บทที่ 7 Stack

บทเรียนย่อย

7.1 Stack Operations and Concept

7.2 Stack with Array Component

7.3 Stack with Array Implementation

7.4 Stack with Pointer Component

7.5 Stack with Pointer Implementation

7.6 Stack Apply

วัตถุประสงค์

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

บทที่ 7 Stack

บทเรียนย่อย

7.1 Stack Operations and Concept

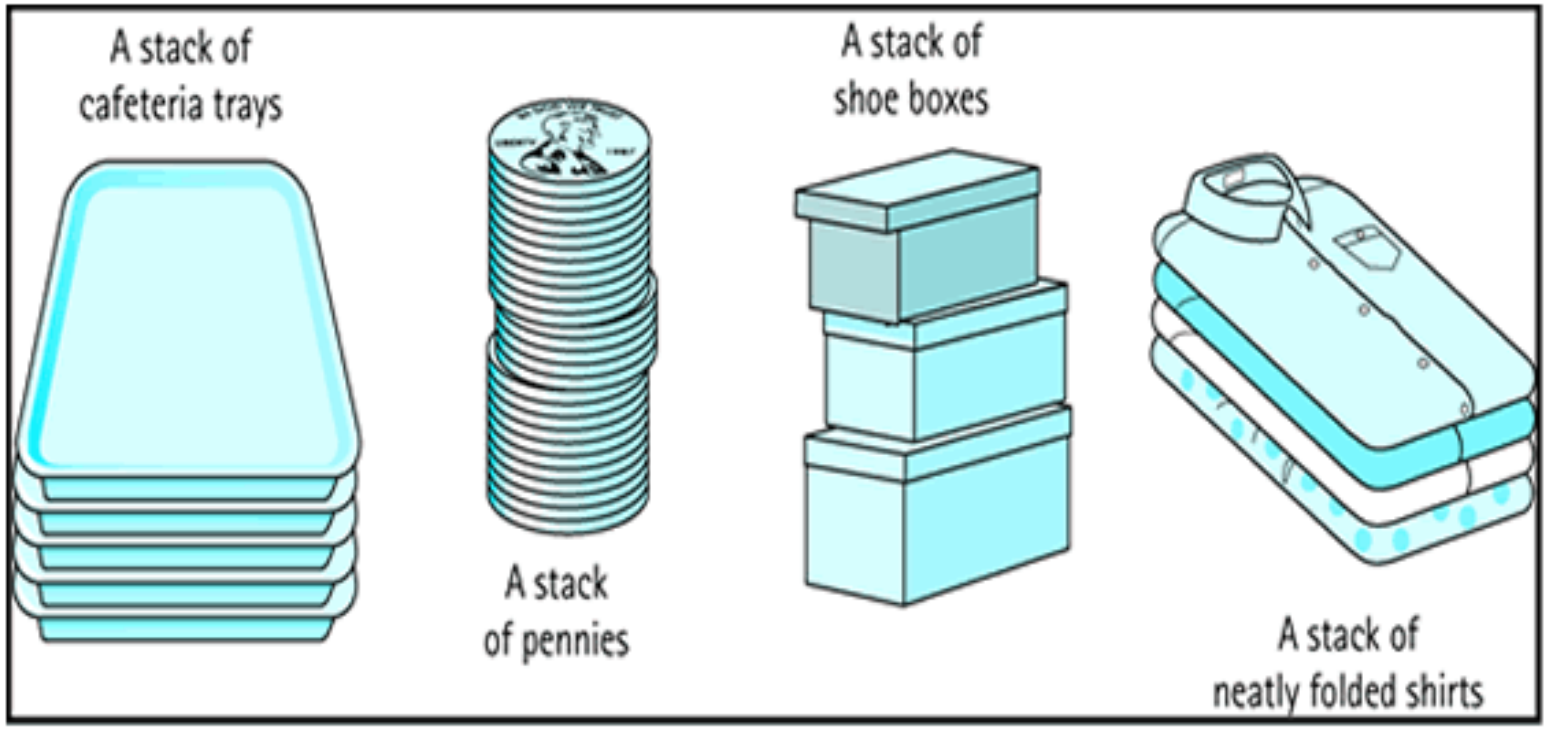
Stack (สแตก) คือ โครงสร้างข้อมูลชนิดหนึ่งที่ออกแบบมาให้มีลักษณะทั้งแบบ Linear Structure (เชิงเส้น) และแบบ Non Linear Structure (แบบไม่เชิงเส้น) ซึ่งนำไปใช้กับการเขียนโปรแกรมได้ ทั้งการใช้ Array (อาร์เรย์) และ Pointer (พอยเตอร์) โดยการนำข้อมูลเข้าและออกจากสแตกจะมีลำดับการทำงานแบบ “เข้าหลังออกก่อน” (Last In First Out) หรือเรียกสั้น ๆ ว่า LIFO เนื่องจากการนำข้อมูลเข้าและออก จะใช้ปลายด้านเดียวกันจึงทำให้ข้อมูลตัวที่นำเข้าไปเก็บก่อนถูกจัดเก็บด้านในสุด และข้อมูลตัวที่จัดเก็บตัวสุดท้ายจะอยู่บนสุด การนำข้อมูลออกจึงต้องนำข้อมูลตัวบนสุดออกก่อน

ลักษณะของ Stack

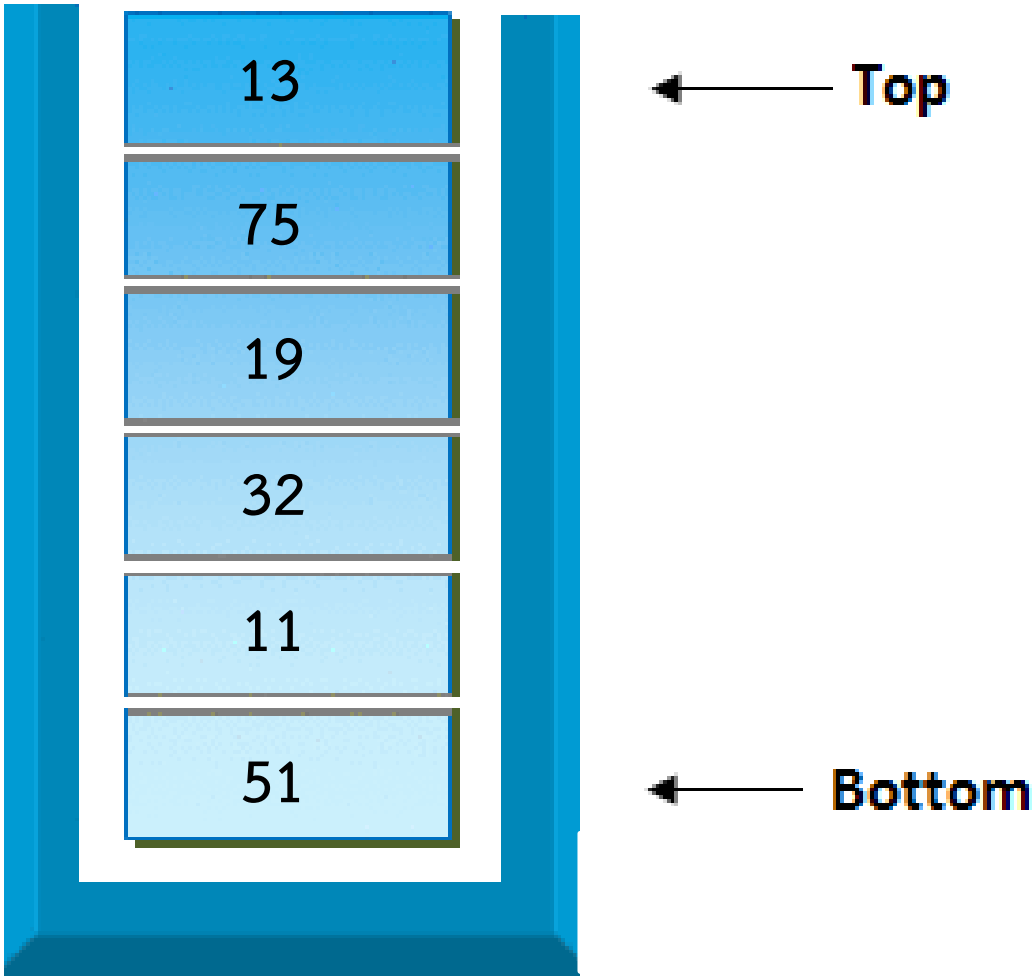
ลักษณะสำคัญของ Stack มีดังนี้

- เป็นโครงสร้างข้อมูลทั้ง 2 ชนิด คือ Linear Structure (เชิงเส้น) และแบบ Non Linear Structure (แบบไม่เชิงเส้น)
- มีทางเข้าและออกของข้อมูลทางเดียว
- มีการทำงานแบบตามลำดับ
- สามารถนำข้อมูลเข้าและนำข้อมูลออกสลับกันได้
- มีลำดับการทำงานแบบเข้าหลังออกก่อน (LIFO)

ลักษณะของ Stack [2]



ลักษณะของ Stack [3]



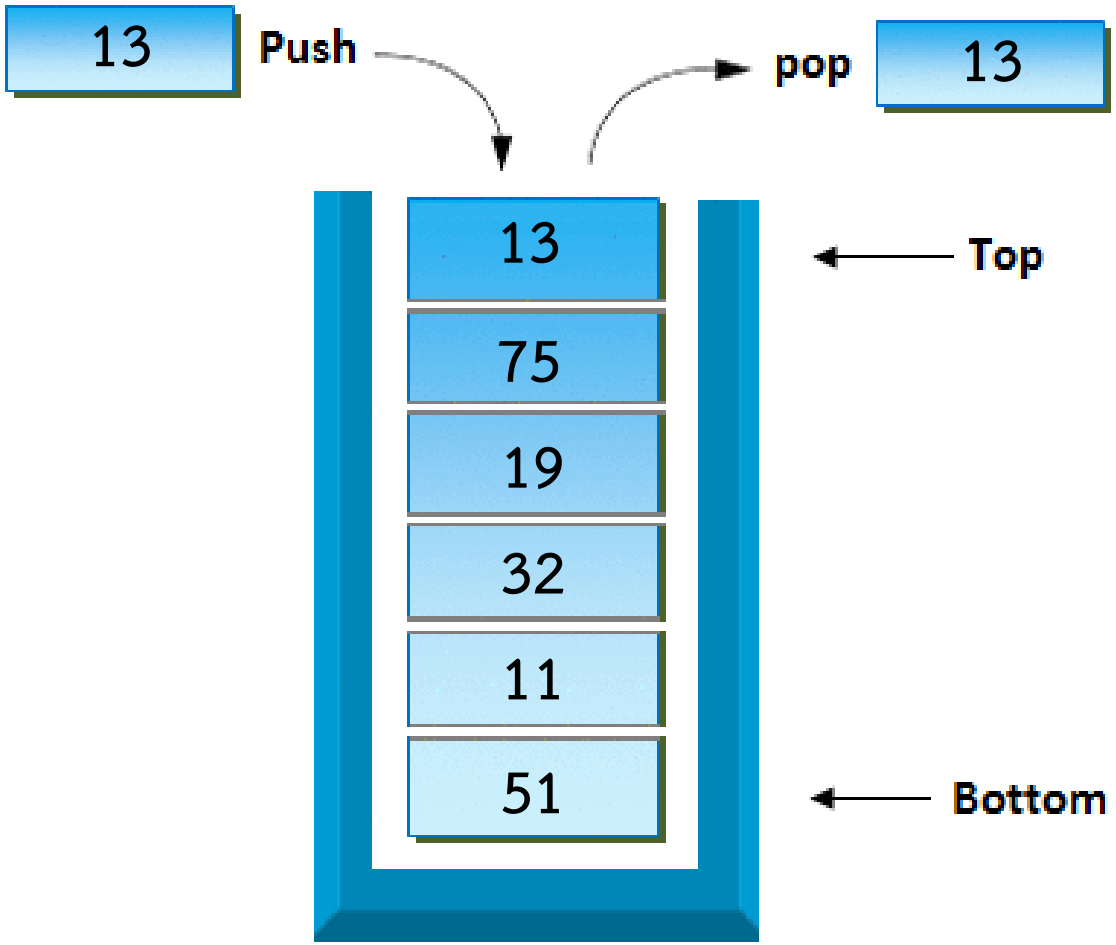
การดำเนินการของ Stack

การดำเนินการของ Stack มี 2 ส่วน ดังนี้

- การนำข้อมูลเข้าไปเก็บในสแตก ซึ่งเรียกว่า Push Stack (การพุชสแตก) คือ การนำข้อมูลไปเก็บไว้ที่ช่องว่างบนสุดของสแตก (Top)

- การนำข้อมูลออกจากสแตก ซึ่งเรียกว่า Pop Stack (การป็อปสแตก) คือ การนำข้อมูลตัวที่เก็บไว้บนสุดของสแตกออกไปใช้งาน (Top)

การดำเนินการของ Stack [2]



บทที่ 7 Stack

บทเรียนย่อย

7.1 Stack Operations and Concept

7.2 Stack with Array Component

7.3 Stack with Array Implementation

7.4 Stack with Pointer Component

7.5 Stack with Pointer Implementation

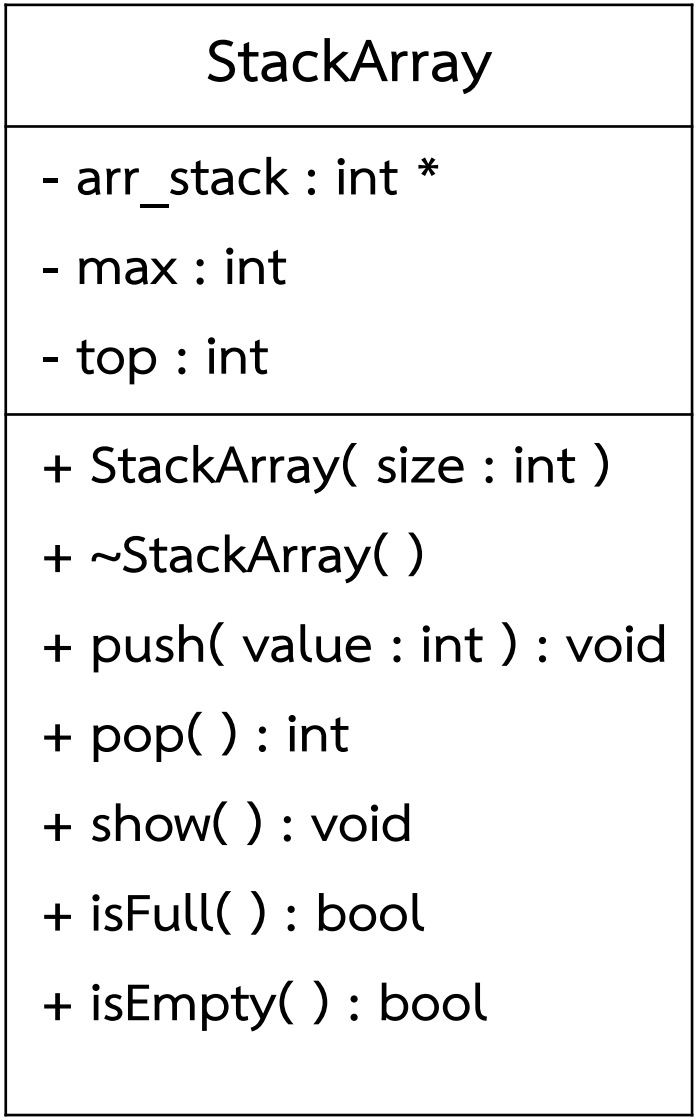
7.6 Stack Apply

7.2 Stack with Array Component

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

คุณสมบัติ (Property)	กระบวนการทำงาน (method)
arr_stack	push
max	pop
top	show
	isFull
	isEmpty

Stack with Array Class



รายละเอียดคุณสมบัติของ Stack with Array

คุณสมบัติ (Property)	รายละเอียด
arr_stack	ตัวแปรอาร์เรย์สำหรับเก็บข้อมูลในรูปแบบสแตก
max	ตัวแปรสำหรับเก็บจำนวนพื้นที่สูงสุดที่สามารถจัดเก็บข้อมูลได้
top	ตัวแปรสำหรับเก็บตำแหน่งบนสุดของสแตก

รายละเอียดกระบวนการทำงานของ Stack with Array

กระบวนการทำงาน (method)	รายละเอียดการทำงาน
StackArray(int size)	Constructor สำหรับสร้าง Array โดยระบุขนาดสูงสุด ตามค่าของพารามิเตอร์ที่ส่งเข้ามา
~StackArray( )	Deconstructor สำหรับลบข้อมูลที่กำหนดขึ้นออกจากหน่วยความจำ
push(int value)	เพิ่มข้อมูลโดยนำไปเก็บไว้ที่ช่องว่างบนสุดของสแตก
int pop( )	นำข้อมูลตัวที่เก็บไว้บนสุดของสแตกออกไปใช้งาน

รายละเอียดกระบวนการทำงานของ Stack with Array [2]

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

บทที่ 7 Stack

บทเรียนย่อย

7.1 Stack Operations and Concept

7.2 Stack with Array Component

7.3 Stack with Array Implementation

7.4 Stack with Pointer Component

7.5 Stack with Pointer Implementation

7.6 Stack Apply

7.3 Stack with Array Implementation

การสร้างคลาส Stack ด้วยภาษา C++

```
class StackArray {
    private :
        int * arr_stack;
        int max;
        int top;
     public :
        StackArray( int size );
        ~StackArray( );
                ...
};
```

7.3 Stack with Array Implementation [2]

การสร้างคลาส Stack ด้วยภาษา C++

```
class StackArray {
        ...
    push( int  value );
    int pop( );
    show( );
    bool isFull( );
    bool isEmpty( );
};
```

การดำเนินการใน Constructor และ Deconstructor

```
StackArray :: StackArray( int size ){
        arr_stack = new int[ size ];
        max = size;
        top = -1;
}

StackArray :: ~ StackArray( ){
      delete [] arr_stack;
}
```

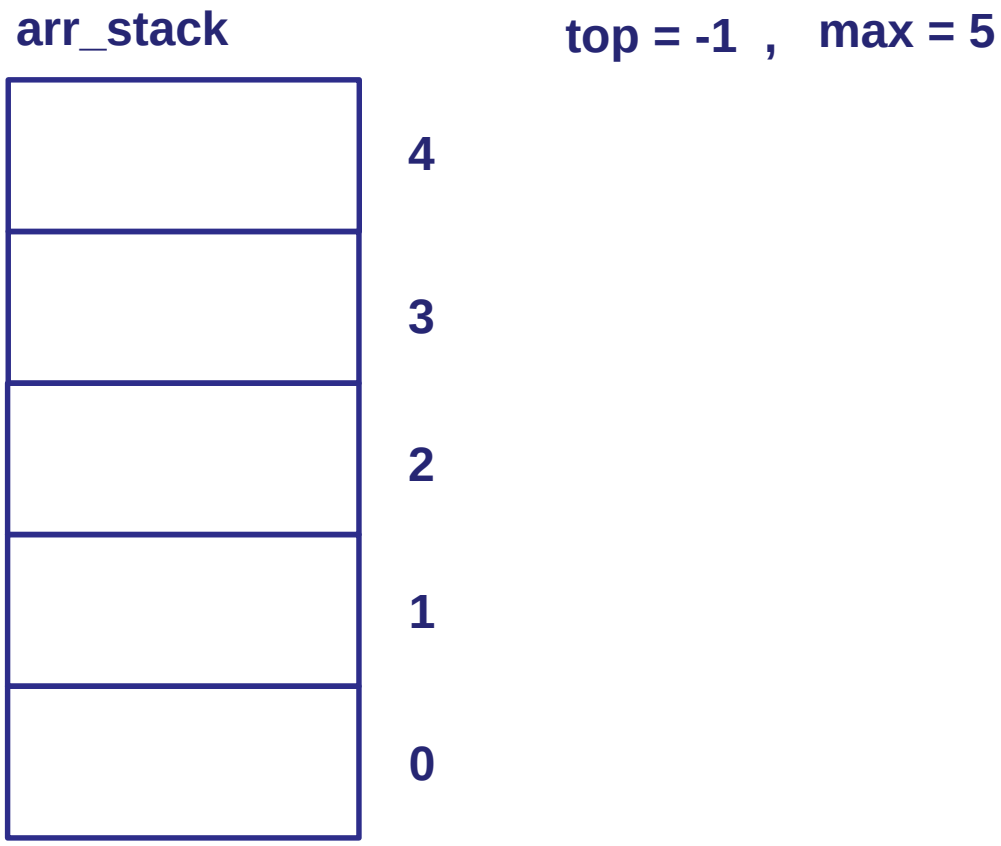
การเรียกใช้งานคลาส StackArray

```
int main(void){
    StackArray * obj_stackArr = new StackArray(5);
    obj_stackArr->push(5);
    obj_stackArr->push(10);
    obj_stackArr->show();

    StackArray obj_stackArr(5);
    obj_stackArr.push(5);
    obj_stackArr.push(10);
    obj_stackArr.show();
}
```

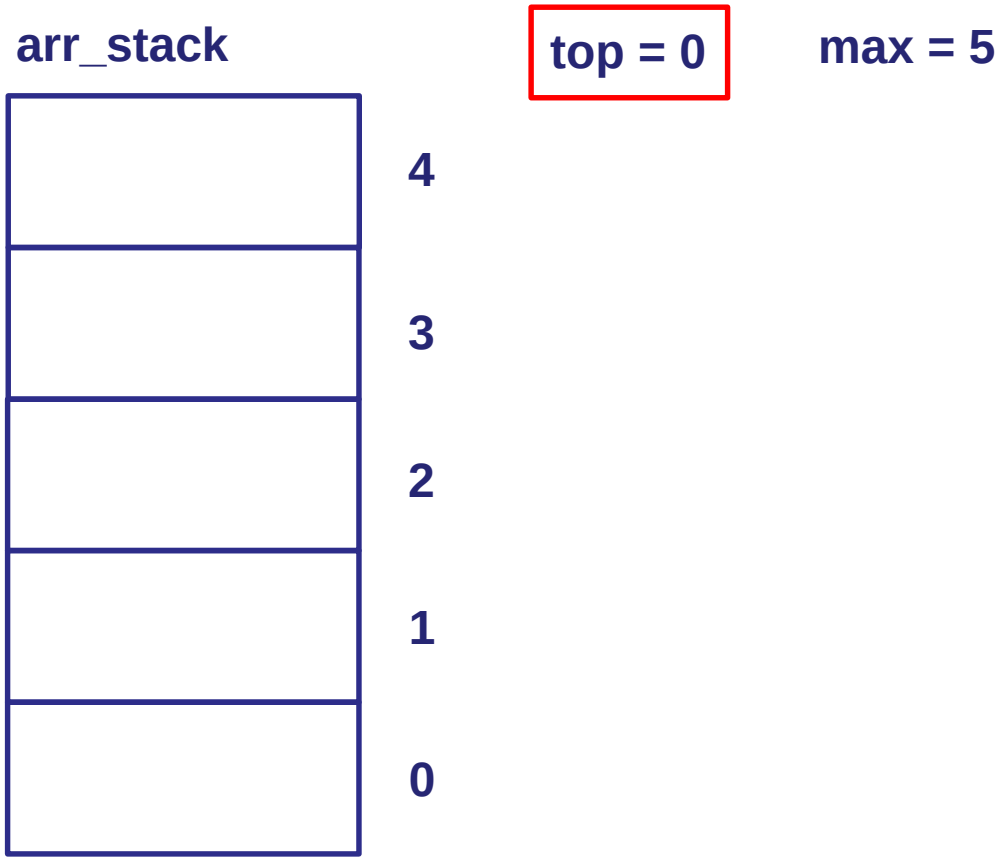
แนวคิดการกำหนดค่าเริ่มต้นของสแตก

StackArray obj_stackArr(5);



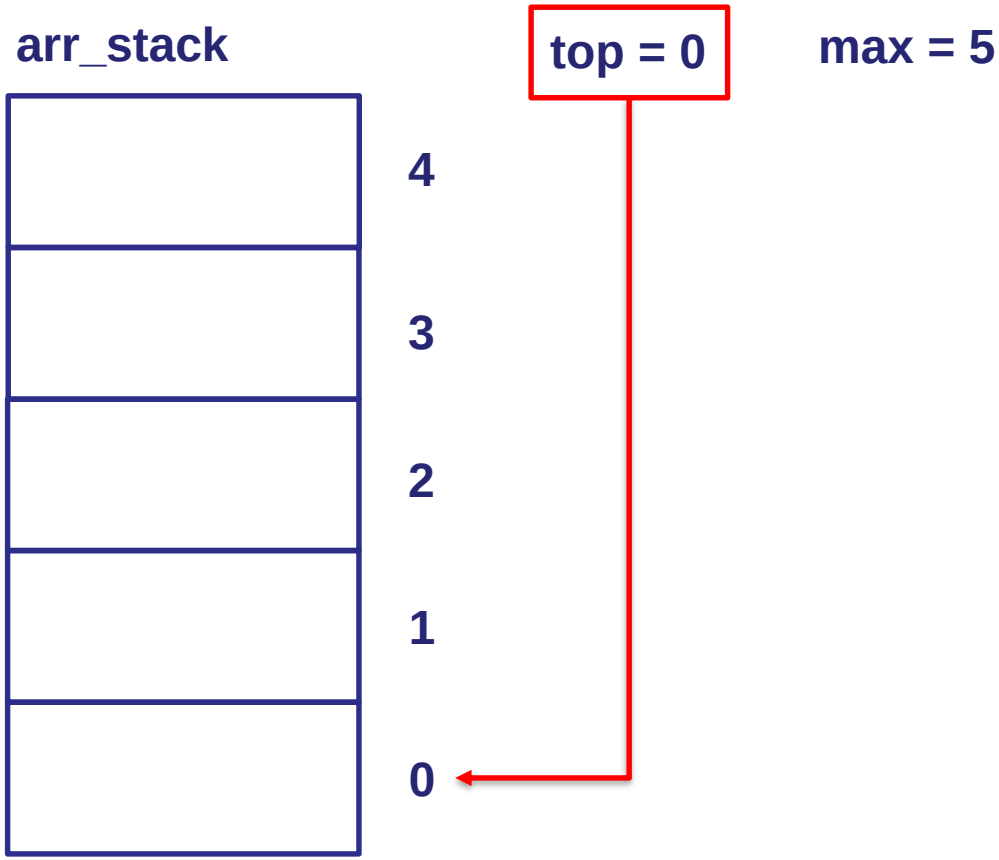
แนวคิดการนำข้อมูลเข้าสแตก

obj_stackArr.push(5);



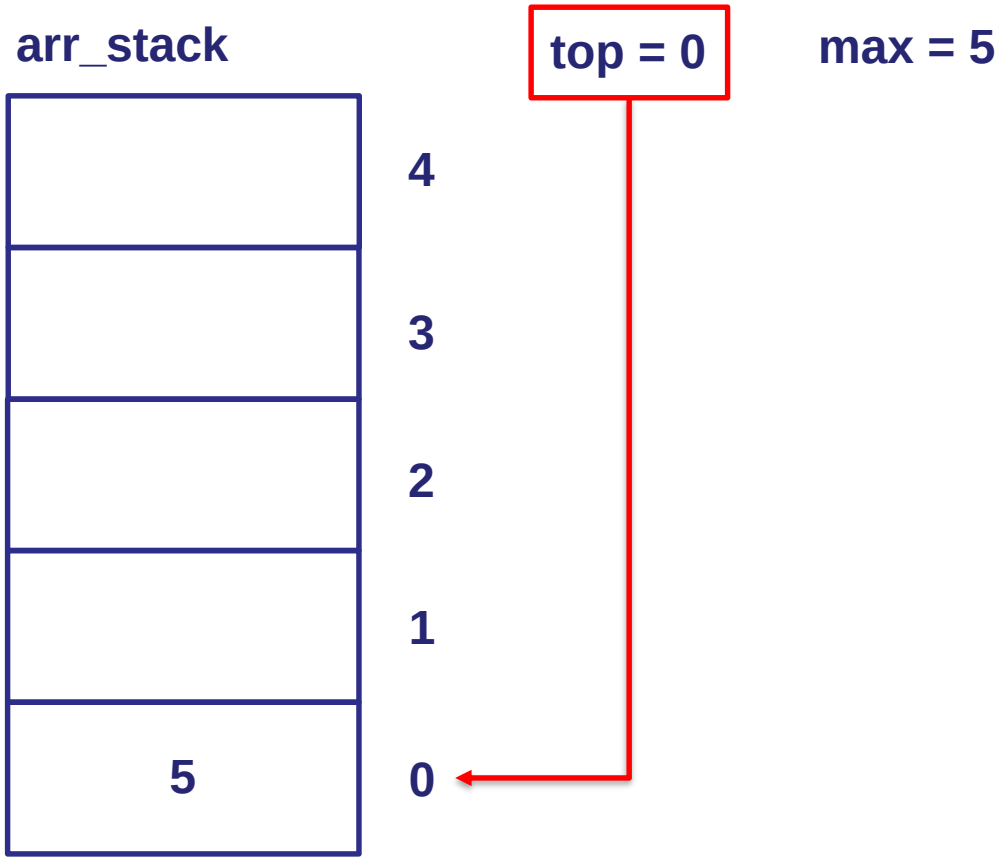
แนวคิดการนำข้อมูลเข้าสแตก [2]

obj_stackArr.push(5);



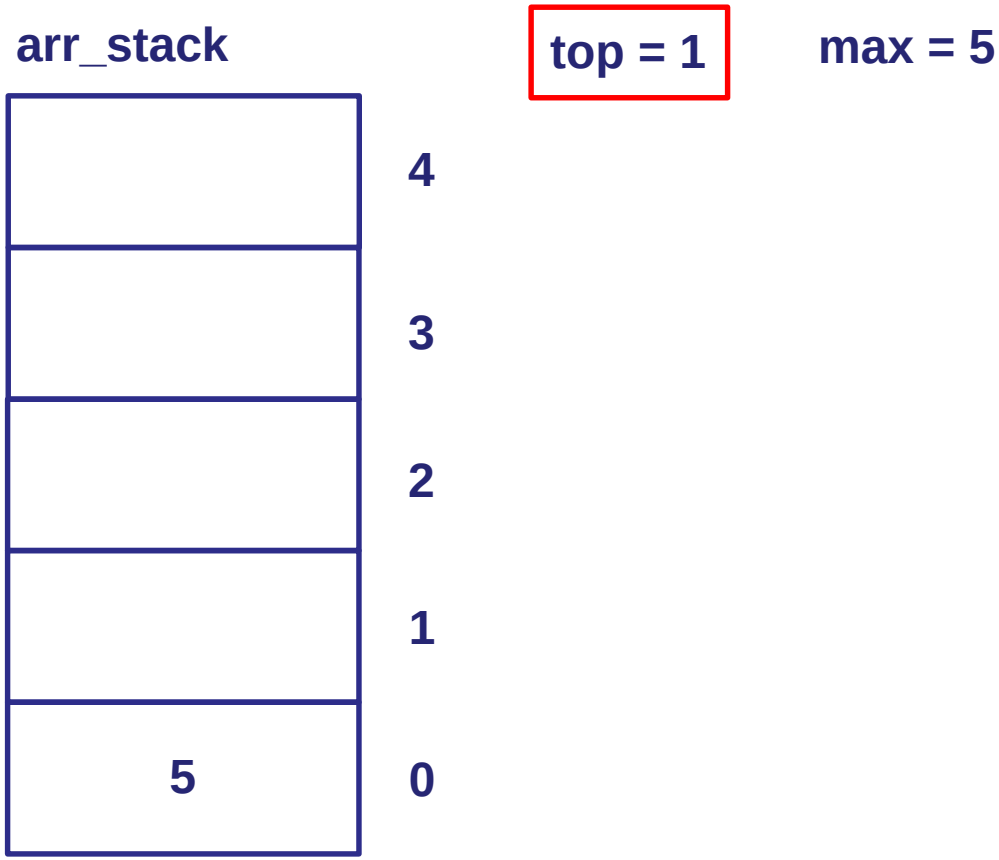
แนวคิดการนำข้อมูลเข้าสแตก [3]

obj_stackArr.push(5);



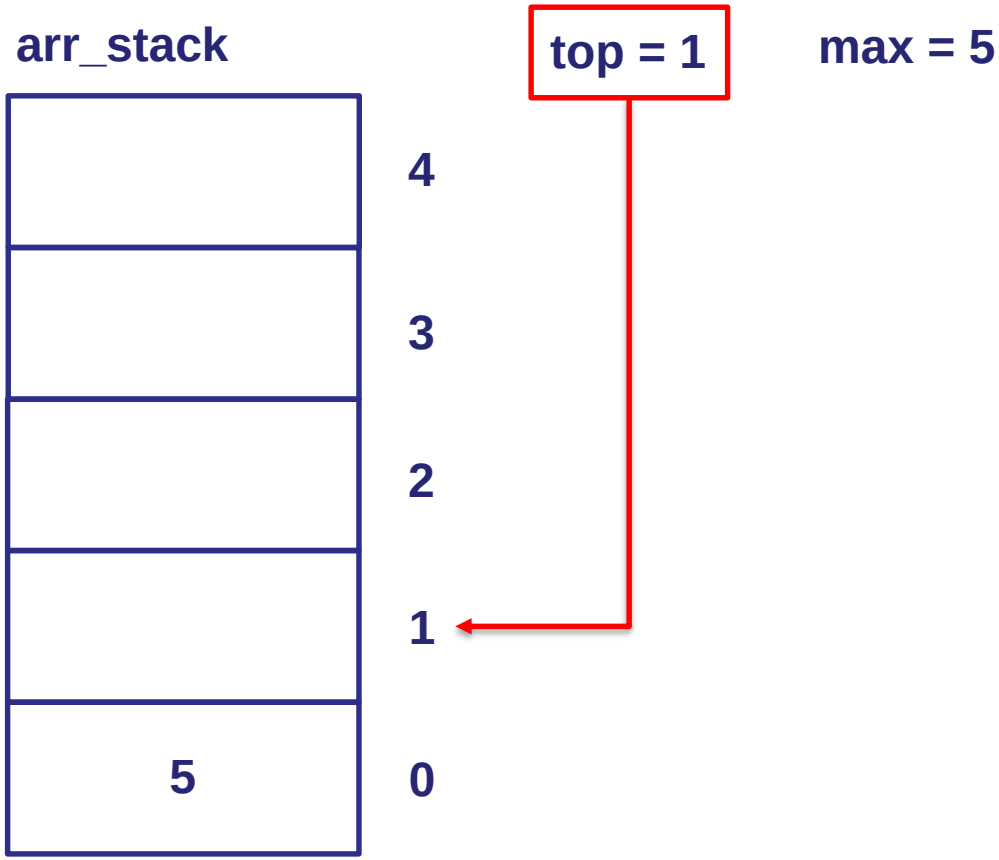
แนวคิดการนำข้อมูลเข้าสแตก [4]

obj_stackArr.push(10);



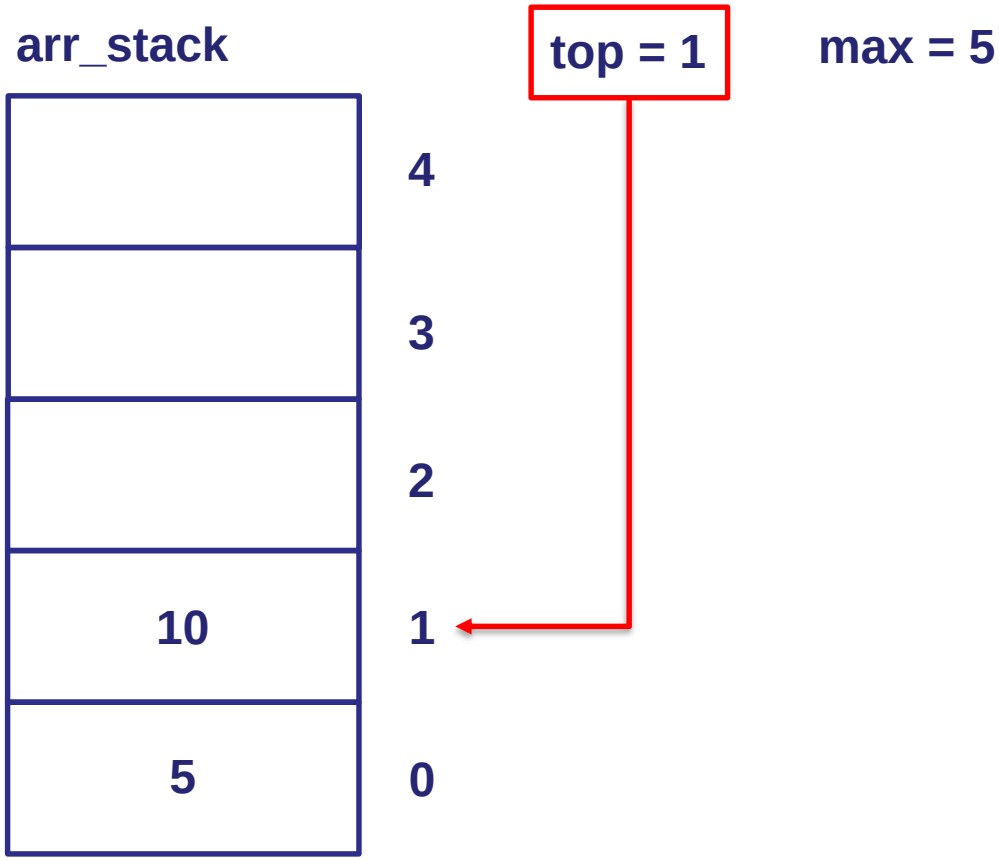
แนวคิดการนำข้อมูลเข้าสแตก [5]

obj_stackArr.push(10);



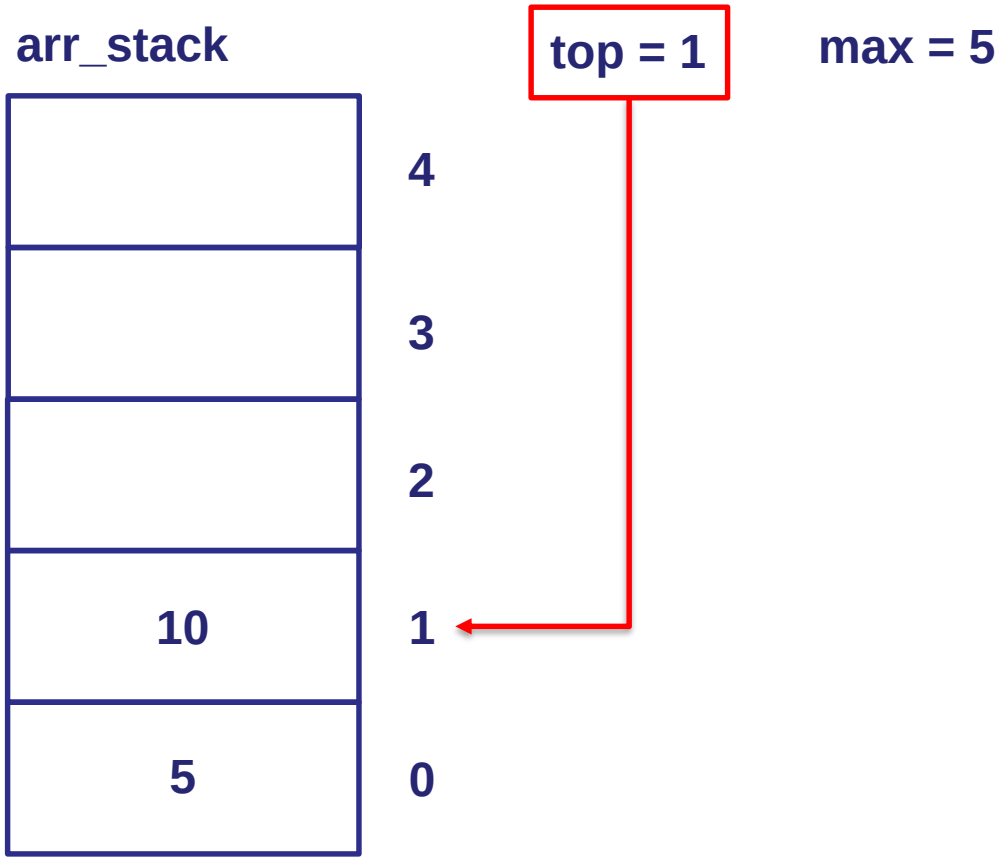
แนวคิดการนำข้อมูลเข้าสแตก [6]

obj_stackArr.push(10);



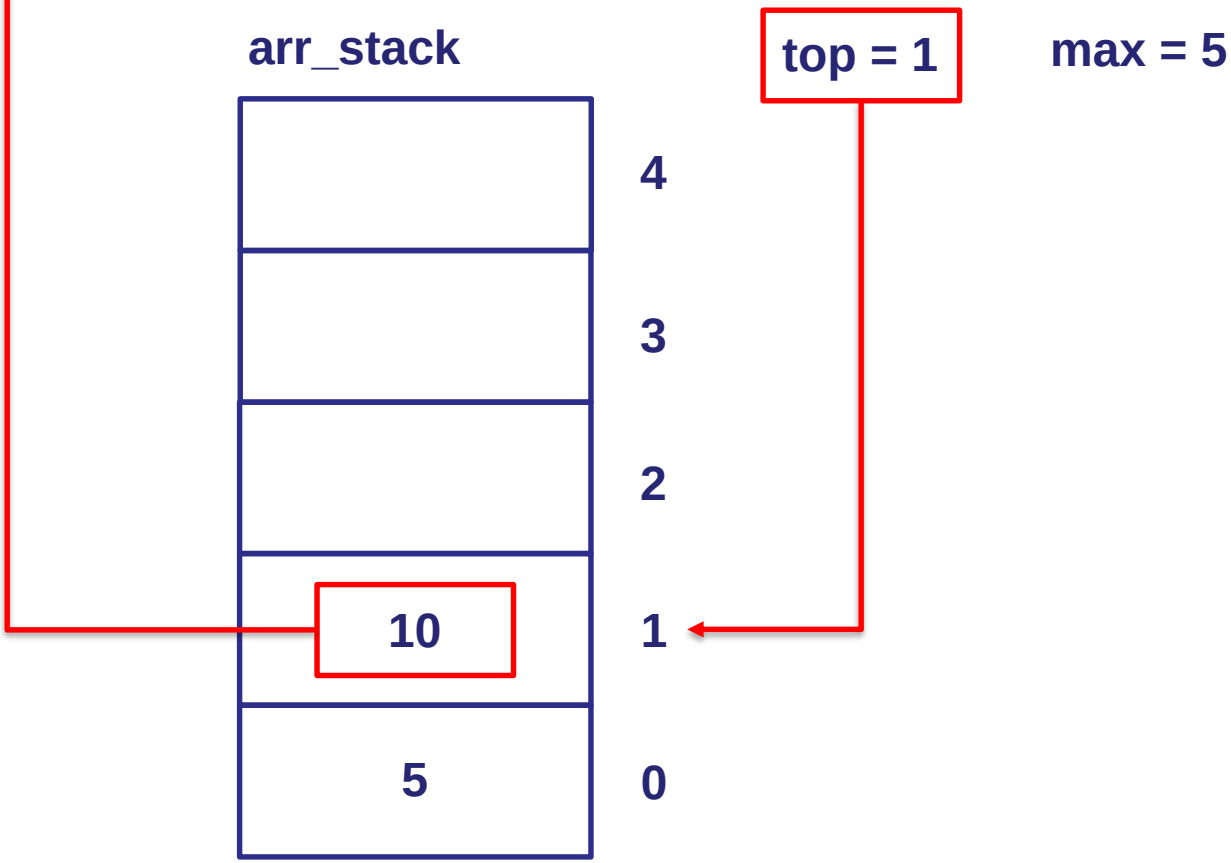
แนวคิดการนำข้อมูลออกจากสแตก

obj_stackArr.pop( );



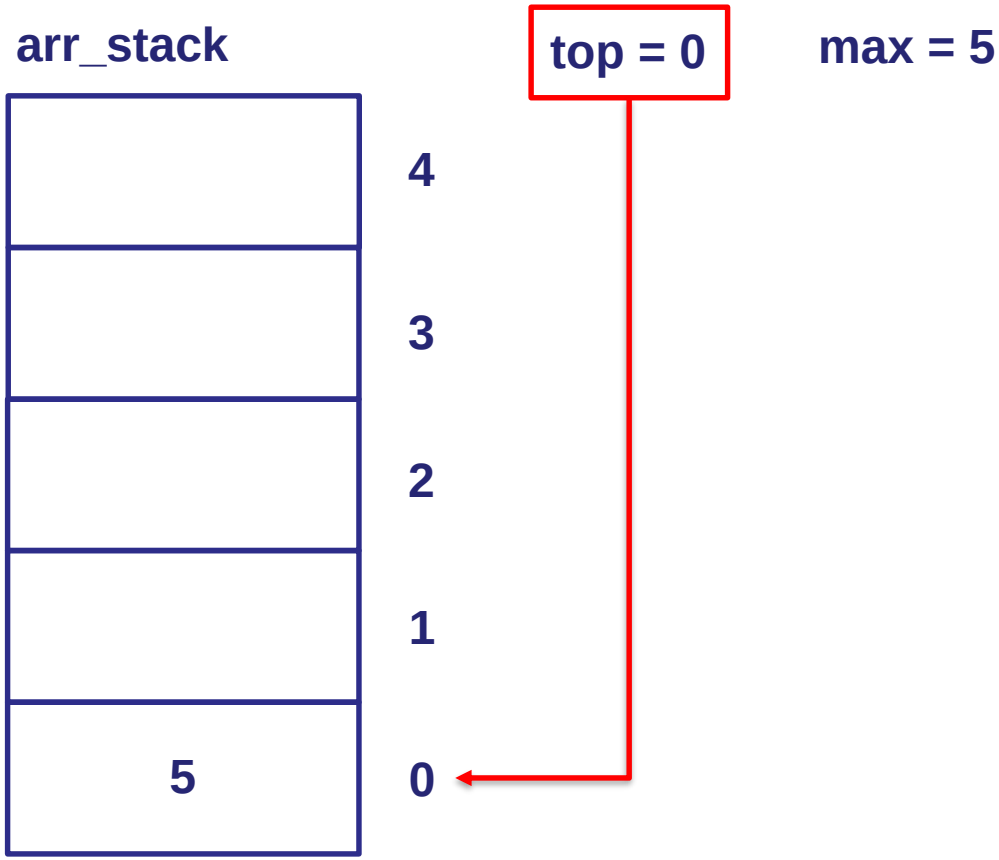
แนวคิดการนำข้อมูลออกจากสแตก [2]

obj_stackArr.pop( );



แนวคิดการนำข้อมูลออกจากสแตก [3]

obj_stackArr.pop( );



บทที่ 7 Stack

บทเรียนย่อย

7.1 Stack Operations and Concept

7.2 Stack with Array Component

7.3 Stack with Array Implementation

7.4 Stack with Pointer Component

7.5 Stack with Pointer Implementation

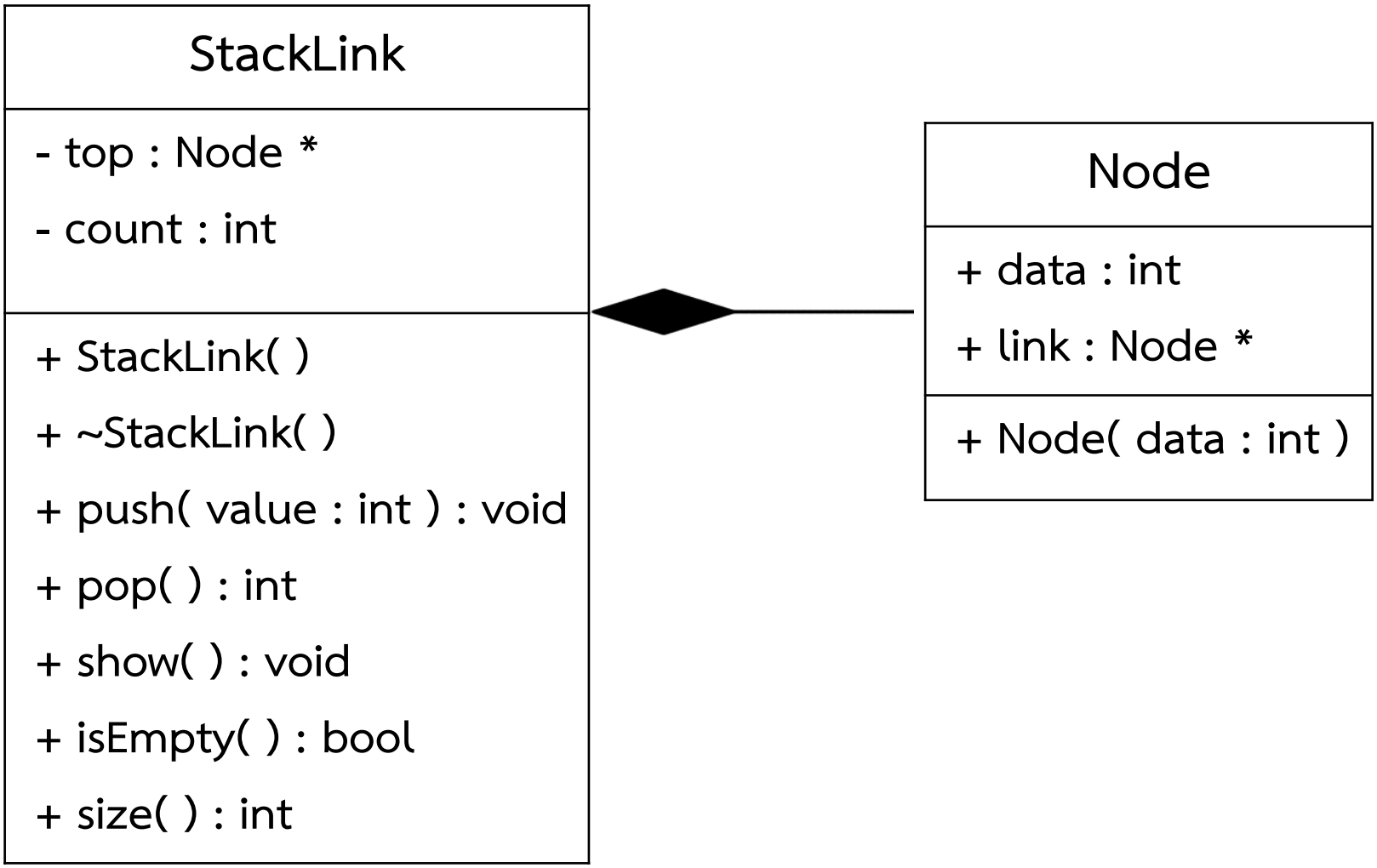
7.6 Stack Apply

7.4 Stack with Pointer Component

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

คุณสมบัติ (Property)	กระบวนการทำงาน (method)
class node	push
top	pop
count	show
	isEmpty
	size

Stack with Pointer Class



รายละเอียดคุณสมบัติของ Stack with Pointer

คุณสมบัติ (Property)	รายละเอียด
Node	เป็น Class ที่เก็บข้อมูล และเก็บที่อยู่ของตัวถัดไป
top	ตัวแปรสำหรับเก็บที่อยู่ของข้อมูลตัวบนสุดของสแตก
count	ตัวแปรสำหรับการใช้นับจำนวนข้อมูลที่เก็บไว้ทั้งหมด

รายละเอียดกระบวนการทำงานของ Stack with Pointer

กระบวนการทำงาน (method)	รายละเอียดการทำงาน
StackLink( )	Constructor สำหรับกำหนดค่าเริ่มต้น top = NULL และ count = 0
~StackLink( )	Deconstructor สำหรับลบข้อมูลที่กำหนดขึ้นออกจากหน่วยความจำ
push(int value)	เพิ่มข้อมูลโดยนำไปเก็บไว้ที่บนสุดของสแตก
int pop( )	นำข้อมูลตัวที่เก็บไว้บนสุดของสแตกออกไปใช้งาน

รายละเอียดกระบวนการทำงานของ Stack with Pointer [2]

กระบวนการทำงาน (method)	รายละเอียดการทำงาน
show( )	แสดงผลข้อมูลที่มีในสแตกทั้งหมด ผ่านทางหน้าจอ
bool isEmpty( )	ตรวจสอบข้อมูลในสแตก โดยถ้าไม่มีข้อมูลจะคืนค่า TRUE หากมีข้อมูลจะคืนค่า FALSE
int size( )	คืนค่าจำนวนข้อมูลที่มีอยู่ในสแตก

บทที่ 7 Stack

บทเรียนย่อย

7.5 Stack with Pointer Implementation

การสร้างคลาส Stack ด้วยภาษา C++

```
class StackLink {
    private :
        Node * top;
        int count;
    public :
        StackLink( );
        ~ StackLink( );
            ...
};
```

7.5 Stack with Pointer Implementation [2]

การสร้างคลาส Stack ด้วยภาษา C++

```
class StackArray {
        ...
    push( int  value );
    int pop( );
    show( );
    bool isEmpty( );
    int size( );
};
```

การดำเนินการใน Constructor และ Deconstructor

```
StackLink :: StackLink( int size ){
        top = NULL;
        count = 0;
}

StackLink :: ~ StackLink( ){
      for(Node * tmp = top; tmp != NULL; top = top->link){
            delete tmp;
             tmp = NULL;
       }
      top = NULL;
}
```

การเรียกใช้งานคลาส StackArray

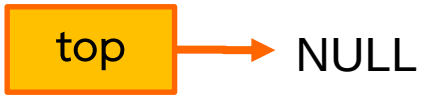
```
int main(void){
    StackLink * obj_stackLink = new StackLink( );
    obj_stackLink->push(5);
    obj_stackLink- >push(10);
    obj_stackLink- >show();

    StackArray obj_stackLink;
    obj_stackLink.push(5);
    obj_stackLink.push(10);
    obj_stackLink.show();
}
```

แนวคิดการกำหนดค่าเริ่มต้นของสแตก

StackLink obj_stackLink;

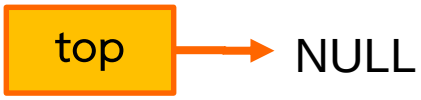
count = 0



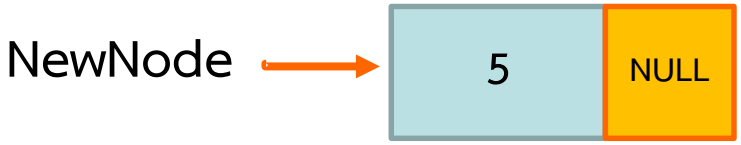
แนวคิดการนำข้อมูลเข้าสแตก

obj_stackLink.push(5);

count = 0



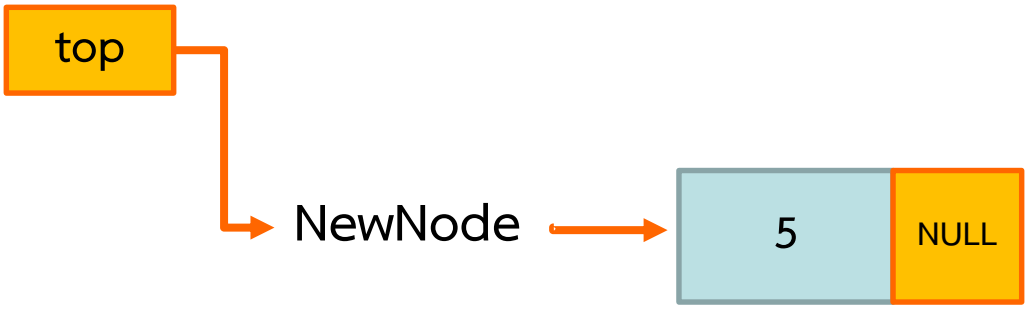
สร้าง new object จาก class node โดยกำหนดค่า data มีค่าเท่ากับ 5 และ link มีค่าเป็น NULL



แนวคิดการนำข้อมูลเข้าสแตก [2]

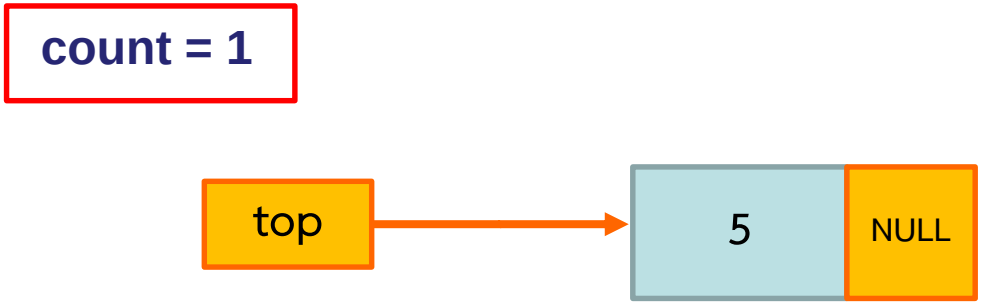
obj_stackLink.push(5);

count = 0



แนวคิดการนำข้อมูลเข้าสแตก [3]

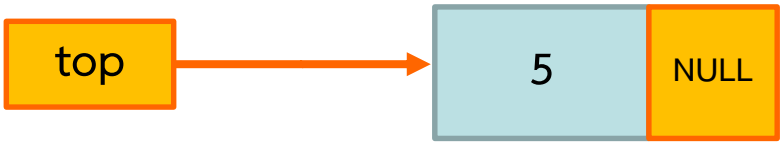
obj_stackLink.push(5);



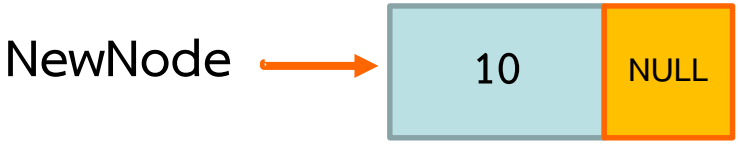
แนวคิดการนำข้อมูลเข้าสแตก [4]

obj_stackLink.push(10);

count = 1



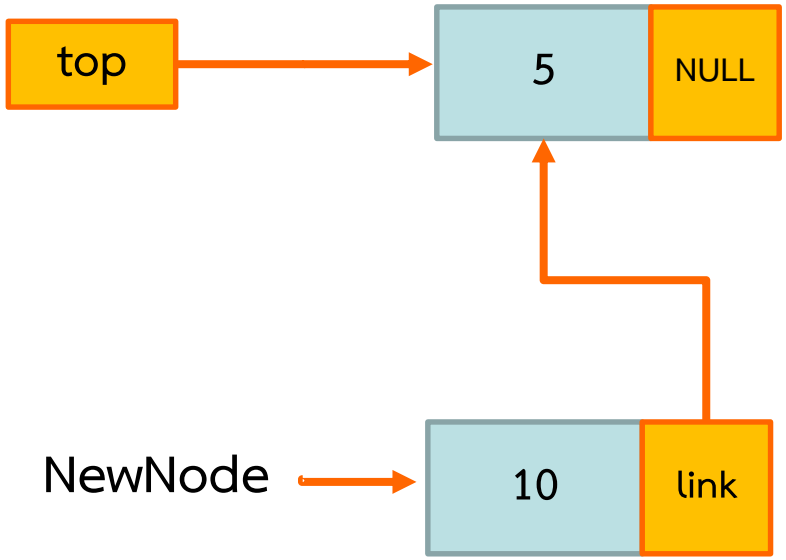
สร้าง new object จาก class node โดยกำหนดค่า data มีค่าเท่ากับ 10 และ link มีค่าเป็น NULL



แนวคิดการนำข้อมูลเข้าสแตก [5]

obj_stackLink.push(10);

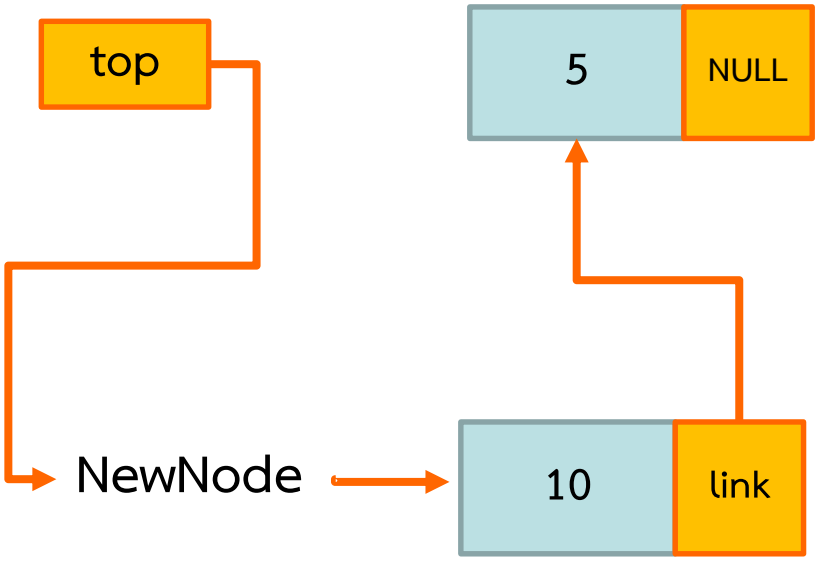
count = 1



แนวคิดการนำข้อมูลเข้าสแตก [6]

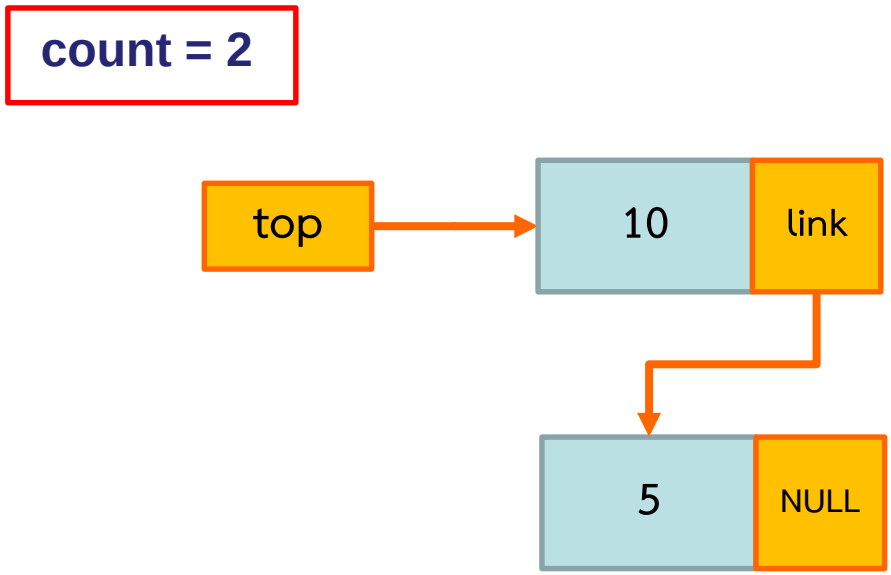
obj_stackLink.push(10);

count = 1

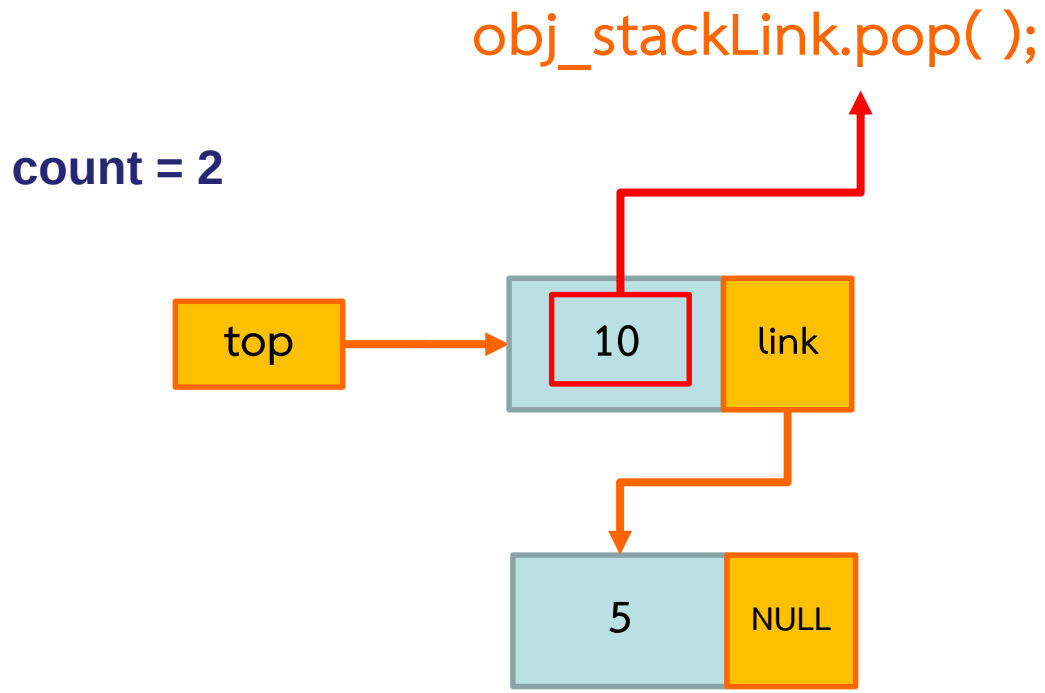


แนวคิดการนำข้อมูลเข้าสแตก [7]

obj_stackLink.push(10);



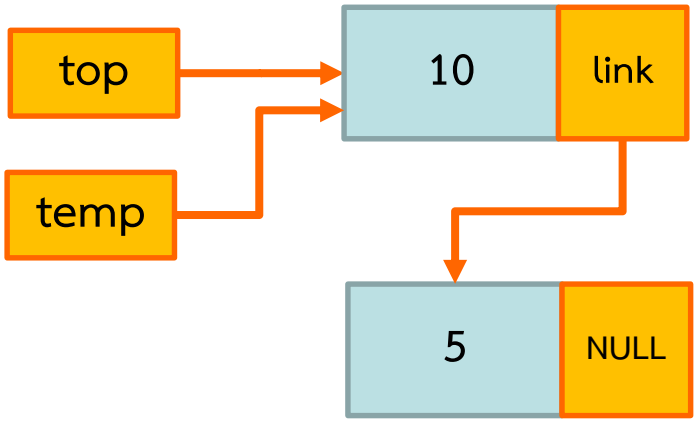
แนวคิดการนำข้อมูลออกจากสแตก



แนวคิดการนำข้อมูลออกจากสแตก [2]

obj_stackLink.pop( );

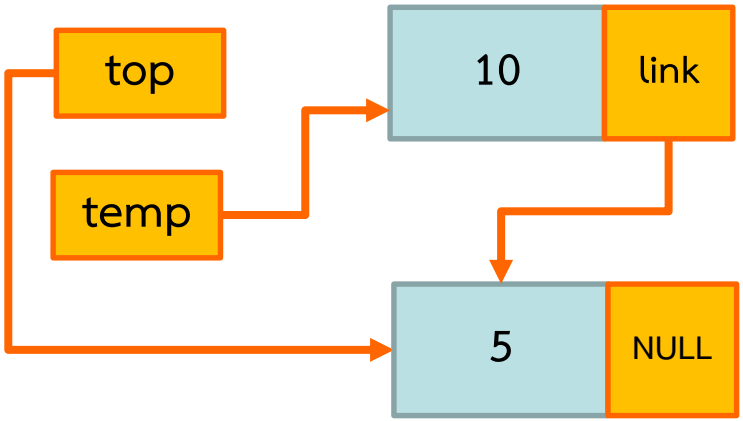
count = 2



แนวคิดการนำข้อมูลออกจากสแตก [3]

obj_stackLink.pop( );

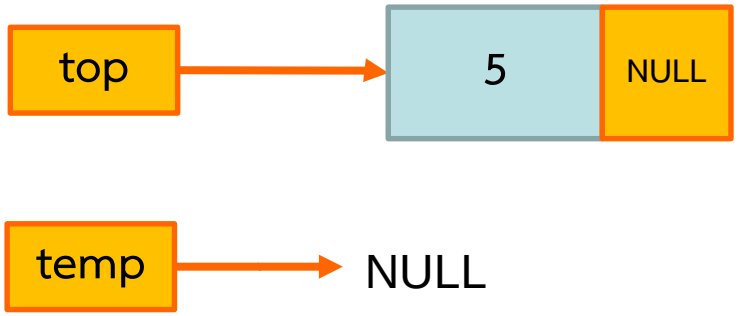
count = 2



แนวคิดการนำข้อมูลออกจากสแตก [4]

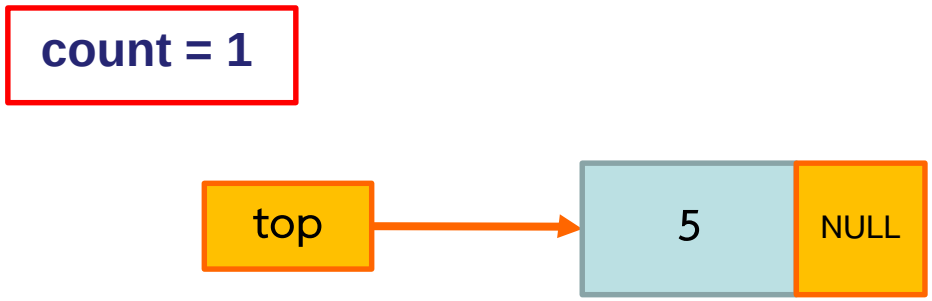
obj_stackLink.pop( );

count = 2

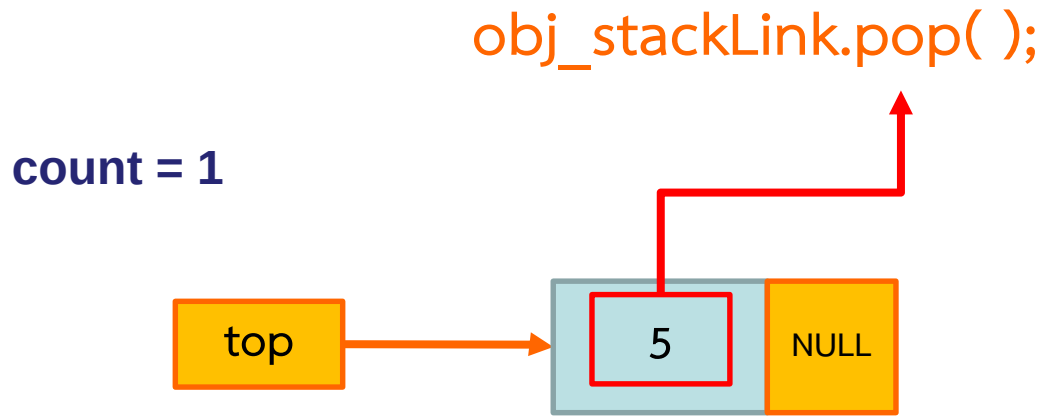


แนวคิดการนำข้อมูลออกจากสแตก [5]

obj_stackLink.pop( );



แนวคิดการนำข้อมูลออกจากสแตก [6]



หมายเหตุ : กรณีนำข้อมูลตัวสุดท้ายออกจากสแตก

แนวคิดการนำข้อมูลออกจากสแตก [7]

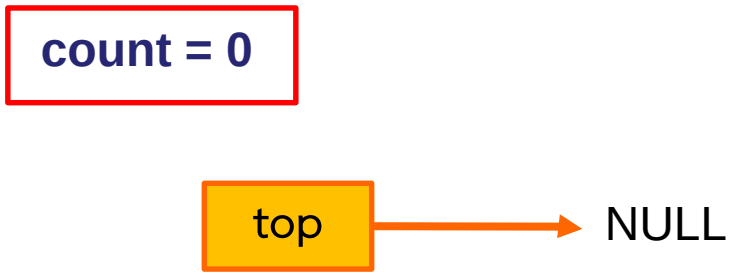
obj_stackLink.pop( );

count = 1

หมายเหตุ : กรณีนำข้อมูลตัวสุดท้ายออกจากสแตก

แนวคิดการนำข้อมูลออกจากสแตก [8]

obj_stackLink.pop( );



หมายเหตุ : กรณีนำข้อมูลตัวสุดท้ายออกจากสแตก

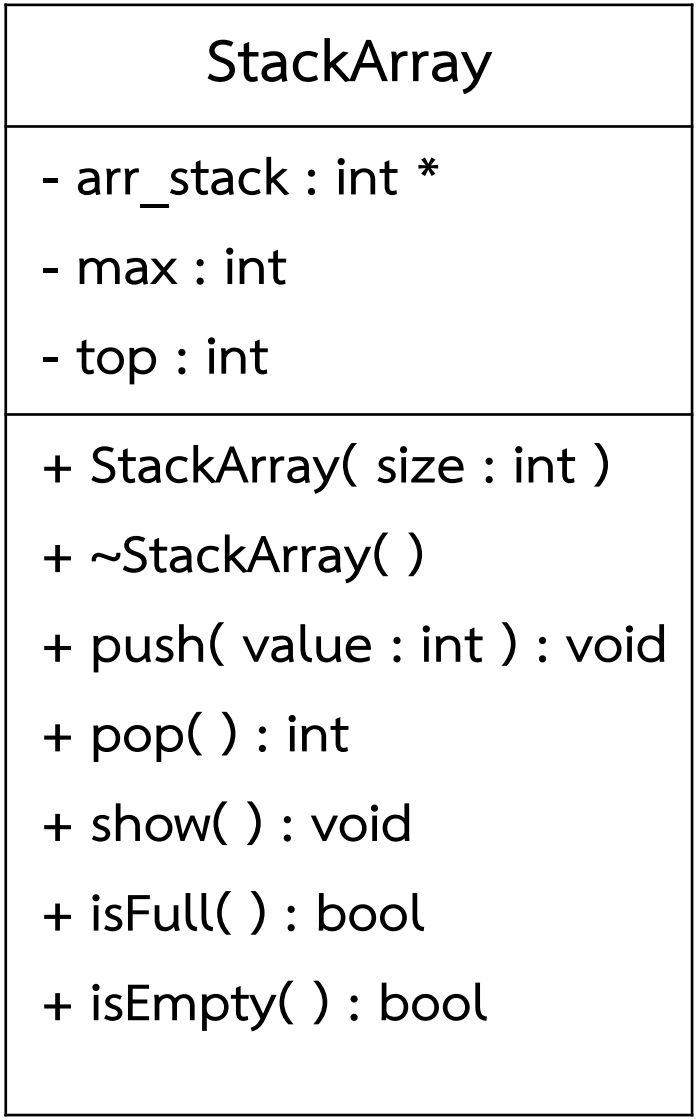
แบบฝึกหัดที่ 1

1. สร้างคลาสสแตกด้วยอาร์เรย์ชื่อ StackArray เพื่อจัดเก็บข้อมูลเลขจำนวนเต็ม โดยมีความสามารถในการจัดการข้อมูล ดังนี้

- สามารถนำข้อมูลเข้าสแตก
- สามารถนำข้อมูลออกจากสแตก
- สามารถแสดงข้อมูลทั้งหมด
- สามารถตรวจสอบพื้นที่เต็มในการเก็บข้อมูล
- สามารถตรวจสอบพื้นที่ว่างในการเก็บข้อมูล

2. นำคลาสที่สร้างขึ้นไปทดสอบการใช้งานในฟังก์ชัน main โดยทำการสร้างเป็นลักษณะเมนูสำหรับทดลองทุกความสามารถที่มีในคลาส StackArray

Class ของแบบฝึกหัดที่ 1



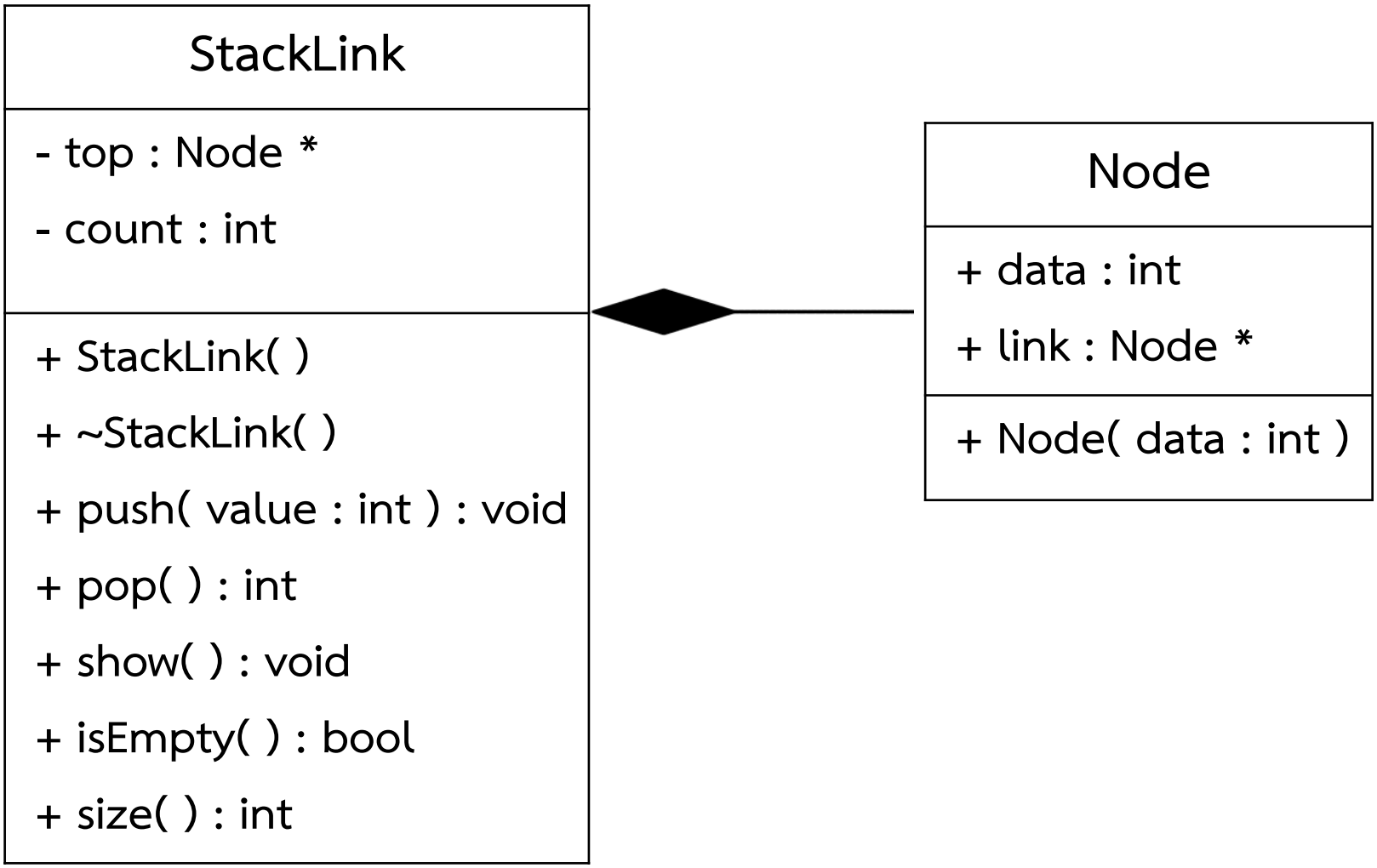
แบบฝึกหัดที่ 2

1. สร้างคลาสสแตกด้วยพอยเตอร์ชื่อ StackLink เพื่อจัดเก็บข้อมูลเลขจำนวนเต็ม โดยมีความสามารถในการจัดการข้อมูล ดังนี้

- สามารถนำข้อมูลเข้าสแตก
- สามารถนำข้อมูลออกจากสแตก
- สามารถแสดงข้อมูลทั้งหมด
- สามารถตรวจสอบพื้นที่ว่างในการเก็บข้อมูล
- สามารถตรวจสอบจำนวนข้อมูลที่มีทั้งหมด

2. นำคลาสที่สร้างขึ้นไปทดสอบการใช้งานในฟังก์ชัน main โดยทำการสร้างเป็นลักษณะเมนูสำหรับทดลองทุกความสามารถที่มีในคลาส StackLink

Class ของแบบฝึกหัดที่ 2



บทที่ 7 Stack

บทเรียนย่อย

7.6 Stack Apply (infix to postfix)

การประยุกต์ใช้สแตกสำหรับการเปลี่ยนนิพจน์แบบ infix ให้เป็นนิพจน์แบบ Postfix โดยที่

นิพจน์ Infix คือ นิพจน์คณิตศาสตร์ที่เครื่องหมายคำนวณอยู่ระหว่างตัวถูกดำเนินการ (ตัวเลขหรือตัวแปร) เช่น

- 5 + 6 * 2 – 3
- A + B * C – D
- A ^ 5 / 2 + B * 8

7.6 Stack Apply (infix to postfix) [2]

นิพจน์ Postfix คือ นิพจน์คณิตศาสตร์ที่เครื่องหมายคำนวณอยู่หลังตัวถูกดำเนินการ (ตัวเลขหรือตัวแปร) เช่น

- 5 6 2 * + 3 -
- A B C * + D -
- A 5 ^ 2 / B 8 * +

ซึ่งโดยปกติคนเรามีความคุ้นเคยกับนิพจน์แบบ Infix ส่วนในระบบคอมพิวเตอร์นั้นจะคำนวณโดยใช้นิพจน์แบบ Postfix จึงจำเป็นต้องเปลี่ยนนิพจน์แบบ Infix ให้เป็นแบบ Postfix

ลำดับความสำคัญในการประมวลผลของเครื่องหมายคำนวณต่าง ๆ

เครื่องหมาย	ค่าเมื่อตอนอ่านเข้ามา	ค่าเมื่ออยู่ในสแตก
+ , -	1	2
* , /	3	4
^	6	5
(	7	0
)	0	ไม่นำเข้าสแตก

หมายเหตุ : เครื่องหมายที่ตัวเลขมาก จะมีสำคัญมากกว่าเครื่องหมายที่มีเลขน้อยกว่า

ขั้นตอนการเปลี่ยนนิพจน์แบบ infix ให้เป็นนิพจน์แบบ Postfix

1. ถ้าค่าที่อ่านเข้ามาเป็นตัวถูกดำเนินการ (ตัวเลขหรือตัวแปร) ให้เขียนเป็นผลลัพธ์ได้เลย

2. ถ้าค่าที่อ่านเข้ามาเป็นเครื่องหมาย ให้พิจารณาดังนี้
   - ถ้าสแตกว่างให้เครื่อง push เครื่องหมายนั้นลงสแตก
   - ถ้าสแตกไม่ว่างให้เปรียบเทียบค่าของเครื่องหมายที่อยู่ตัวบนสุดในสแตกกับค่าของเครื่องหมายใหม่
     - หากเครื่องหมายเก่าในสแตกมีค่าน้อยกว่าเครื่องหมายตัวใหม่ให้ทำการ push ลงสแตก
     - หากเครื่องหมายเก่าในสแตกมีค่ามากกว่า ให้ทำการ pop เครื่องหมายออกมาเป็นผลลัพธ์ และเปรียบเทียบจนกว่าจะมีค่าน้อยกว่า หรือจนกว่าสแตกว่างแล้วจึง push เครื่องหมายใหม่ลงสแตก

ขั้นตอนการเปลี่ยนนิพจน์แบบ infix ให้เป็นนิพจน์แบบ Postfix [2]

- ถ้าเป็นเครื่องหมาย “)” ให้ pop เครื่องหมายออกจากสแตกมาพิมพ์เป็นผลลัพธ์ทีละตัว จนกว่าจะพบเครื่องหมาย “(” แล้วให้ตัดเครื่องหมายทิ้งไป

3. เมื่ออ่านค่าเข้าจนหมดนิพนจ์ของ Infix แล้วให้ pop เครื่องหมายที่เหลือในสแตกออกมาพิพ์เป็นผลลัพธ์ให้หมด ซึ่งจะได้นิพจน์แบบ Postfix

ตัวอย่างการเปลี่ยนนิพจน์แบบ infix ให้เป็นนิพจน์แบบ Postfix

กำหนดให้นิพจน์ Infix : 5 + 6 * 2 – 3

ค่าที่อ่านเข้ามา	ค่าในสแตก	ผลลัพธ์นิพจน์แบบ Postfix
5		5
+	+	5
6	+	5 6
*	+ *	5 6
2	+ *	5 6 2
-	-	5 6 2 * +
3	-	5 6 2 * + 3
		5 6 2 * + 3 -