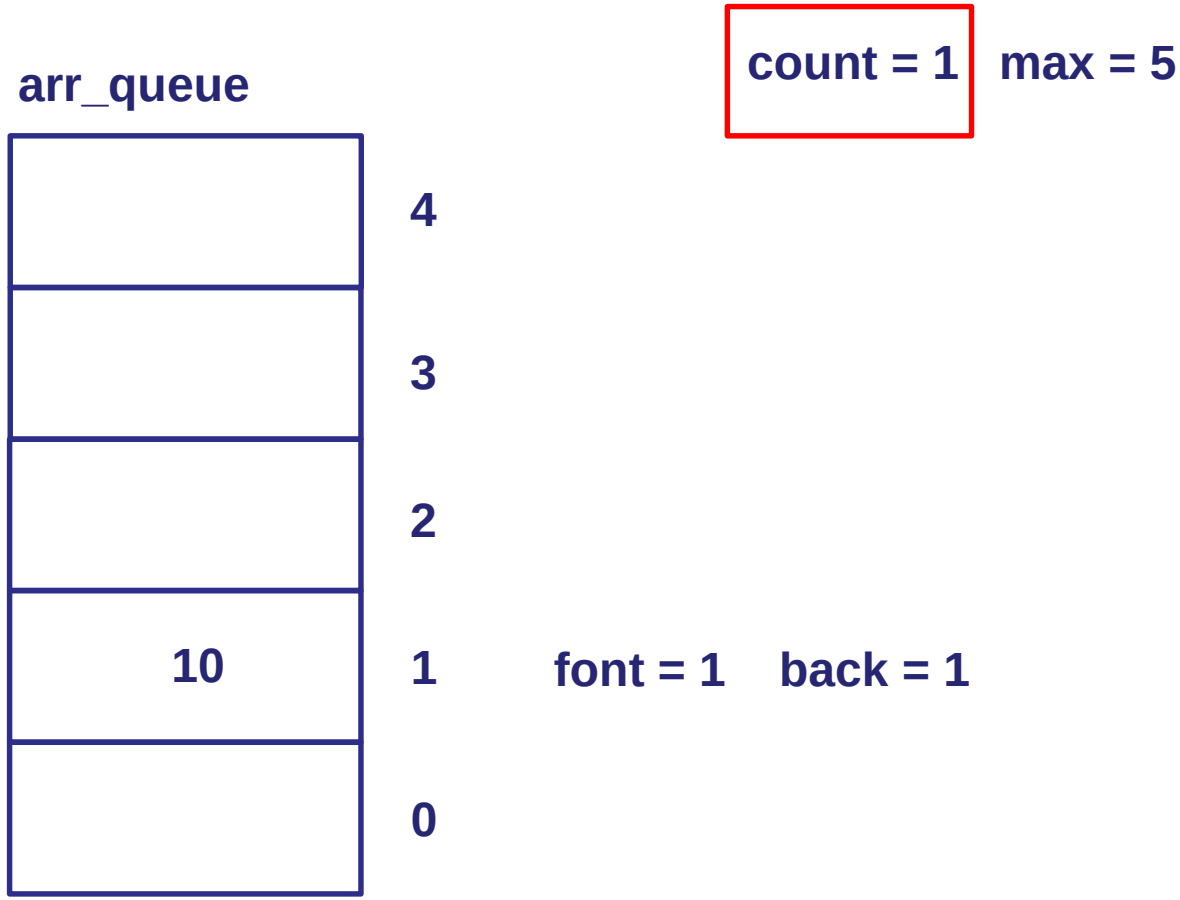
obj_queueArr.dequeue( );



* กรณีที่ในคิวมีข้อมูลมากกว่า 1 ตัว

แนวคิดการนำข้อมูลออกจากคิว [3]

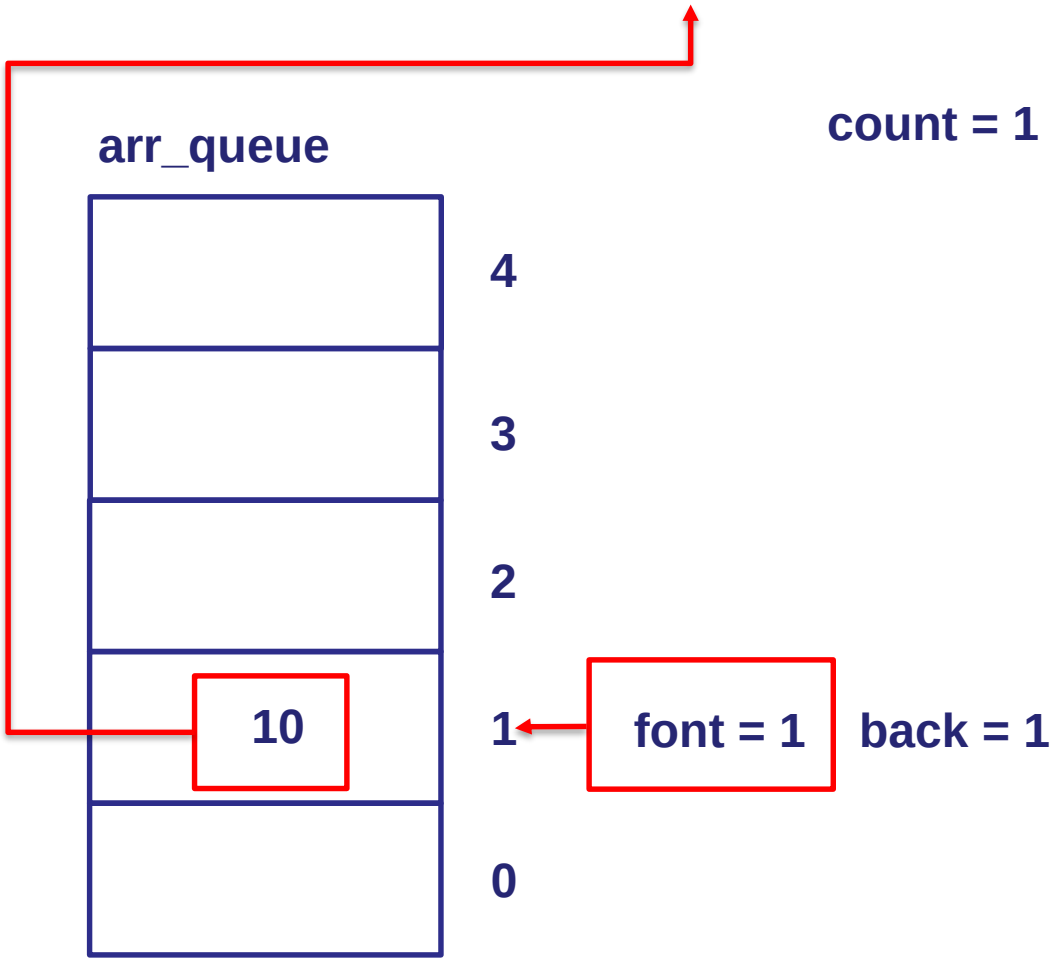
obj_queueArr.dequeue( );



* กรณีที่ในคิวมีข้อมูลมากกว่า 1 ตัว

แนวคิดการนำข้อมูลออกจากคิว [4]

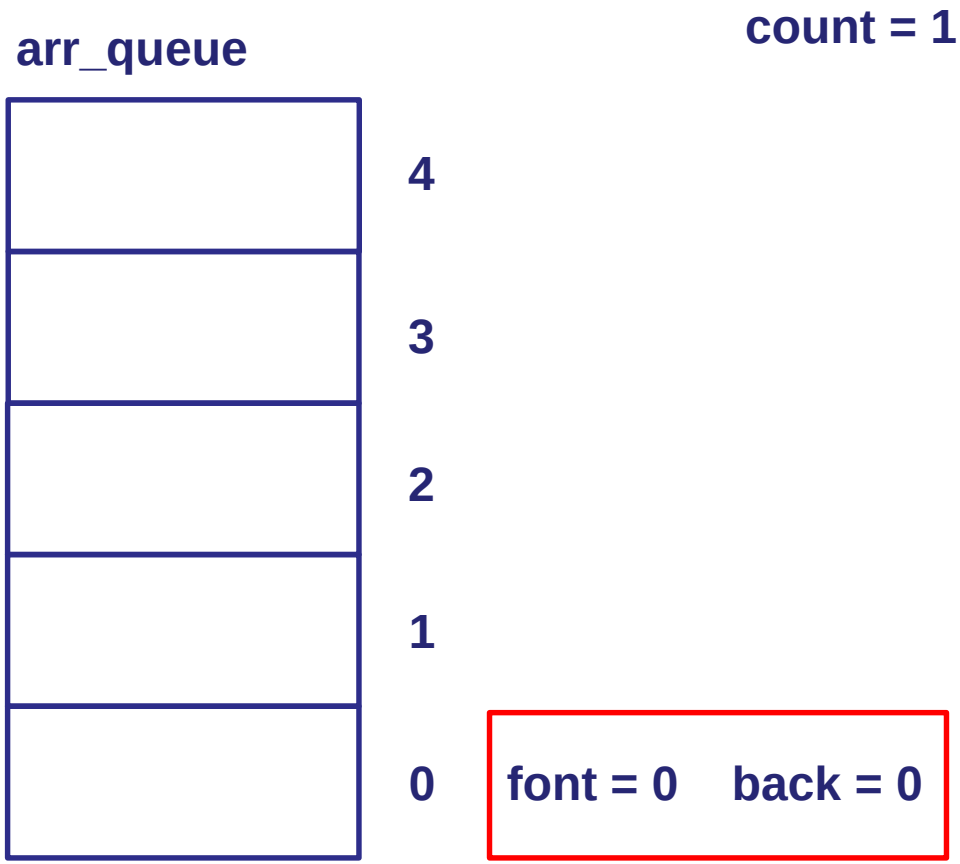
obj_queueArr.dequeue( );



* กรณีที่ในคิวยังมีข้อมูล 1 ตัวเท่านั้น

แนวคิดการนำข้อมูลออกจากคิว [5]

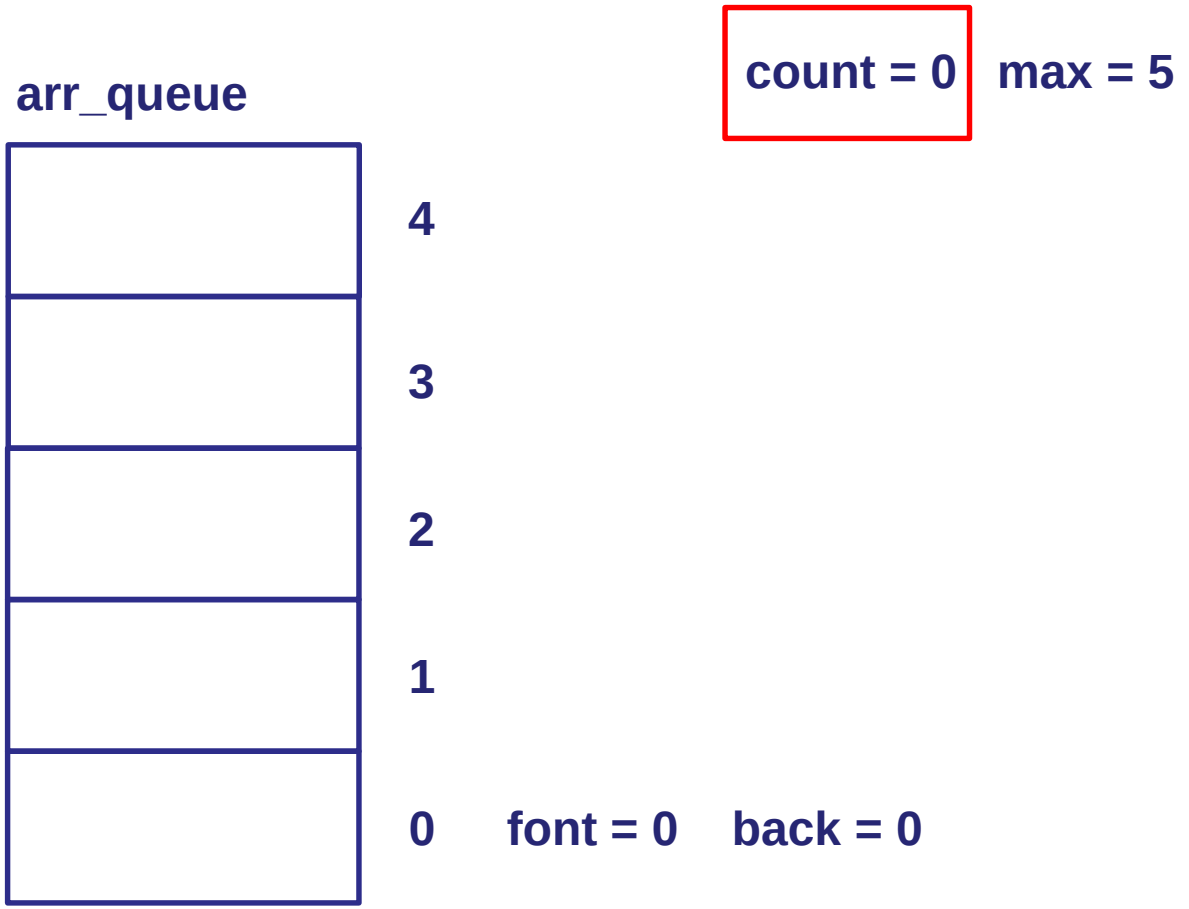
obj_queueArr.dequeue( );



* กรณีที่ในคิวยังมีข้อมูล 1 ตัวเท่านั้น

แนวคิดการนำข้อมูลออกจากคิว [6]

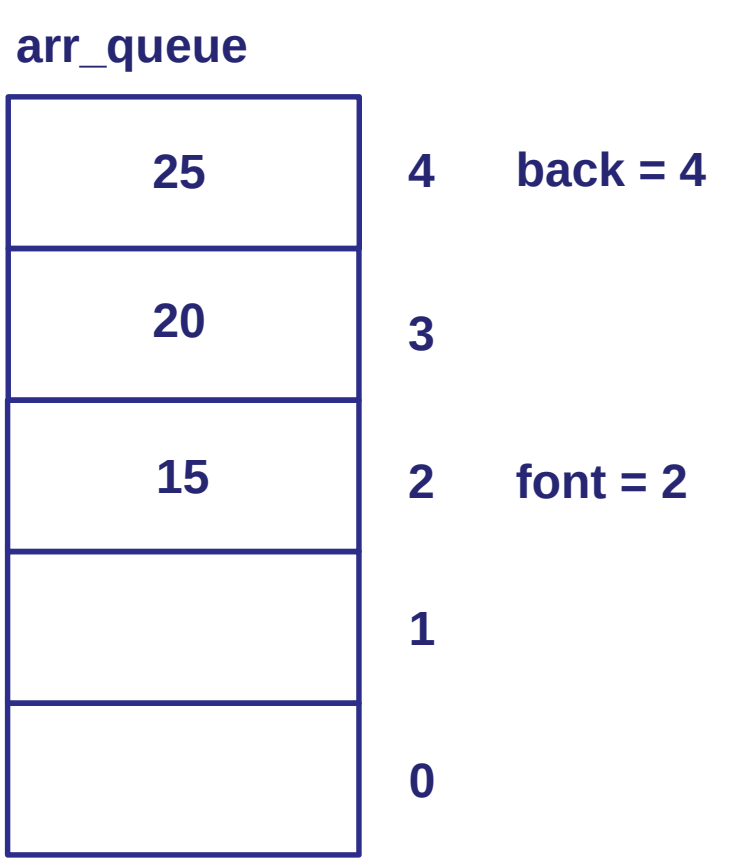
obj_queueArr.dequeue( );



* กรณีที่ในคิวยังมีข้อมูล 1 ตัวเท่านั้น

แนวคิดการจัดการข้อมูลแบบ Circular Queue (คิวแบบวงกลม)

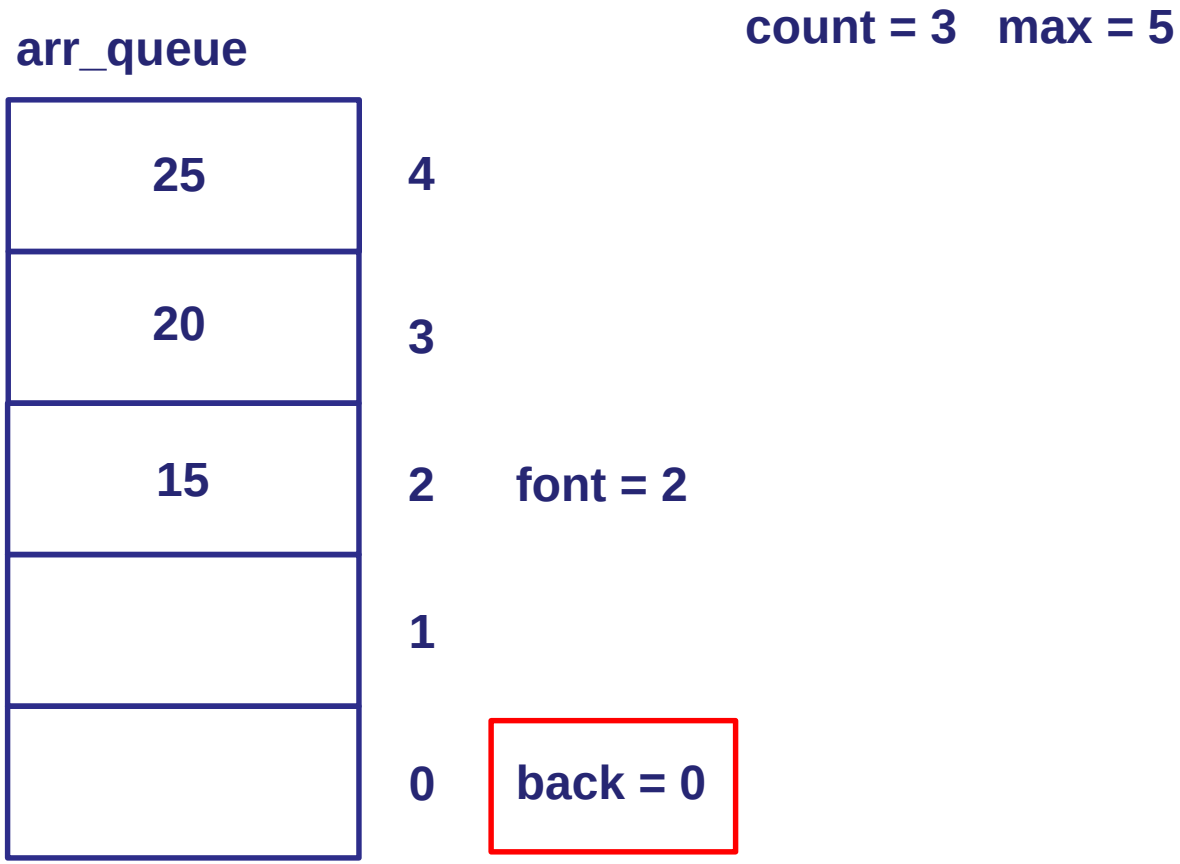
obj_queueArr.enqueue(30);



** สถานการณ์ตอนนำข้อมูลเข้าที่มีโอกาสเกิดขึ้นในการทำคิวแบบปกติ

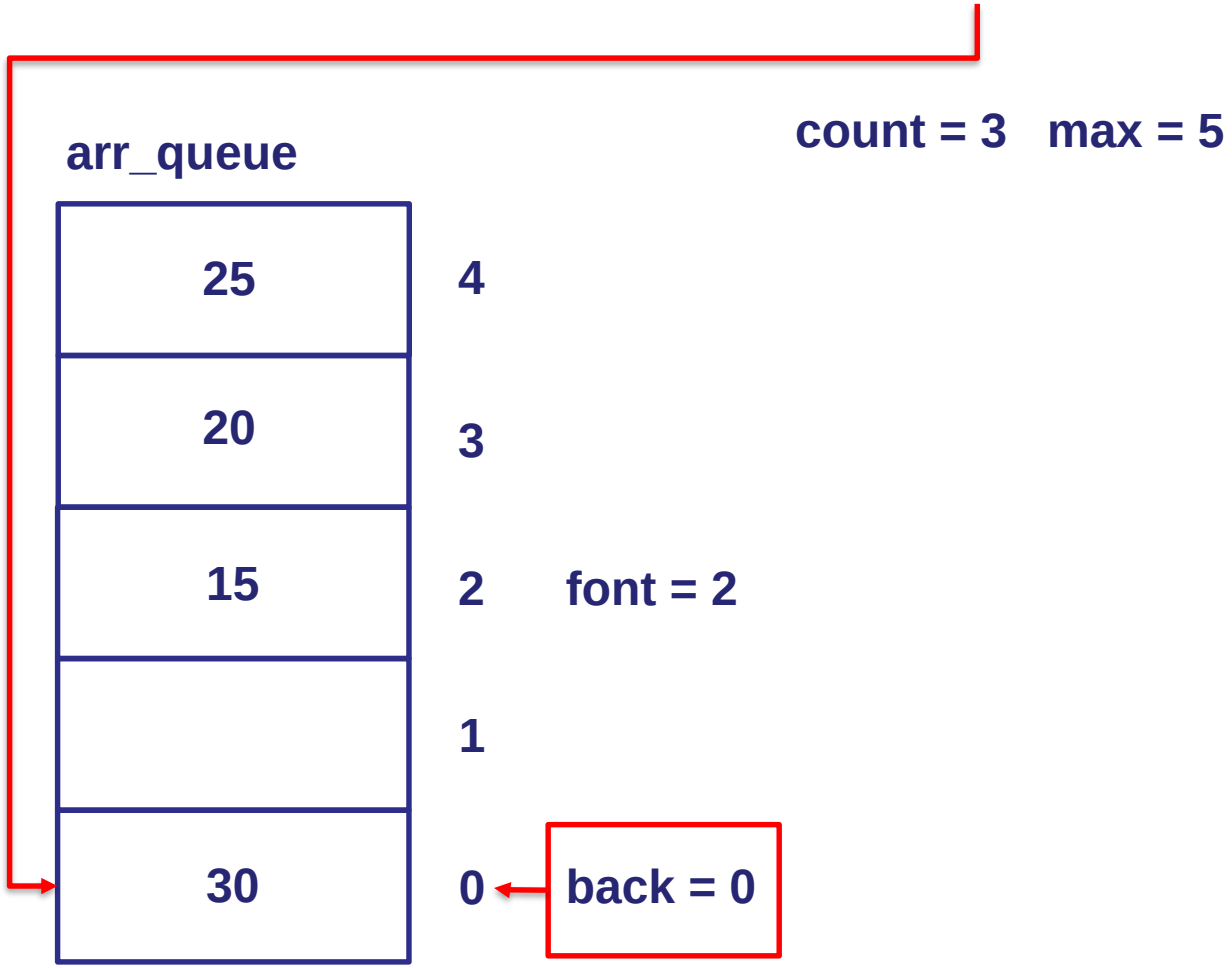
แนวคิดการจัดการข้อมูลแบบ Circular Queue [2]

obj_queueArr.enqueue(30);



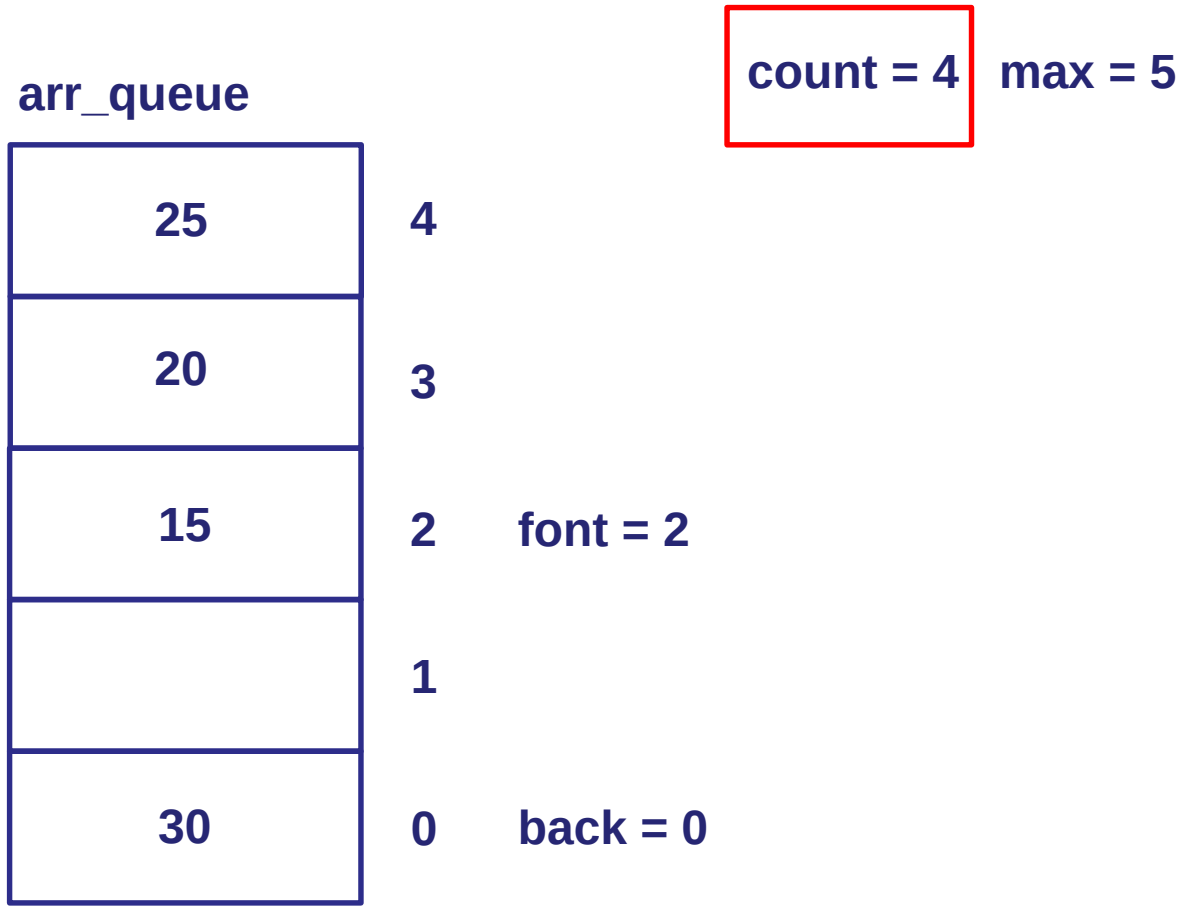
แนวคิดการจัดการข้อมูลแบบ Circular Queue [3]

obj_queueArr.enqueue(30);



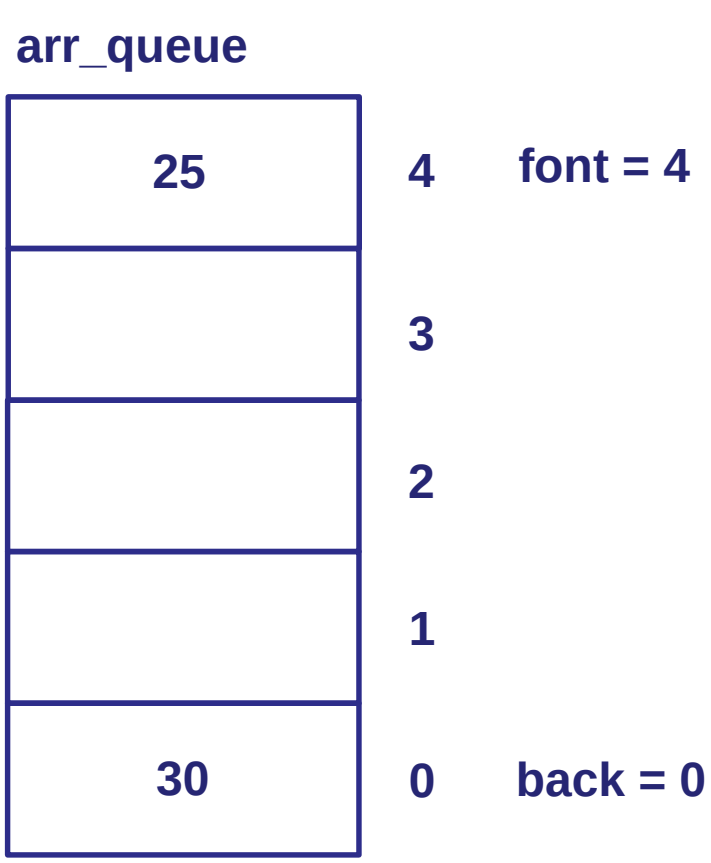
แนวคิดการจัดการข้อมูลแบบ Circular Queue [3]

obj_queueArr.enqueue(30);



แนวคิดการจัดการข้อมูลแบบ Circular Queue [4]

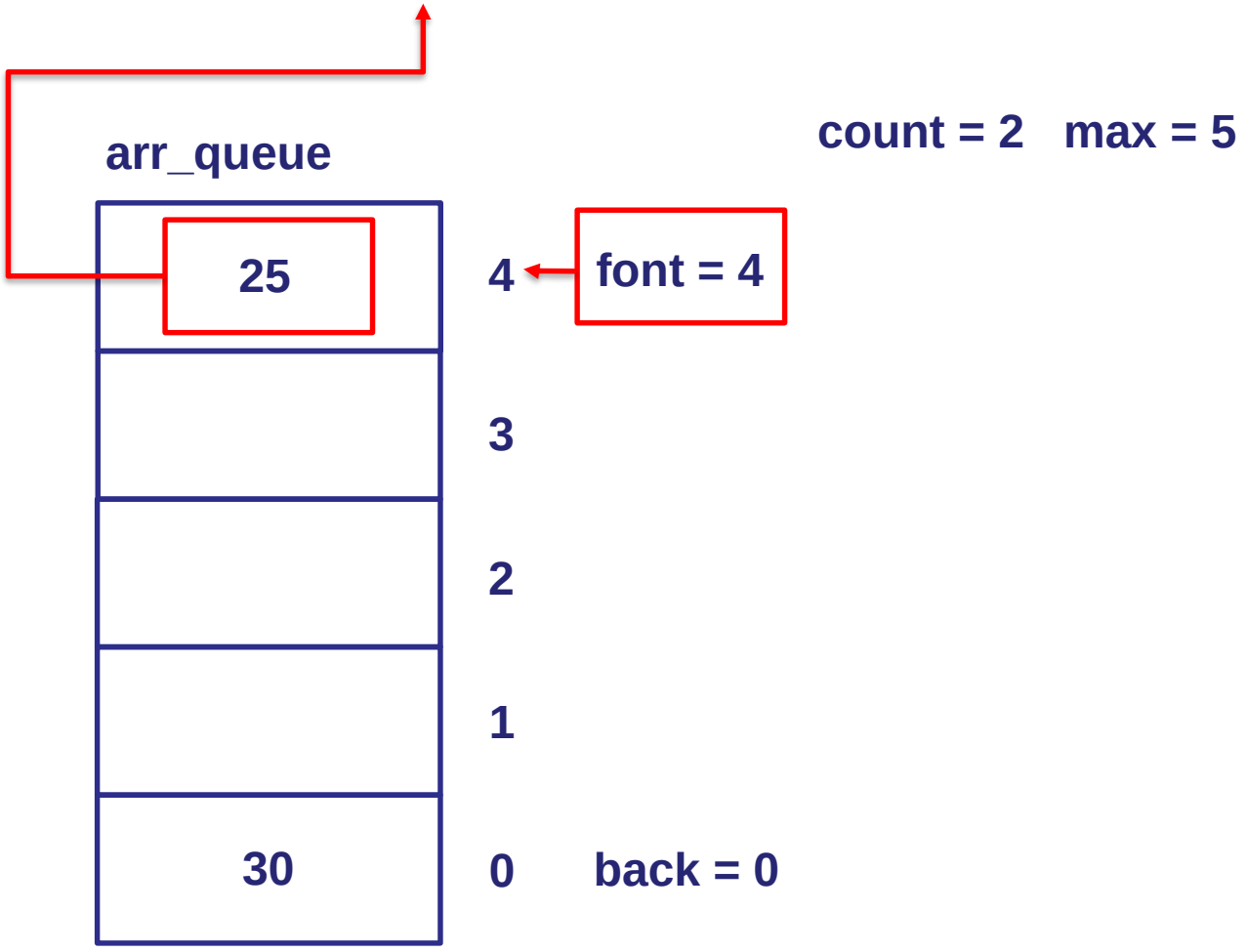
obj_queueArr.dequeue();



**สถานการณ์ตอนนำข้อมูลออกไปใช้งานที่มีโอกาสเกิดขึ้นในการทำคิวแบบปกติ

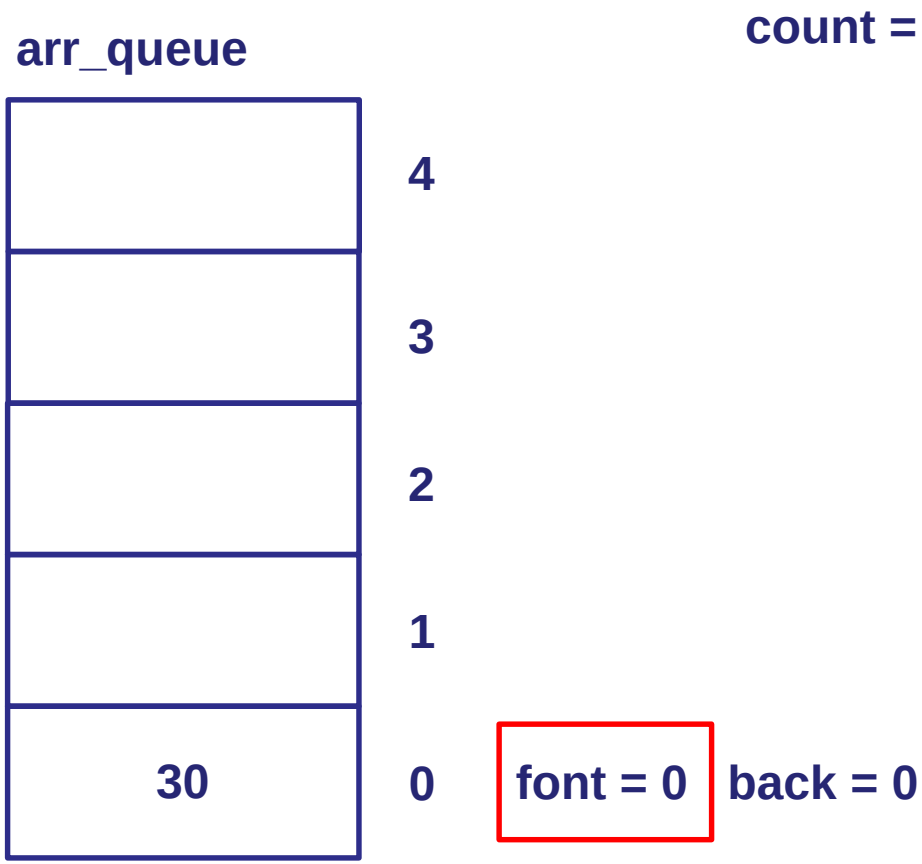
แนวคิดการจัดการข้อมูลแบบ Circular Queue [5]

obj_queueArr.dequeue();



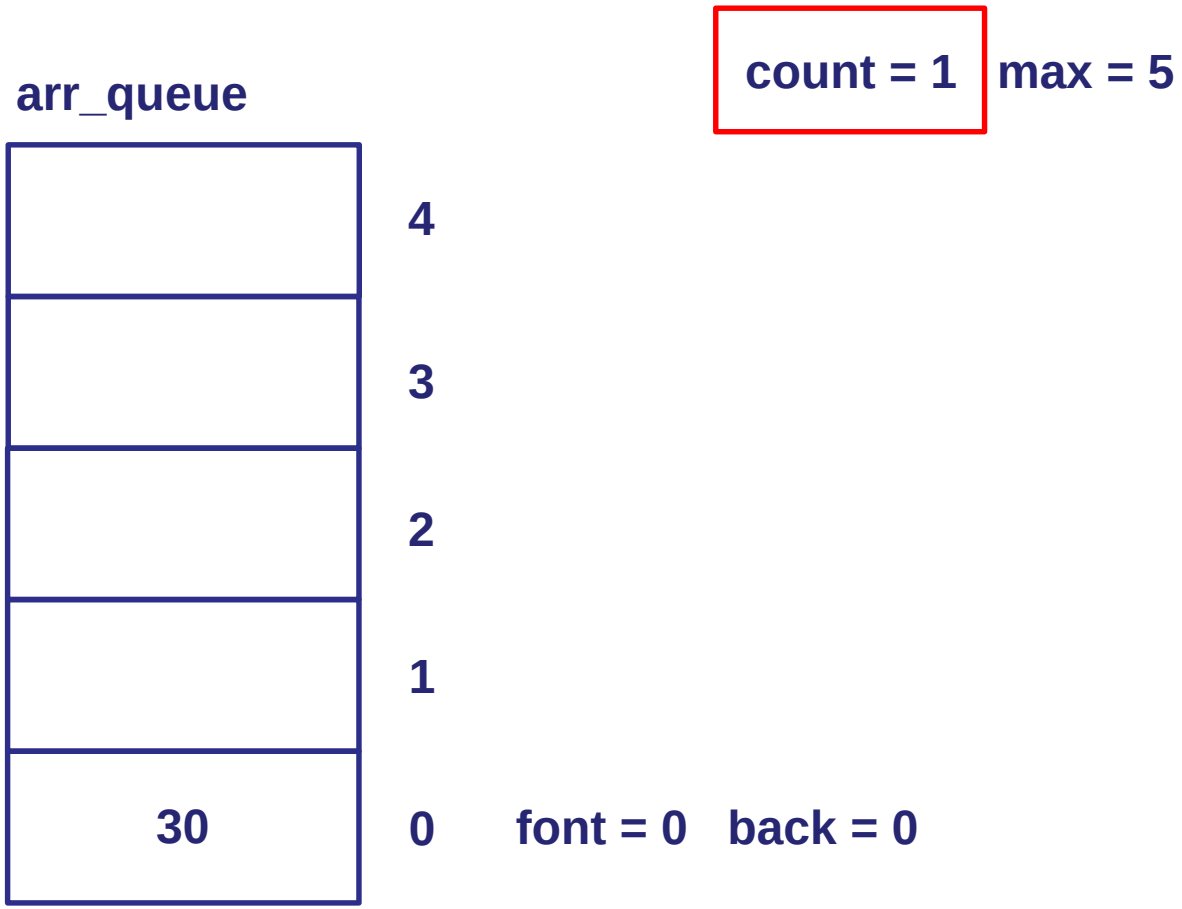
แนวคิดการจัดการข้อมูลแบบ Circular Queue [6]

obj_queueArr.dequeue();



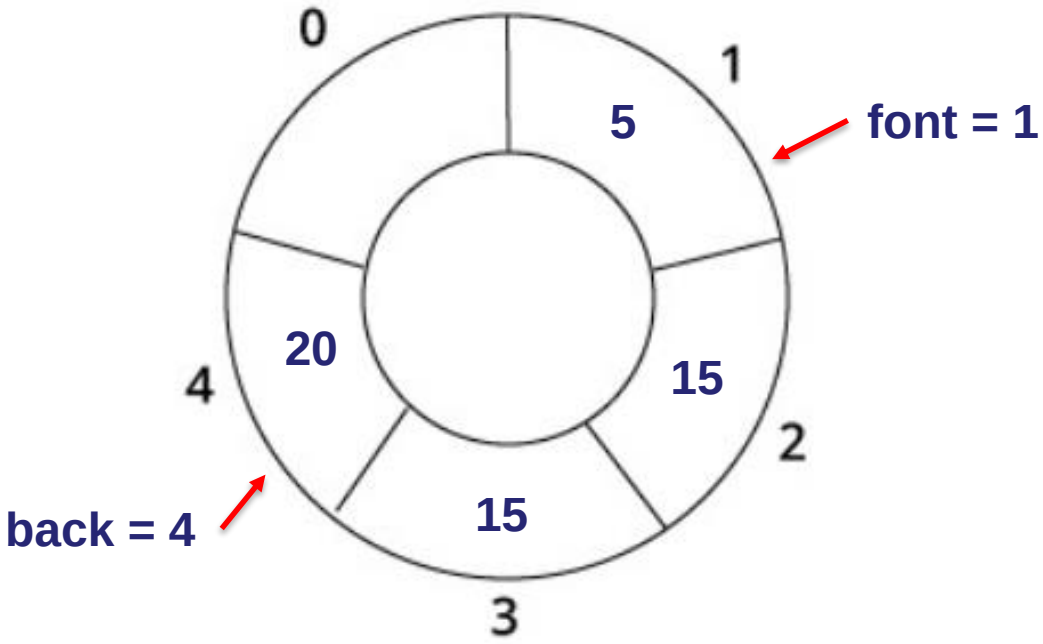
แนวคิดการจัดการข้อมูลแบบ Circular Queue [7]

obj_queueArr.dequeue();



ภาพจำลองลักษณะของ Circular Queue

count = 4 max = 5



บทที่ 8 Queue

บทเรียนย่อย

8.1 Queue Operations and Concept

8.2 Queue with Array Component

8.3 Queue with Array Implementation

8.4 Queue with Pointer Component

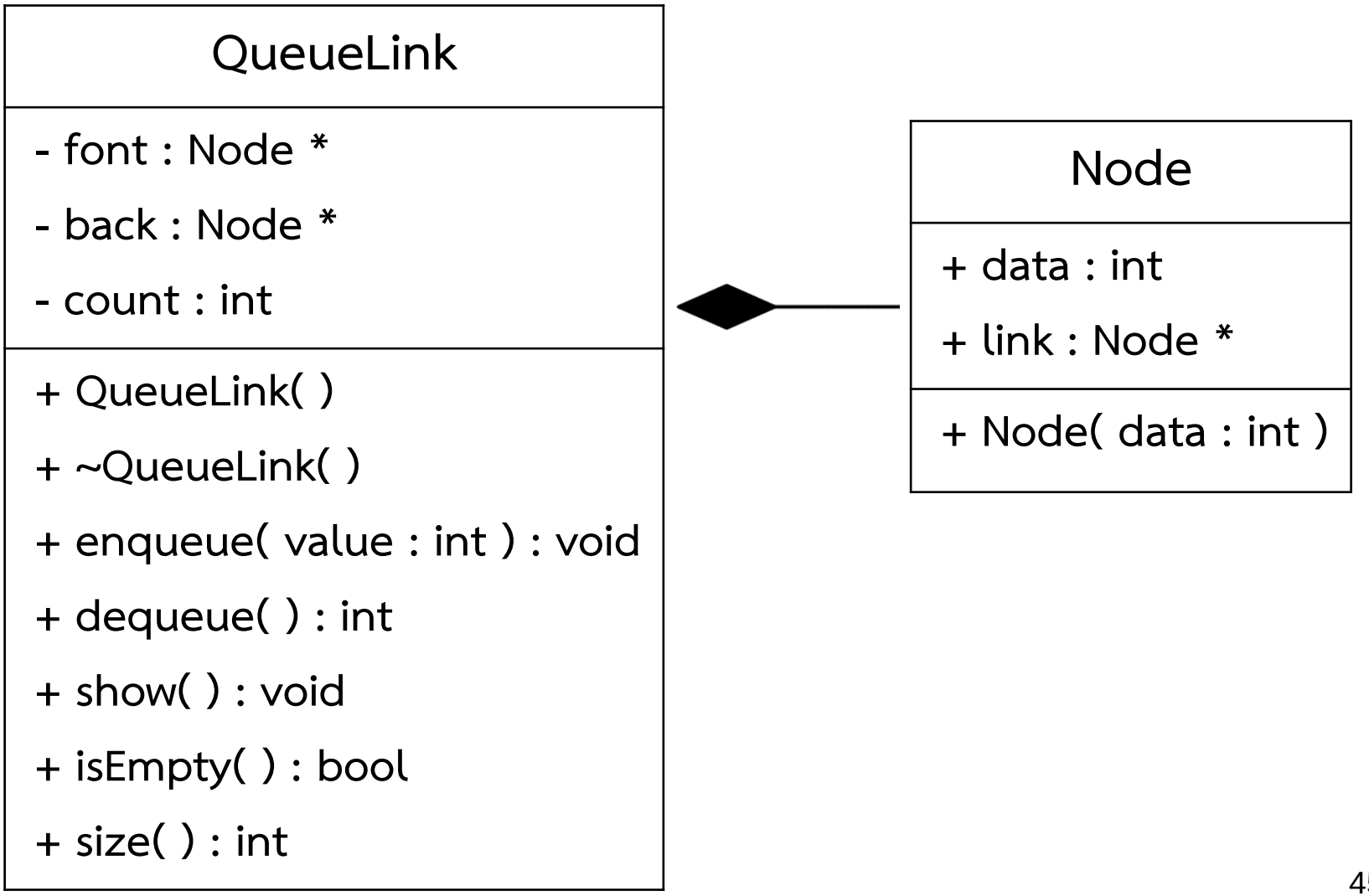
8.5 Queue with Pointer Implementation

7.4 Queue with Pointer Component

องค์ประกอบของการสร้างสแตกด้วยพอยเตอร์ จะประกอบด้วยคุณสมบัติ (Property) และกระบวนการทำงาน (Method) เนื่องจากมีการสร้างขึ้นให้อยู่ในรูปแบบของคลาส (Class) ซึ่งมีรายละเอียดดังนี้

คุณสมบัติ (Property)	กระบวนการทำงาน (method)
class node	enqueue
font	dequeue
back	show
count	isEmpty
	size

Queue with Pointer Class



รายละเอียดคุณสมบัติของ Queue with Pointer

คุณสมบัติ (Property)	รายละเอียด
Node	เป็น Class ที่เก็บข้อมูล และเก็บที่อยู่ของตัวถัดไป
font	ตัวแปรสำหรับเก็บที่อยู่ของข้อมูลตัวแรกของคิว
back	ตัวแปรสำหรับเก็บที่อยู่ของข้อมูลตัวสุดท้ายของคิว
count	ตัวแปรสำหรับการใช้นับจำนวนข้อมูลที่เก็บไว้ทั้งหมด

รายละเอียดกระบวนการทำงานของ Queue with Pointer

กระบวนการทำงาน (method)	รายละเอียดการทำงาน
QueueLink( )	Constructor สำหรับกำหนดค่าเริ่มต้น font = NULL , back = NULL และ count = 0
~QueueLink( )	Deconstructor สำหรับลบข้อมูลที่กำหนดขึ้นออกจากหน่วยความจำ
enqueue(int value)	เพิ่มข้อมูลโดยนำไปเก็บไว้ลำดับสุดท้ายของคิว
int dequeue( )	นำข้อมูลตัวแรกของคิวออกมาใช้งาน

รายละเอียดกระบวนการทำงานของ Queue with Pointer [2]

กระบวนการทำงาน (method)	รายละเอียดการทำงาน
show( )	แสดงผลข้อมูลที่มีในคิวทั้งหมด ผ่านทางหน้าจอ
bool isEmpty( )	ตรวจสอบข้อมูลในคิว โดยถ้าไม่มีข้อมูลจะคืนค่า TRUE หากมีข้อมูลจะคืนค่า FALSE
int size( )	คืนค่าจำนวนข้อมูลที่มีอยู่ในคิว

บทที่ 8 Queue

บทเรียนย่อย

8.1 Queue Operations and Concept

8.2 Queue with Array Component

8.3 Queue with Array Implementation

8.4 Queue with Pointer Component

8.5 Queue with Pointer Implementation

7.5 Queue with Pointer Implementation

การสร้างคลาส Queue ด้วยภาษา C++

```
class QueueLink {
    private :
        Node * font;
        Node * back;
        int count;
    public :
        QueueLink( );
        ~ QueueLink( );
            ...
};
```

7.5 Queue with Pointer Implementation [2]

การสร้างคลาส Queue ด้วยภาษา C++

```
class QueueArray {
        ...
    enqueue( int  value );
    int dequeue( );
    show( );
    bool isEmpty( );
    int size( );
};
```

การดำเนินการใน Constructor และ Deconstructor

```
QueueLink :: QueueLink( int size ){
        font = NULL;  back = NULL;
        count = 0;
}

QueueLink :: ~ QueueLink( ){
      for(Node * tmp = font; tmp != NULL; font = font->link){
            delete tmp;
             tmp = NULL;
       }
      font = NULL;    back = NULL;
}
```

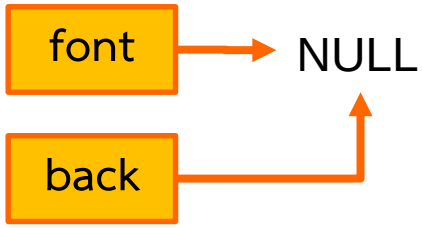
การเรียกใช้งานคลาส QueueLink

```
int main(void){
    QueueLink * obj_queueLink = new QueueLink( );
    obj_queueLink->enqueue(5);
    obj_queueLink- >enqueue(10);
    obj_queueLink- >show();

    QueueArray obj_queueLink;
    obj_queueLink.enqueue(5);
    obj_queueLink.enqueue(10);
    obj_queueLink.show();
}
```

แนวคิดการกำหนดค่าเริ่มต้นของคิว

QueueLink obj_queueLink;

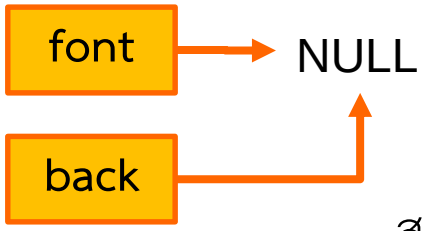
count = 0



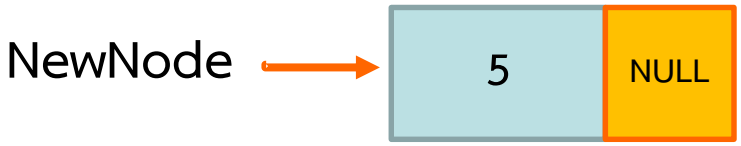
แนวคิดการนำข้อมูลเข้าคิว

obj_queueLink.enqueue(5);

count = 0



สร้าง new object จาก class node โดยกำหนดค่า data มีค่าเท่ากับ 5 และ link มีค่าเป็น NULL

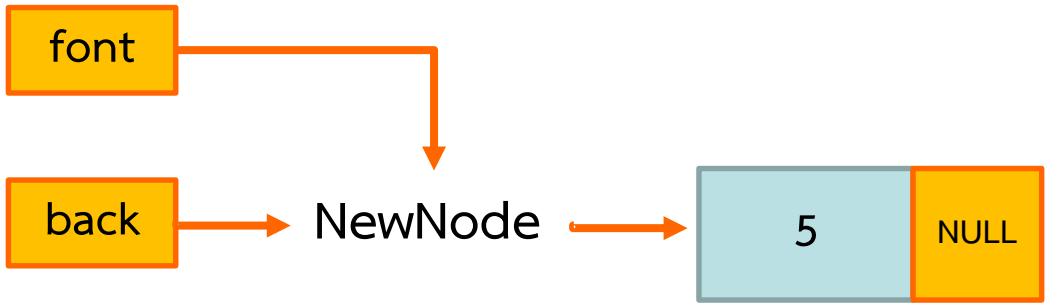


*** กรณีที่ในคิวไม่มีข้อมูล**

แนวคิดการนำข้อมูลเข้าคิว [2]

obj_queueLink.enqueue(5);

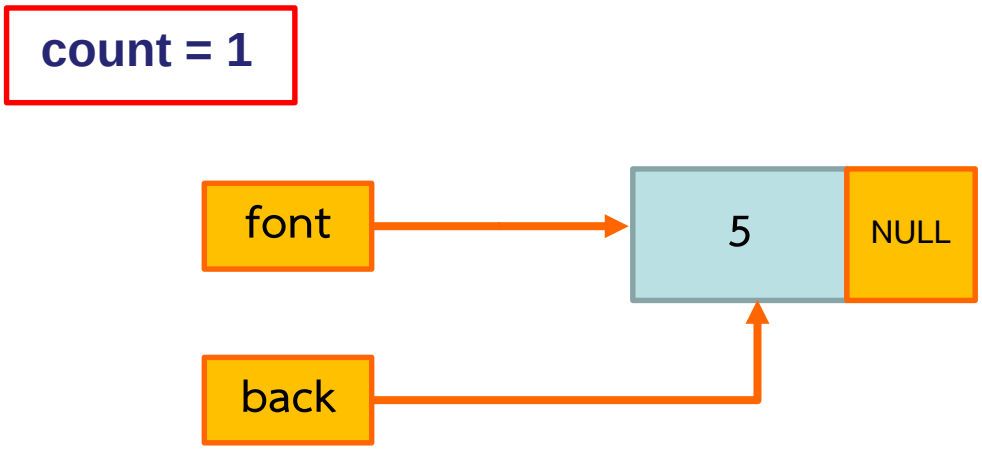
count = 0



*** กรณีที่ในคิวไม่มีข้อมูล**

แนวคิดการนำข้อมูลเข้าคิว [3]

obj_queueLink.enqueue(5);

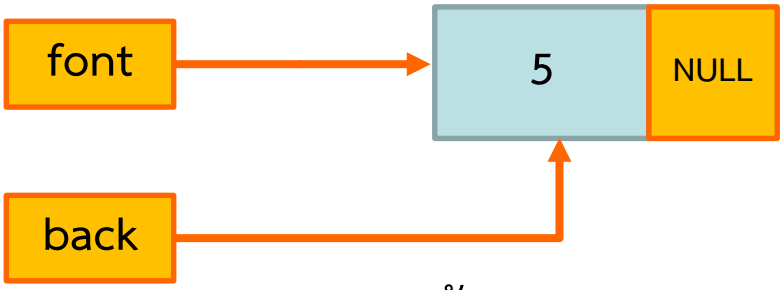


* กรณีที่ในคิวไม่มีข้อมูล

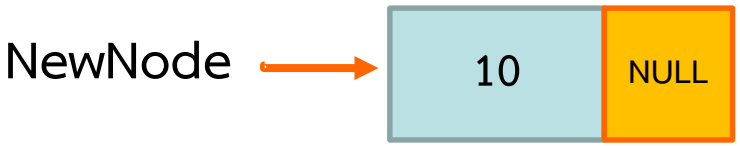
แนวคิดการนำข้อมูลเข้าคิว [4]

obj_queueLink.enqueue(10);

count = 1



สร้าง new object จาก class node โดยกำหนดค่า data มีค่าเท่ากับ 10 และ link มีค่าเป็น NULL

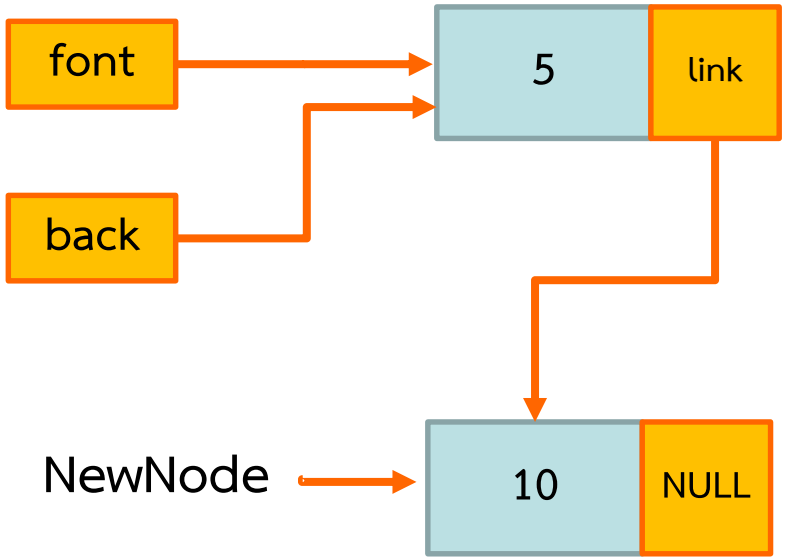


*** กรณีที่ในคิวมีข้อมูลอยู่**

แนวคิดการนำข้อมูลเข้าคิว [5]

obj_queueLink.enqueue(10);

count = 1

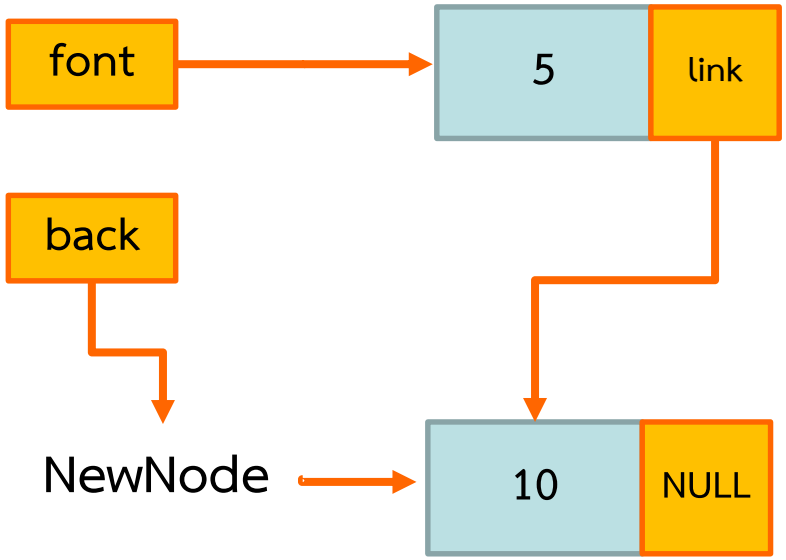


*** กรณีที่ในคิวมีข้อมูลอยู่**

แนวคิดการนำข้อมูลเข้าคิว [6]

obj_queueLink.enqueue(10);

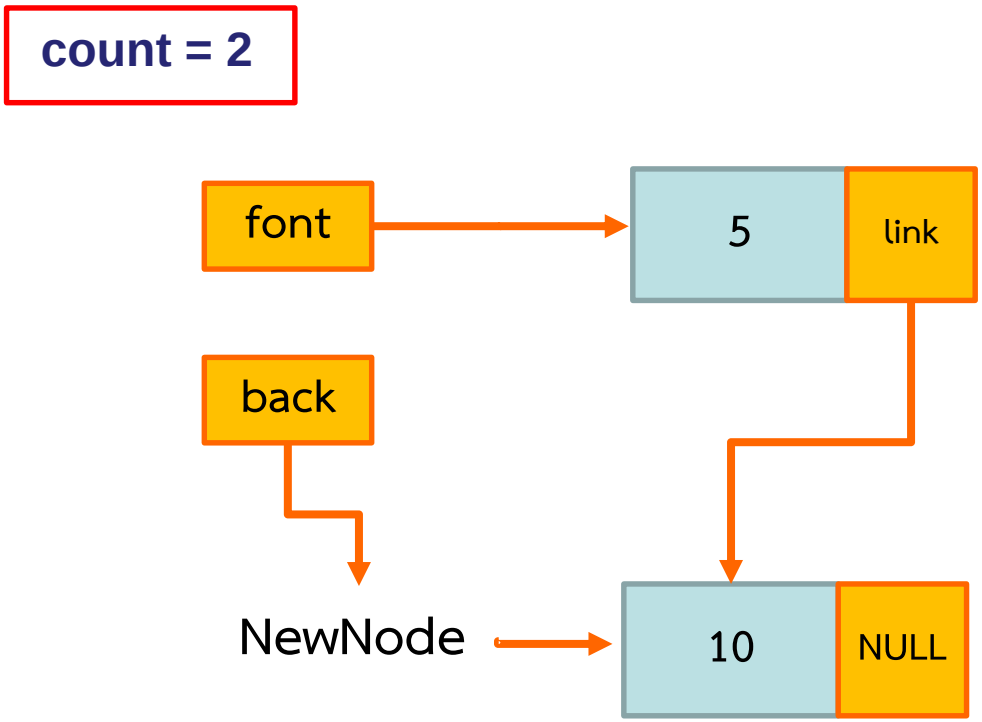
count = 1



*** กรณีที่ในคิวมีข้อมูลอยู่**

แนวคิดการนำข้อมูลเข้าคิว [7]

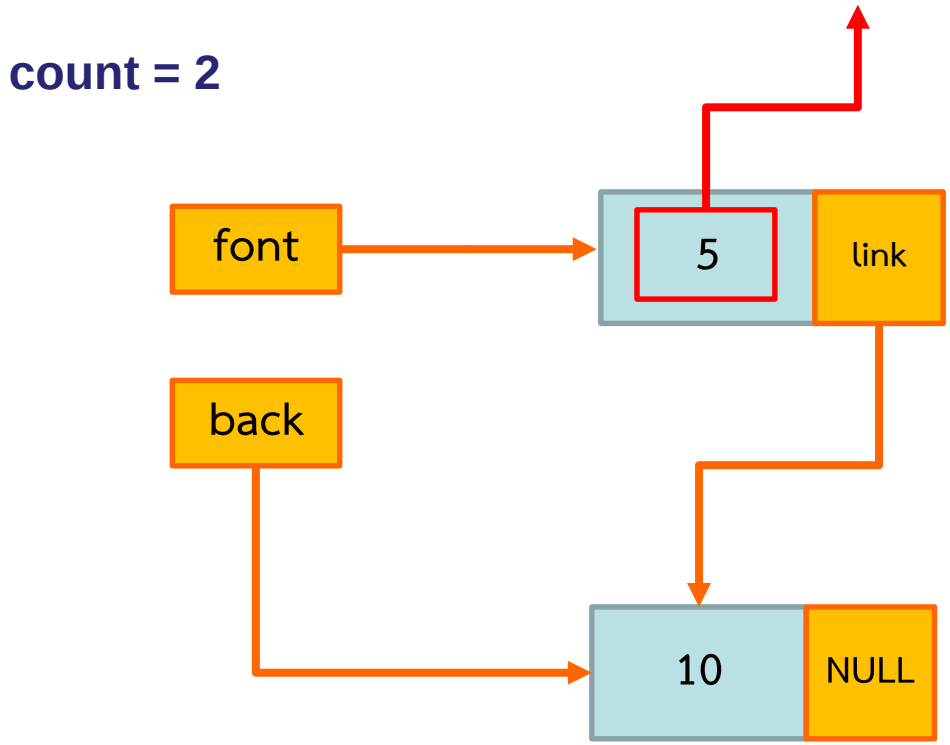
obj_queueLink.enqueue(10);



* กรณีที่ในคิวมีข้อมูลอยู่

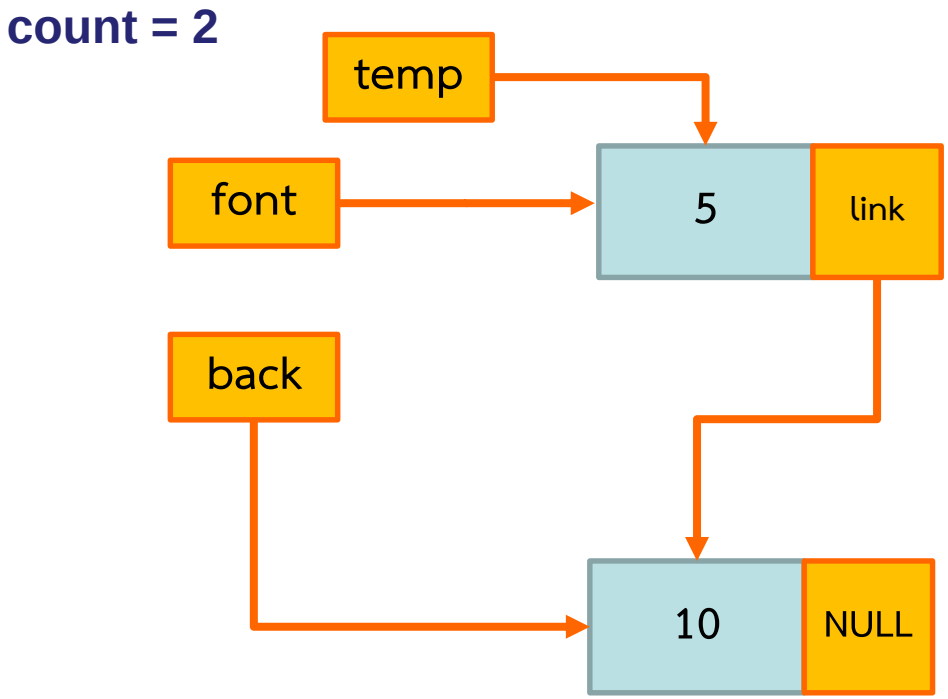
แนวคิดการนำข้อมูลออกจากคิว

obj_queueLink.dequeue( );



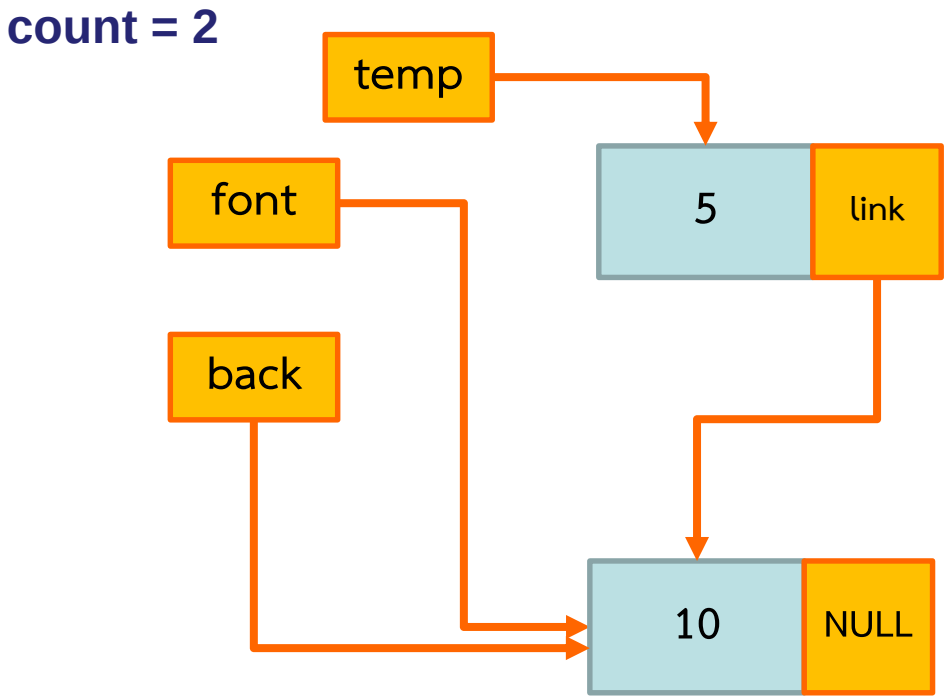
แนวคิดการนำข้อมูลออกจากคิว [2]

obj_queueLink.dequeue( );



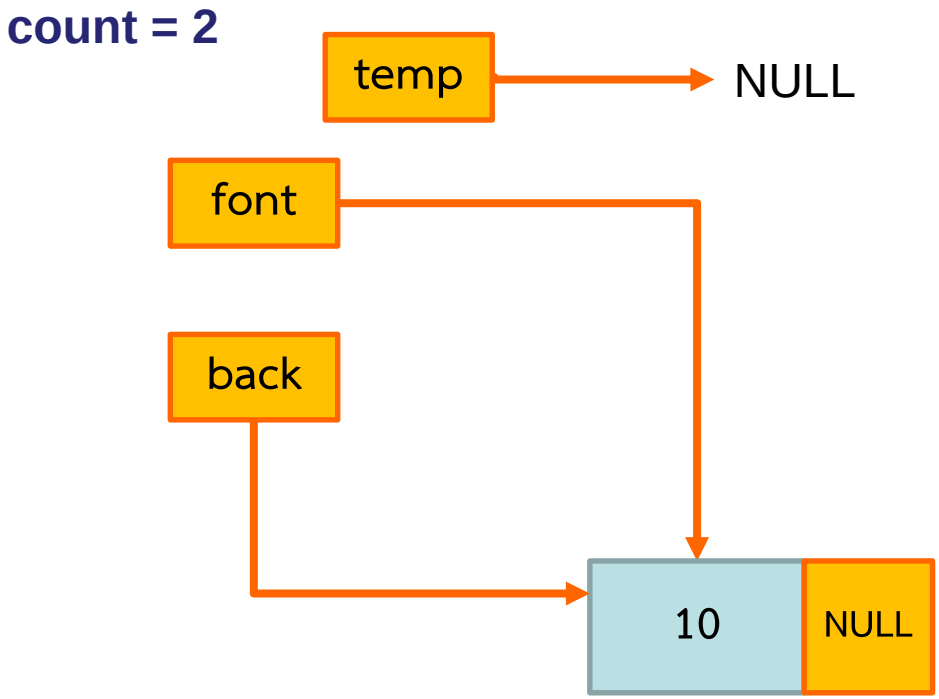
แนวคิดการนำข้อมูลออกจากคิว [3]

obj_queueLink.dequeue( );



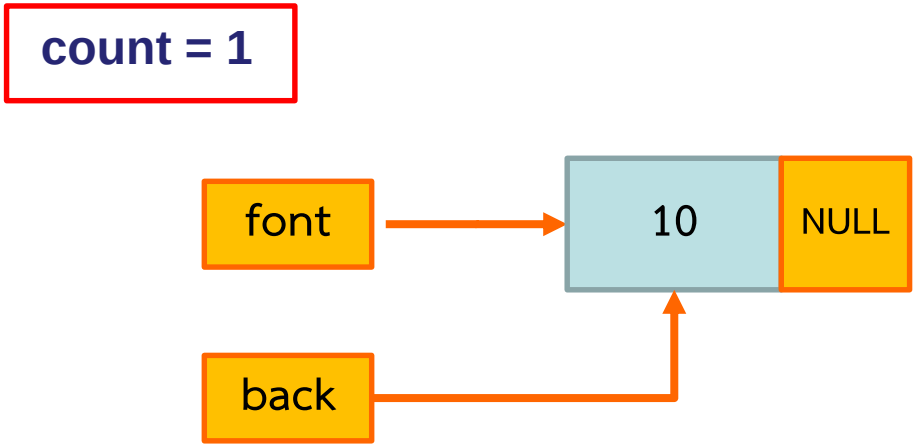
แนวคิดการนำข้อมูลออกจากคิว [4]

obj_queueLink.dequeue( );



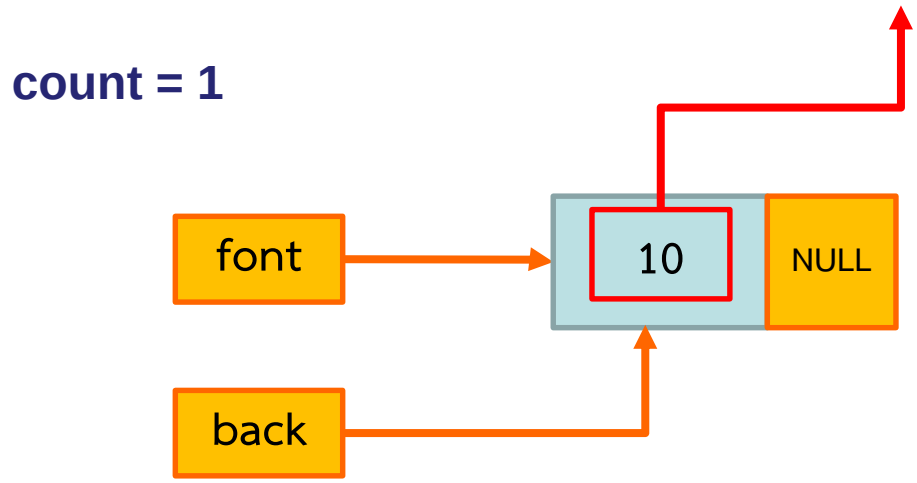
แนวคิดการนำข้อมูลออกจากคิว [5]

obj_queueLink.dequeue( );



แนวคิดการนำข้อมูลออกจากคิว [6]

obj_queueLink.dequeue( );

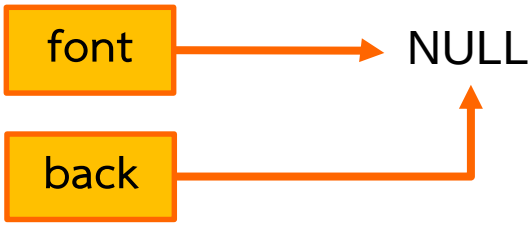


* กรณีนำข้อมูลตัวสุดท้ายออกจากคิว

แนวคิดการนำข้อมูลออกจากคิว [7]

obj_queueLink.dequeue( );

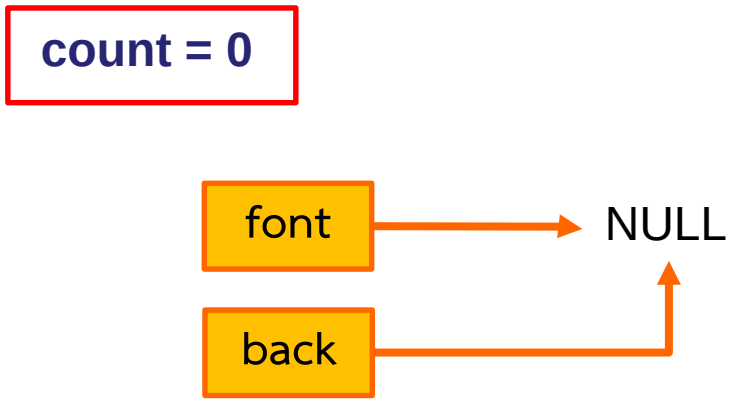
count = 1



* กรณีนำข้อมูลตัวสุดท้ายออกจากคิว

แนวคิดการนำข้อมูลออกจากคิว [8]

obj_QueueLink.dequeue( );



* กรณีนำข้อมูลตัวสุดท้ายออกจากคิว

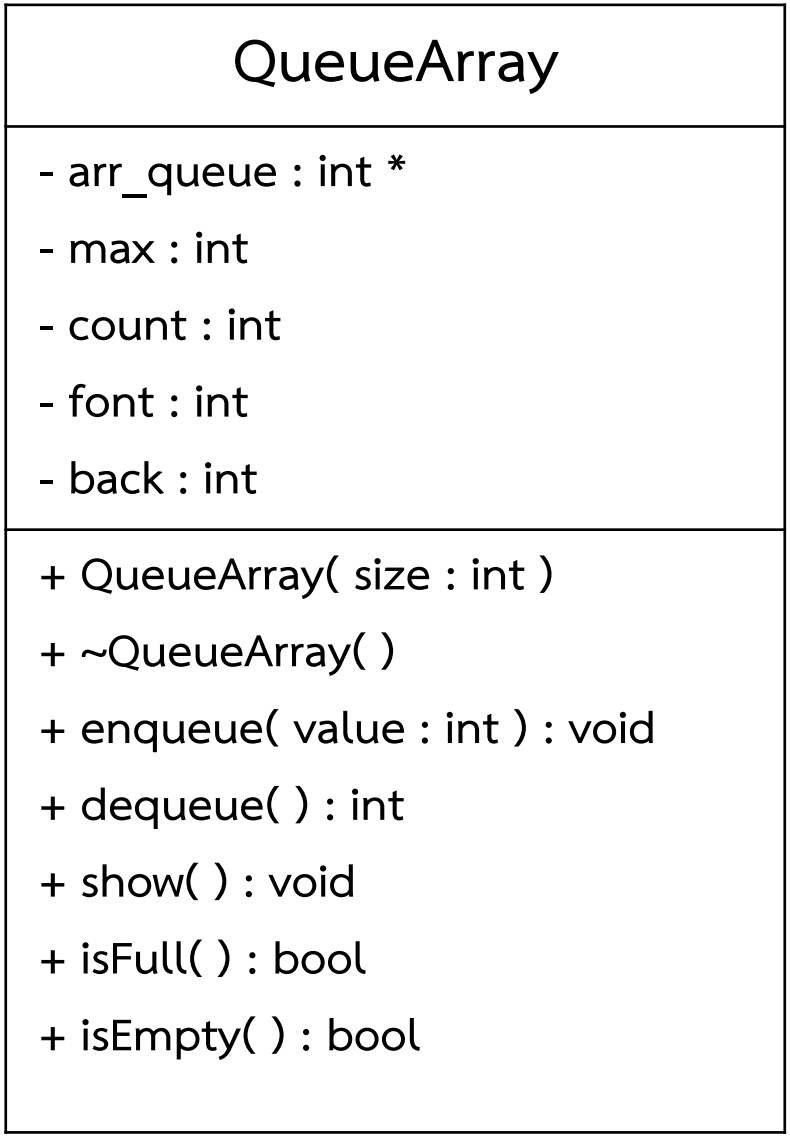
แบบฝึกหัดที่ 1

1. สร้างคลาสคิวด้วยอาร์เรย์ชื่อ QueueArray เพื่อจัดเก็บข้อมูลเลขจำนวนเต็ม โดยมีความสามารถในการจัดการข้อมูล ดังนี้

- สามารถนำข้อมูลเข้าคิว
- สามารถนำข้อมูลออกจากคิว
- สามารถแสดงข้อมูลทั้งหมด
- สามารถตรวจสอบพื้นที่เต็มในการเก็บข้อมูล
- สามารถตรวจสอบพื้นที่ว่างในการเก็บข้อมูล

2. นำคลาสที่สร้างขึ้นไปทดสอบการใช้งานในฟังก์ชัน main โดยทำการสร้างเป็นลักษณะเมนูสำหรับทดลองทุกความสามารถที่มีในคลาส queueArray

Class ของแบบฝึกหัดที่ 1



แบบฝึกหัดที่ 2

1. สร้างคลาสคิวด้วยพอยเตอร์ชื่อ QueueLink เพื่อจัดเก็บข้อมูลเลขจำนวนเต็ม โดยมีความสามารถในการจัดการข้อมูล ดังนี้

- สามารถนำข้อมูลเข้าคิว
- สามารถนำข้อมูลออกจากคิว
- สามารถแสดงข้อมูลทั้งหมด
- สามารถตรวจสอบพื้นที่ว่างในการเก็บข้อมูล
- สามารถตรวจสอบจำนวนข้อมูลที่มีทั้งหมด

2. นำคลาสที่สร้างขึ้นไปทดสอบการใช้งานในฟังก์ชัน main โดยทำการสร้างเป็นลักษณะเมนูสำหรับทดลองทุกความสามารถที่มีในคลาส queueLink

Class ของแบบฝึกหัดที่ 2