C++
STANDARD TEMPLATE LIBRARY (STL)
(STACK)

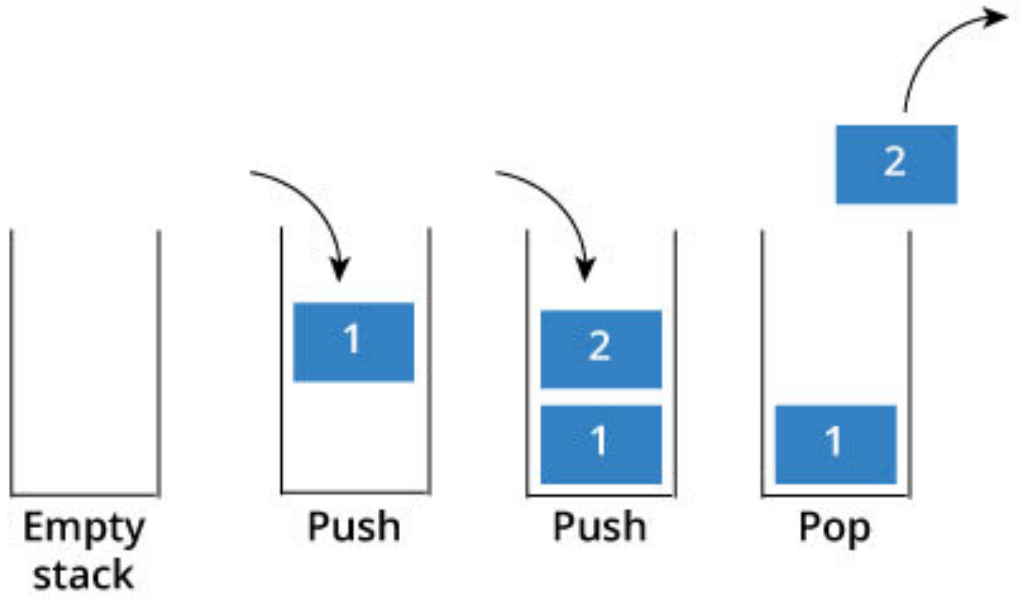
WASARA RODHETBHAI

DEPARTMENT OF COMPUTING • SILPAKORN UNIVERSITY

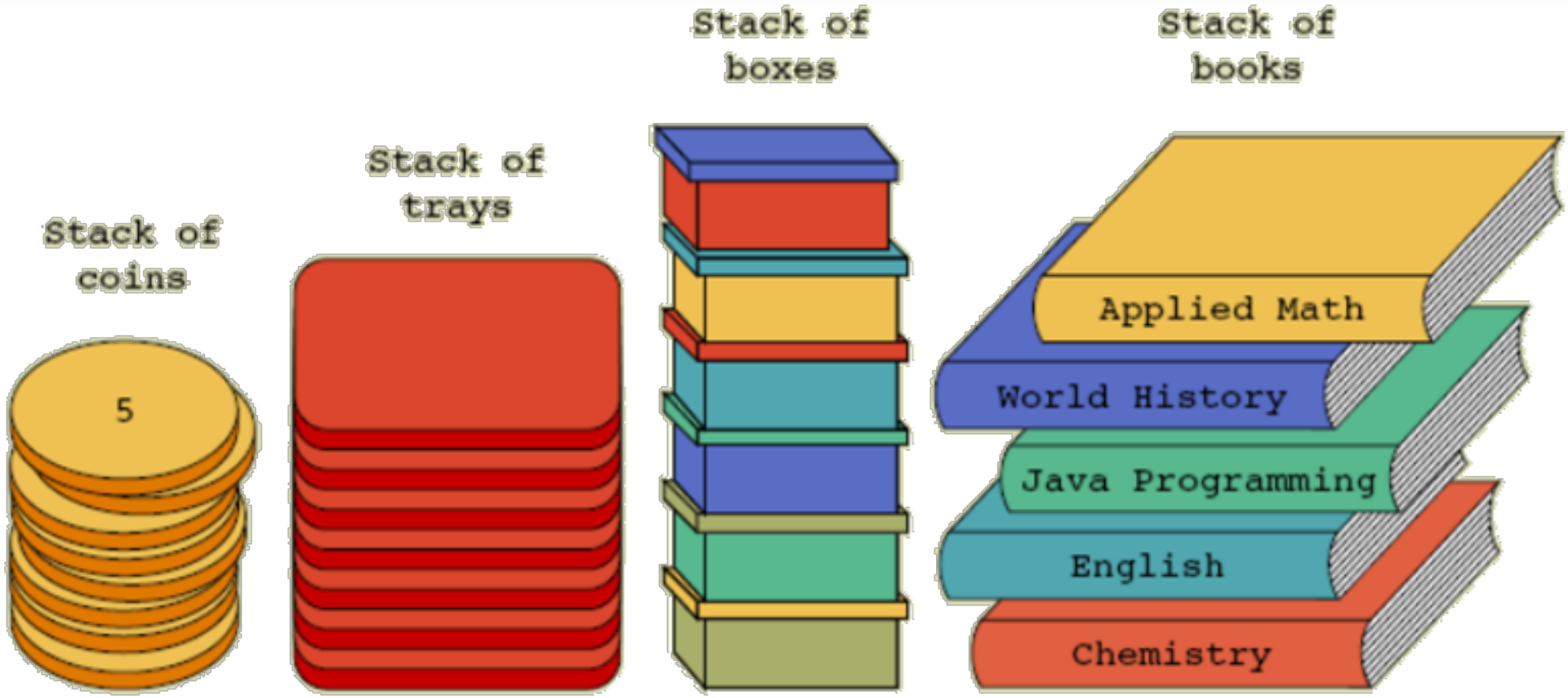
03/2024

Stack (กองซ้อน)

- โครงสร้างข้อมูลเชิงเส้นที่มีการจัดเก็บข้อมูลในรูปแบบของ Last In First Out (LIFO)



Stack ในชีวิตประจำวัน



Stack ในระบบคอมพิวเตอร์

- Web Browser
- Undo
- Call Function

การสร้าง Stack Container

```
#include <stack>
```

รูปแบบ

```
stack<data_type> stack_name;
```

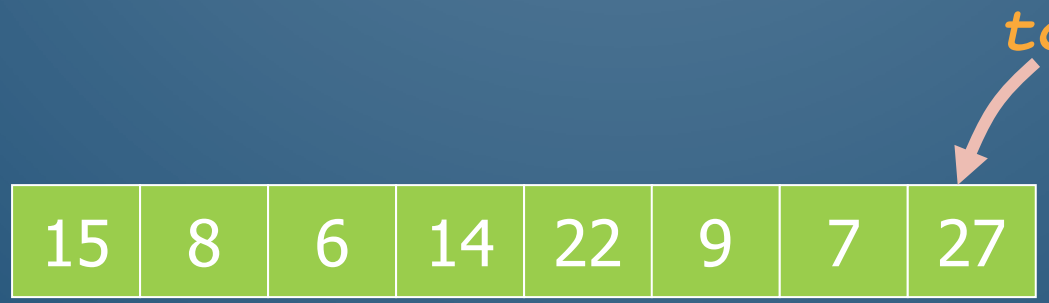
ตัวอย่าง

```
stack<double> scores;
stack<string> names;
stack<int> ages;
```

การสร้าง Stack Container พร้อมค่าเริ่มต้น

ตัวอย่าง

```
stack<int> task({15,8,6,14,22,9,7,27});
```



การสร้าง Stack Container ของ Record (Struct)

ตัวอย่าง

```
struct Student {
        int no;
        string name;
        double score;
};
stack<Student> classroom;
```

การสร้าง Stack Container ของ Object

ตัวอย่าง

```
class Person {
    public:
        string name;
        int age;
        double income;
}
stack<Person> people;
```

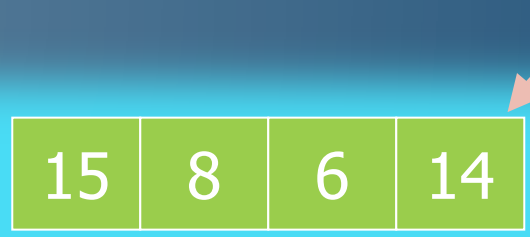
Method ของ Stack Container

- เมธอดที่สำคัญของ Stack Container เช่น

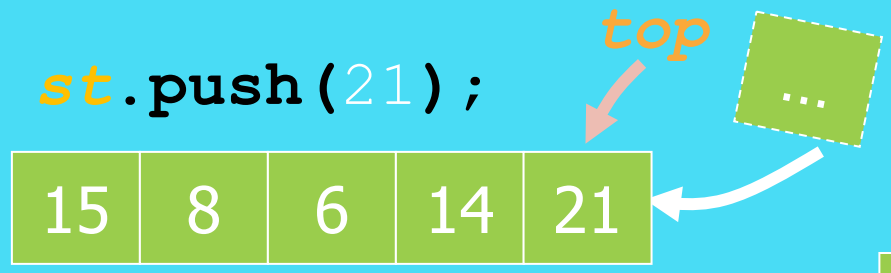
```
push();          // นำข้อมูลเพิ่มเข้าในตำแหน่งบนสุดของ stack
pop();           // ลบข้อมูลในตำแหน่งบนสุดออกจาก stack
top();           // คืนค่าข้อมูลในตำแหน่งบนสุดของ stack
size();          // คืนค่าจำนวนข้อมูลใน stack
empty();         // ตรวจสอบว่า stack ว่างหรือไม่
swap();          // สับเปลี่ยนข้อมูลที่อยู่ใน stack ทั้งสอง
```

การนำข้อมูลเข้าและออกใน Stack Container

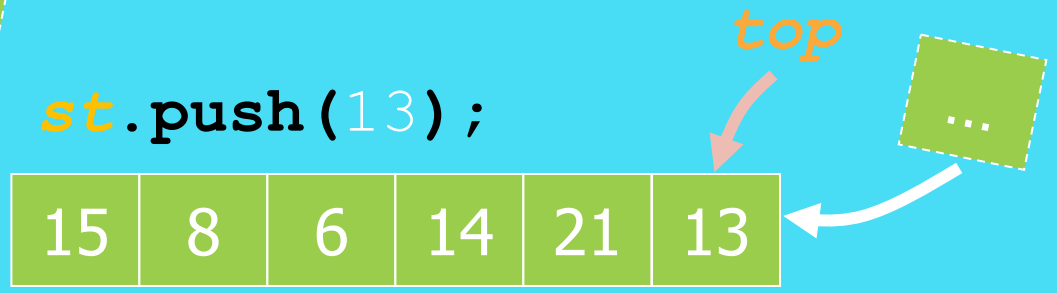
```
stack<int> st({15,8,6,14});
```



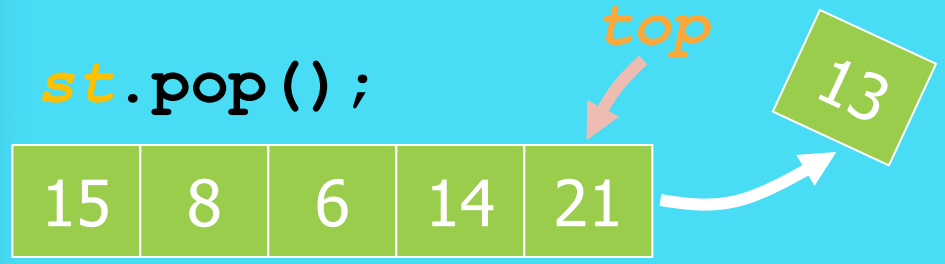

```
st.push(21);
```



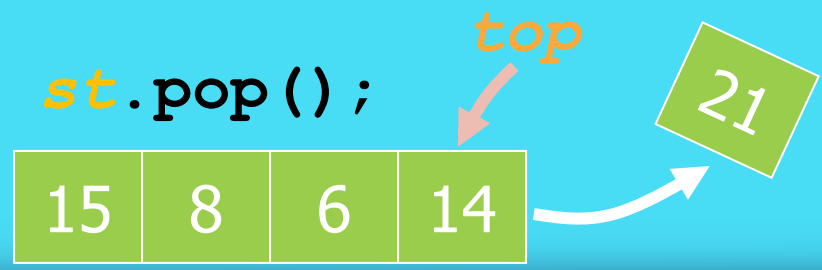

```
st.push(13);
```




```
st.pop();
```




```
st.pop();
```



ข้อมูลใน Stack Container ณ ตำแหน่งบนสุด

ตัวอย่าง

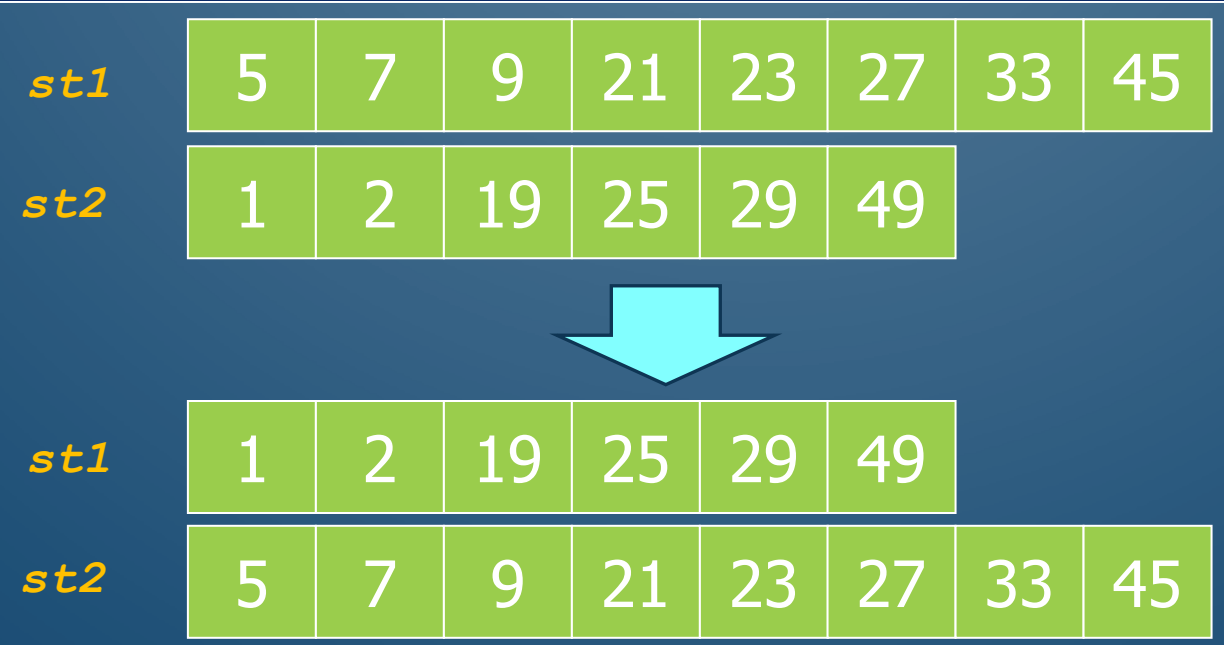
```
stack<int> st({5,7,9,21,23,27,33,45});

int val=25;
while(!st.empty() && st.top()>val) {
        cout << st.top() << " ";
        st.pop();
}
```

การสับเปลี่ยนข้อมูลระหว่าง Stack Container

ตัวอย่าง

```
stack<int> st1{5,7,9,21,23,27,33,45};
stack<int> st2{1,2,19,25,29,49};
st1.swap(st2);
```



การแปลงเลขฐานสิบ (decimal) เป็นเลขฐานสอง (binary)

ขั้นตอนวิธี

อ่านค่าตัวเลข n

การทำวนซ้ำในขณะที่ n มีค่ามากกว่า 0

ทำการหาเศษเหลือจากการหารค่า n ด้วย 2

บันทึกค่าเศษเหลือดังกล่าวเอาไว้

หารค่า n ด้วย 2 (ปัดเศษเหลือทิ้ง)

เลขฐานสองคือค่าเศษเหลือทั้งหมดทุกค่าเรียงลำดับจากค่าสุดท้ายย้อนกลับไปจนถึงค่าแรก

n	ตัวหาร	เศษเหลือ
25	2	1
12	2	0
6	2	0
3	2	1
1	2	1
0		
คำตอบ	25 ฐาน 10 เท่ากับ **11001** ฐาน 2	

การตรวจสอบการจับคู่เครื่องหมาย (Balancing Symbols)

- ตรวจสอบ Syntax Error ในการเขียนภาษาโปรแกรม
- ตรวจสอบสูตรคณิตศาสตร์

(54+(2-73+(3*6)))	ผ่าน
((43+90/2-90))+48)	ไม่ผ่าน
54+[43+(9*22)/9]*(6-[8*2])	ผ่าน
90*({7-(4/2)*[8-2]}	ไม่ผ่าน
19+(9*{[17-2)/2-15}]	ไม่ผ่าน

การตรวจสอบการจับคู่เครื่องหมาย (Balancing Symbols)

```
สร้าง Stack (ว่าง)
while (ยังมีข้อมูลในข้อความเหลืออยู่) {
        อ่านข้อมูลในข้อความลำดับถัดไปมา 1 ค่า
        if (ข้อมูลนั้นเป็นสัญลักษณ์เปิด)
                นำข้อมูลนั้นใส่ลงใน Stack
        otherwise if (ข้อมูลนั้นเป็นสัญลักษณ์ปิด) {
                if (Stack ว่าง) {
                        (ยุติ error) สรุปได้ว่าข้อความนี้มีเครื่องหมายที่ไม่สมดุล
                otherwise {
                        นำสัญลักษณ์เปิดออกจาก Stack
                        if (สัญลักษณ์เปิดที่นำออกมาจาก Stack ไม่ใช่คู่กับสัญลักษณ์ปิดที่อ่านเข้ามา)
                                (ยุติ error) สรุปได้ว่าข้อความนี้มีเครื่องหมายที่ไม่สมดุล
                }
        }
}
if (Stack ว่าง)
        สรุปได้ว่าข้อความนี้มีเครื่องหมายที่สมดุล
otherwise
        (ยุติ error) สรุปได้ว่าข้อความนี้มีเครื่องหมายที่ไม่สมดุล
```

การแปลงนิพจน์ Infix ให้เป็น Postfix

```
สร้าง Stack (ว่าง)
while (ยังมีข้อมูลในข้อความ Infix เหลืออยู่) {
        อ่านข้อมูลในข้อความ Infix ลำดับถัดไปมา 1 ค่า
        if (ข้อมูลนั้นเป็น Operand)
                แสดงค่าข้อมูลนั้นออกไป
        if (ข้อมูลนั้นเป็นเครื่องหมายสัญลักษณ์เปิด)
                นำข้อมูลที่เป็นเครื่องหมายสัญลักษณ์เปิดนั้นใส่ลงใน Stack
        if (ข้อมูลนั้นเป็นเครื่องหมายสัญลักษณ์ปิด) {
                while (Stack ไม่ว่าง และ
                        ข้อมูลใน Stack ที่อยู่ข้างบนสุดไม่ใช่เครื่องหมายสัญลักษณ์เปิด)
                        นำข้อมูลใน Stack ออกมาและแสดงค่าข้อมูลนั้นออกไป
                นำข้อมูลใน Stack ที่เป็นเครื่องหมายสัญลักษณ์เปิดออกมาและละทิ้งไป
        }
        if (ข้อมูลนั้นเป็น Operator) {
                if (Stack ว่าง หรือ ข้อมูลใน Stack ที่อยู่ข้างบนสุดเป็นเครื่องหมายสัญลักษณ์เปิด)
                        นำข้อมูลที่เป็น operator นั้นใส่ลงใน Stack
                otherwise {
                        while (Stack ไม่ว่าง และ
                                ข้อมูลใน Stack ที่อยู่ข้างบนสุดไม่ใช่เครื่องหมายสัญลักษณ์เปิด และ
                                Operator ที่อ่านเข้ามานั้นมีความสำคัญน้อยกว่าหรือเท่ากับ Operator ที่อยู่ด้านบนของ Stack)
                                    นำ Operator ออกมาจาก Stack และแสดงค่าออกไป
                        นำ Operator ตัวล่าสุดใส่ลงใน Stack
                }
        }
}
while (Stack ไม่ว่าง)
        นำ Operator ออกมาจาก Stack และแสดงค่าออกไป
```

การประมวลผลนิพจน์ Postfix

```
สร้าง Stack (ว่าง)
while (ยังมีข้อมูลในข้อความ Postfix เหลืออยู่){
        อ่านข้อมูลในข้อความ Postfix ลำดับถัดไปมา 1 ค่า
        if (ข้อมูลเป็น Value)
                นำค่า Value ดังกล่าวใส่ลงใน Stack
        if (ข้อมูลเป็น Operator) {
                นำค่า Value ออกมาจาก Stack จำนวนตามเท่าที่ Operator ต้องการ
                if (ไม่เกิดปัญหาข้อมูลในการนำค่าออกมาจาก Stack ข้างต้น){
                        ทำการประมวลผล Operator กับ Operand ซึ่งเป็นค่า Value ที่ได้นำออกมาจาก Stack ข้างต้น
                        นำผลลัพธ์ของการประมวลผลคำสั่งที่คำนวณได้ใส่ลงใน Stack
                } otherwise
                        (ยุติ error) ไม่สามารถประมวลผลได้สำเร็จ
        }
}
if (มีข้อมูลใน Stack เหลือแค่เพียงหนึ่งค่าเท่านั้น)
        ค่าที่อยู่ใน Stack นั้นเป็นผลลัพธ์ที่ได้จากการประมวลผล
otherwise
        (ยุติ error) ไม่สามารถประมวลผลได้สำเร็จ
```