Stack

ผู้จัดทำ

นาย ธนดล สมชะนะ รหัส 055950201065-0 ปวค. 59 / 2

นาย รวิชา พนิชย์สกุล รหัส 055950201072-6 ปวค. 59 / 2

Stack

สแตกเป็นโครงสร้างข้อมูลแบบลิเนียร์ลิสต์(linear list) ที่สามารถนำข้อมูลเข้าหรือออกได้ทางเดียวคือส่วนบนของสแตก สมาชิกที่เข้าลิสต์ที่หลังสุดจะได้ออกจากลิสต์ก่อน หรือ เข้าหลังออกก่อน

ส่วนประกอบของสแตก

1. ตัวชี้สแตก (Stack Pointer) ซึ่งมีหน้าที่ชี้ไปยังข้อมูลที่อยู่บนสุดของ สแตก (Top stack)
2. สมาชิกของสแตก (Stack Element) เป็นข้อมูลที่จะเก็บลงไปในสแตก ซึ่งจะต้องเป็นข้อมูลชนิดเดียวกัน เช่น ข้อมูลชนิดจำนวนเต็ม เป็นต้น

นอกจากนี้ยังต้องมีตัวกำหนดค่าสูงสุดของสแตก (Max Stack) ซึ่งจะเป็นตัวบอกว่าสแตกนี้สามารถเก็บ จำนวนข้อมูลได้มากที่สุดเท่าไร

กระบวนการของสแตก

สแตกจะประกอบด้วยกระบวนการ 3กระบวนการที่สำคัญ คือ

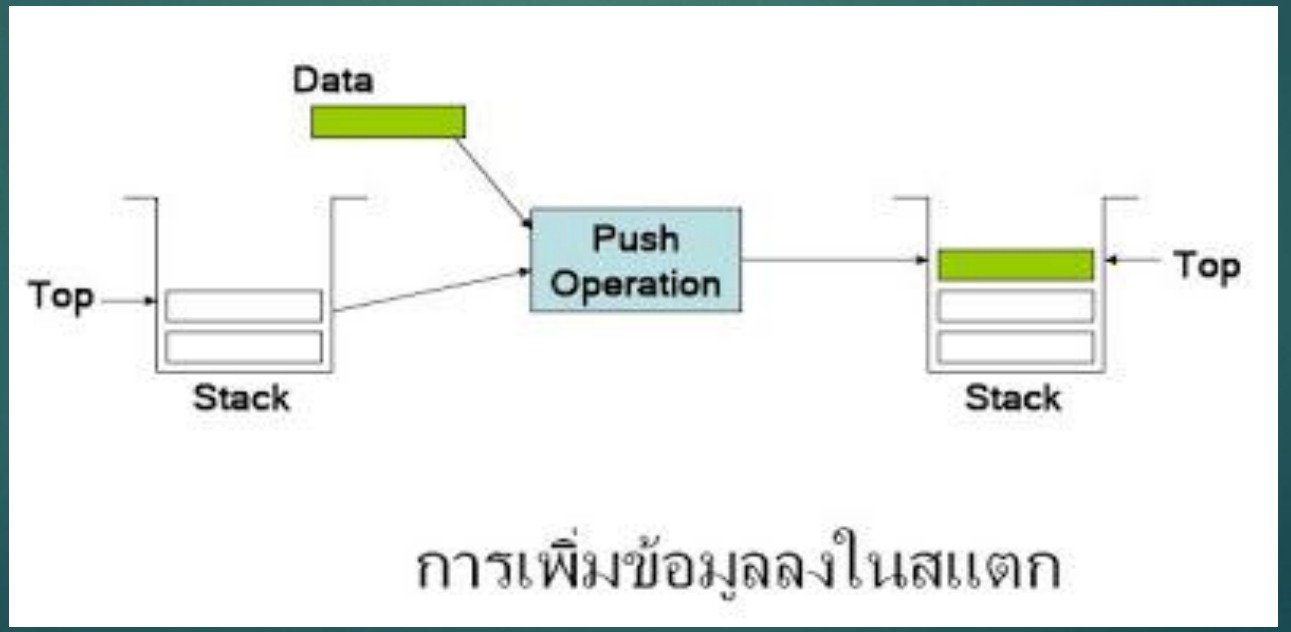
1. Push คือ การนำข้อมูลใส่ลงไปในสแตก

2. Pop คือการนำข้อมูลออกจากส่วนบนสุดของสแตก

3. Stack Top เป็นการคัดลอกข้อมูลที่อยู่บนสุดของสแตก

Push

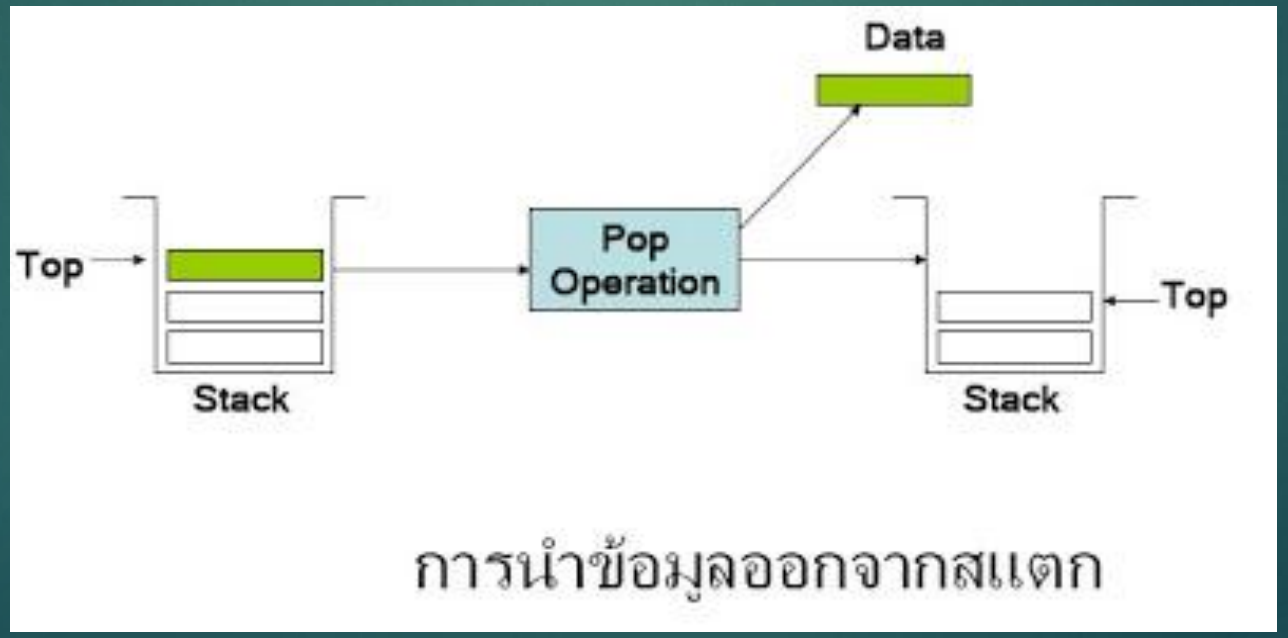
คือ การนำข้อมูลใส่ลงไปในสแตกเช่น สแตก s ต้องการใส่ข้อมูล iในสแตกจะได้ push(s,i) คือ ใส่ข้อมูล i ลงไปที่ท็อปของสแตก sในการเพิ่มข้อมูลลงในสแตก



การเพิ่มข้อมูลลงในสแตก

Pop

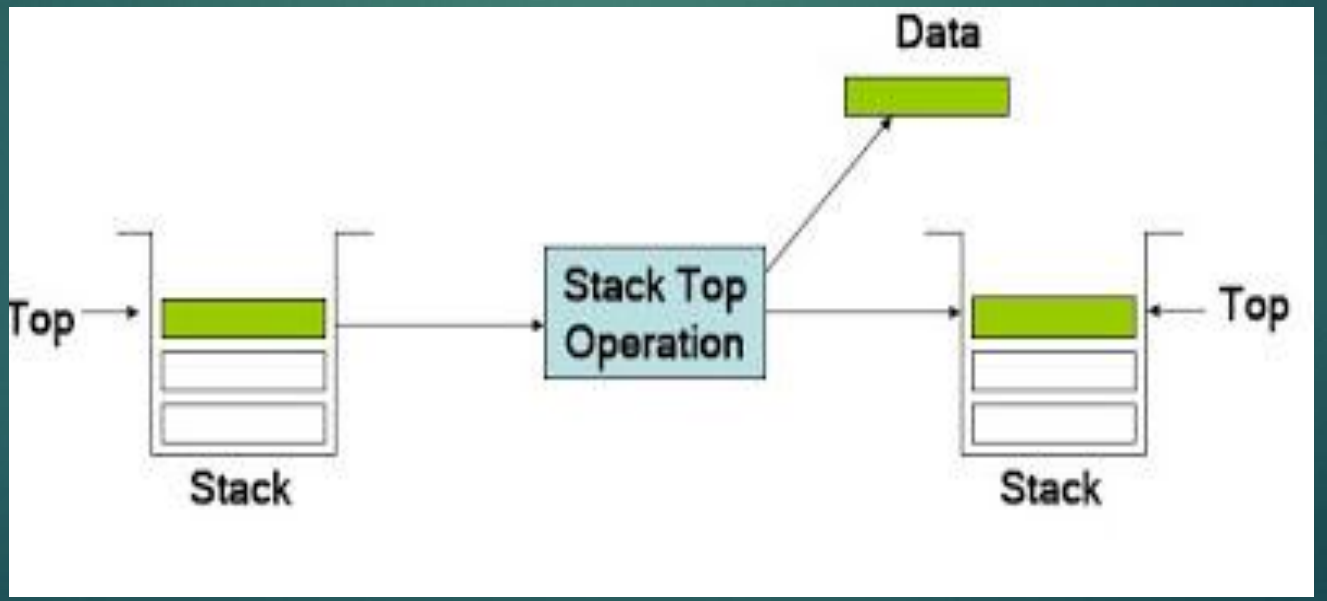
Pop คือการนำข้อมูลออกจากส่วนบนสุดของสแตกเช่น ต้องการนำข้อมูลออกจากสแตกsไปไว้ที่ตัวแปร iจะได้ i = pop (s)



การนำข้อมูลออกจากสแตก

Stack Top

เป็นการคัดลอกข้อมูลที่อยู่บนสุดของสแตกแต่ไม่ได้นำเอาข้อมูลนั้นออกจากสแตก



ปัญหาที่เกิดขึ้นกับสแตก

สแตกเต็ม (Full Stack)

สแตกว่าง (Empty Stack)

สแตกเต็ม (Full Stack)

การ push สแตกทุกครั้งจะมีการตรวจสอบที่ว่างในสแตกว่ามีที่ว่างเหลือหรือไม่ ถ้าไม่มีที่ว่างเหลืออยู่ เราก็จะไม่สามารถทำการ push สแตกได้ ในกรณีเช่นนี้เราเรียกว่าเกิดสถานะล้นเต็ม (Stack Overflow) โดย การตรวจสอบว่าสแตกเต็มหรือไม่ เราจะใช้ตัวชี้สแตก (Stack pointer) มาเปรียบเทียบกับค่าสูงสุดของ สแตก (Max stack) หากตัวชี้สแตกมีค่าเท่ากับค่าสูงสุดของสแตกแล้ว แสดงว่าไม่มีที่ว่างให้ข้อเก็บข้อมูล อีก

สแตกว่าง (Empty Stack)

การ pop สแตกทุกครั้งจะมีการตรวจสอบข้อมูลในสแตกว่ามีข้อมูลในสแตกหรือไม่ ถ้าไม่มีข้อมูลในสแตก เหลืออยู่ เราก็ไม่สามารถทำการ pop สแตกได้ ในกรณีเช่นนี้เรียกว่าเกิดสถานะสแตกจม (Stack Underflow) โดยการตรวจสอบว่าสแตกว่างหรือไม่ เราจะตรวจสอบตัวชี้สแตกว่าเท่ากับ 0 หรือ null หรือไม่ ถ้าเท่ากับ 0 แสดงว่า สแตกว่าง จึงไม่สามารถดึงข้อมูลออกจากสแตกได้

การประยุกต์ใช้สแตก

ตัวอย่างงานที่นิยมใช้กับโครงสร้างข้อมูลแบบสแตก

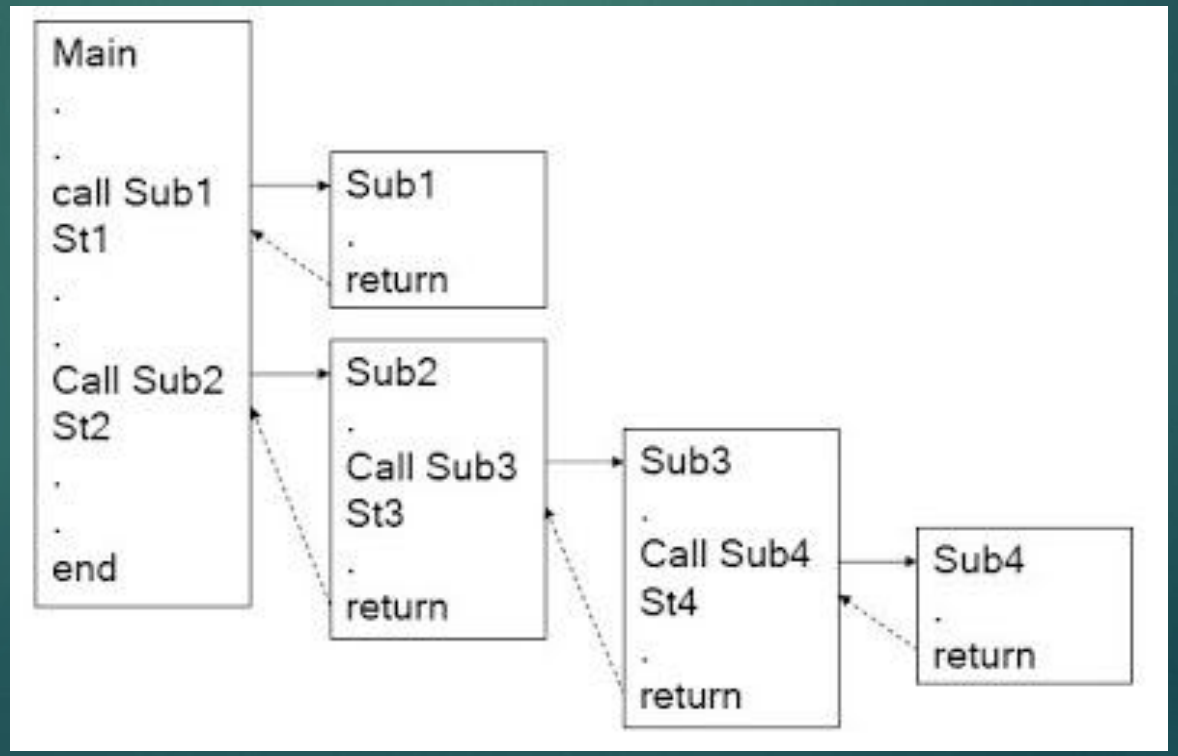
1.ใช้ในการจดจำการกระโดดไปมาระหว่างโปรแกรมย่อย

2.ใช้ในการเขียนโปรแกรมแบบรีเคอร์ซีฟ (Recursive)

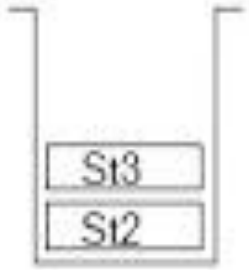
3.ใช้ในการจดจำเส้นทางในการเดินทางของโครงสร้างข่ายงาน (Network) หรือโครงสร้างของต้นไม้ (Tree)

4.การเปรียบเทียบรูปแบบ (Parenthesis Matching)

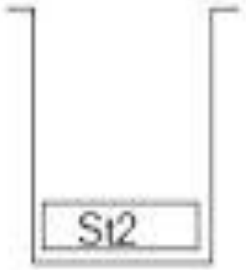
โดยมากโครงสร้างข้อมูลของสแตกจะใช้กันในเรื่องของการทำอินเตอร์รัพต์หรือการเรียกใช้โปรแกรมย่อย เนื่องจากการทำงานเช่นนี้ ต้องมีการเก็บตำแหน่งการทำงานเดิมเอาไว้ก่อนที่จะกระโดดไปทำงานโปรแกรมย่อยอื่น และหากต้องการกลับมาทำงานในโปรแกรมย่อยเดิมได้อย่างเป็นลำดับ



การเรียกใช้โปรแกรมย่อย



7. กลับมาทำงานใน Sub3



8. กลับมาทำงานใน Sub2

9. กลับมาทำงานใน Main สแตกว่างเปล่า

การคำนวณนิพจน์ทางคณิตศาสตร์

ในการเขียนโปรแกรมคอมพิวเตอร์ด้วยภาษาระดับสูงคำสูงที่เป็นนิพจน์ทางคณิตศาสตร์จะเขียนอยู่ในรูปแบบของนิพจน์ Infix การคำนวณคำนิพจน์ในรูปแบบนี้ ตัวแปลภาษาต้อง ทำการ ตรวจสอบนิพจน์จากซ้ายไปขวา เพื่อหาเครื่องหมาย ที่ต้องคำนวณก่อนจงจะเริ่มคำนวณให้ แล้วทำแบบนี้ซ้ำ ๆกันจนกว่าจะ คำนวณเครื่องหมายครบทุกตัวทำให้การทำงานช้าและไม่สะดวก ต่อการคำนวณการแก้ปัญหานี้ ตัวแปลภาษาจะทำงานแปลงนิพจน์ Infix ให้ อยู่ในรูปแบบที่ช่วยให้การคำนวณสะดวกและรวดเร็วขึ้น โดยแปลงให้อยู่ในรูปของนิพจน์ Postfix

โดยทั่วไปนิพจน์ทางคณิตศาสตร์สามารถเขียนได้ 3 รูปแบบ คือ

1. นิพจน์ Infix นิพจน์รูปแบบนี้ operator จะอยู่ตรงกลางระหว่างตัวถูกดำเนินการ 2 ตัว

2. นิพจน์ Postfix นิพจน์รูปแบบนี้ จะต้องเขียนตัวถูกดำเนินการตัวที่ 1 และ 2 ก่อน แล้วตามด้วย operator

3. นิพจน์ Prefix นิพจน์รูปแบบนี้ จะต้องเขียน operatorก่อนแล้วตามด้วยตัวถูกดำเนินการตัวที่ 1 และ 2

ลำดับความสำคัญ Operator

1. เครื่องหมายยกกำลัง ^

2. เครื่องหมายคูณกับหาร *,/

3. เครื่องหมายบวกกับลบ +,-

4. เครื่องหมายวงเล็บ () กระทำก่อนเช่น A+(B*C)

5. เครื่องหมายระดับเดียวกันไม่มีวงเล็บให้ทำจากซ้ายไปขวา เช่น A+B+C

นิพจน์ A*(B+C-D)/E

ตัวที่อ่านเข้ามา	ผลลัพธ์ในสแตก	นิพจน์ Postfix
A	ว่าง	A
*	*	A
(	*(	A
B	*(	AB
+	*(+	AB
C	*(+	ABC
-	*(-	ABC+
D	*(-	ABC+D
)	*	ABC+D-
/	/	ABC+D-*
E	/	ABC+D-*E
		ABC+D-*E/

Thank you!