Practice Session 1
ฝึกปฏิบัติแก้ปัญหาโจทย์ 1

การอบรมคอมพิวเตอร์โอลิมปิก สอวน. ค่ายที่ 2 ปีการศึกษา 2563

อ.ดร.ฐาปนา บุญชู

สาขาวิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์และเทคโนโลยี

มหาวิทยาลัยธรรมศาสตร์

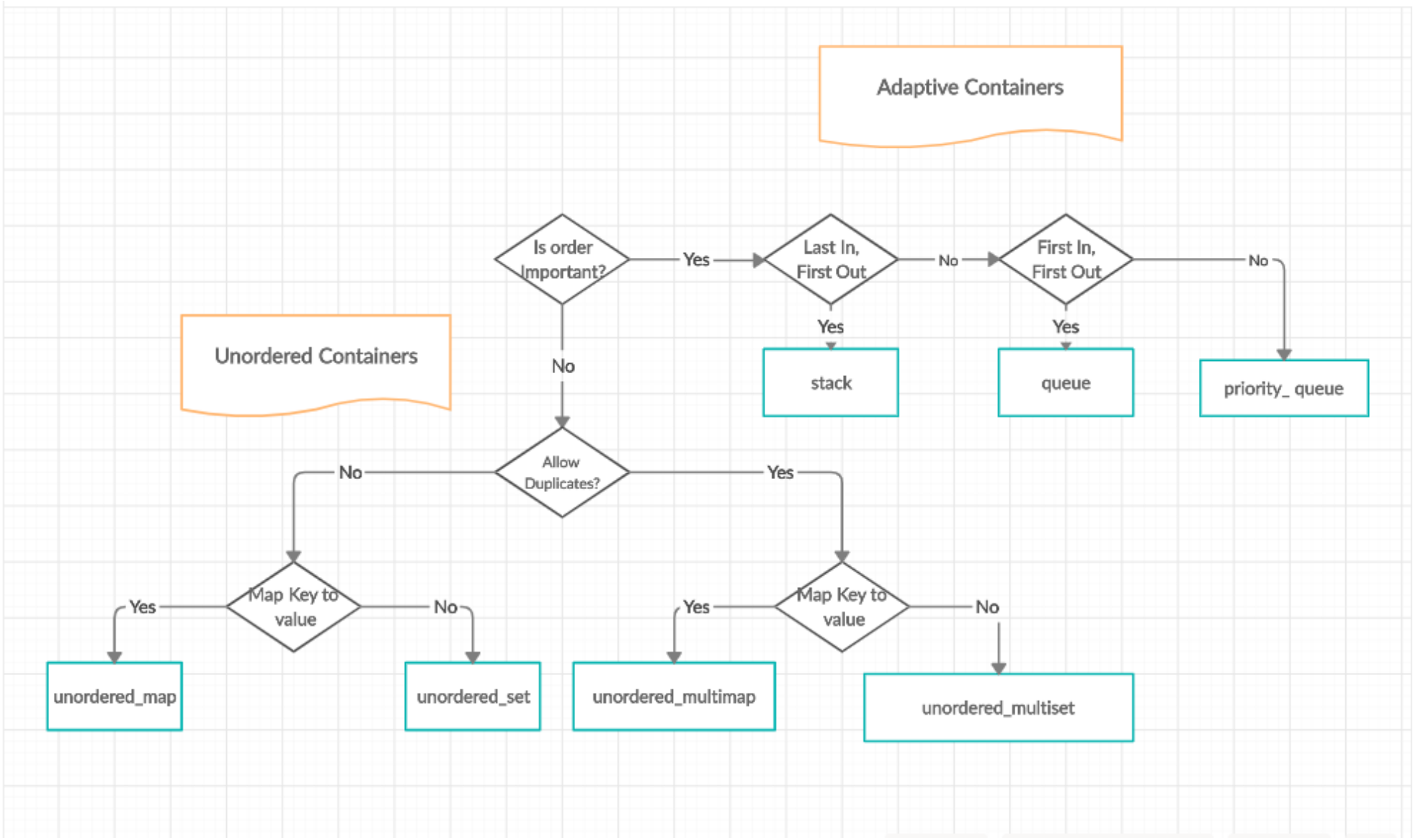
C++ Standard Template Library (STL)

- Containers
- Algorithms
- Iterators
- Functions
- Pairs
- Others

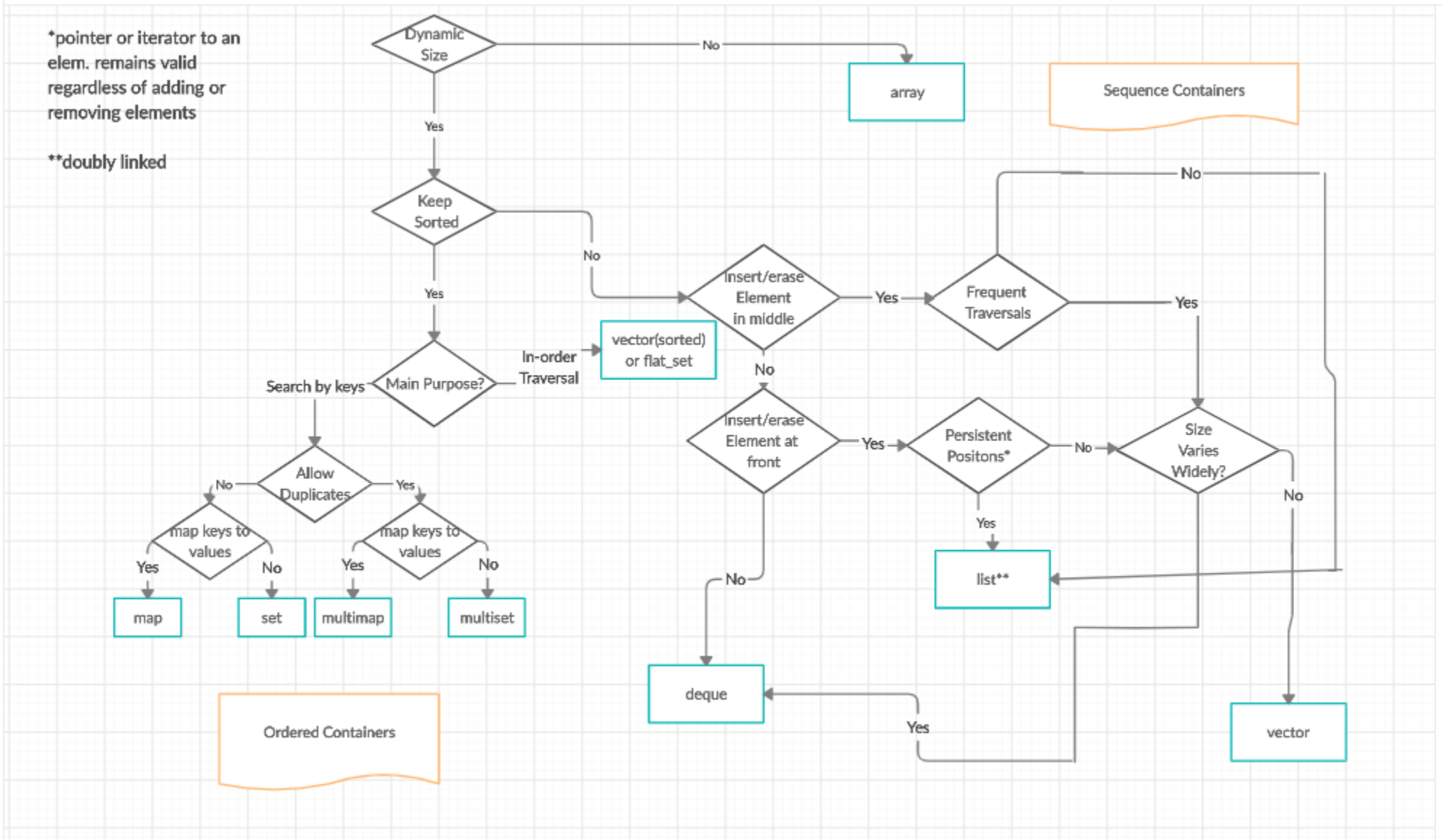
Containers

- Containers คือคลาสมาตรฐานที่เอาไว้สำหรับเก็บข้อมูลต่างๆ ตัวอย่าง Containers เช่น
  - Sequence Containers
    - Vector
    - List
  - Adaptors
    - Queue
    - Priority Queue
    - Stack
  - Associative Containers (เป็นโครงสร้างข้อมูลที่มีการเรียงกันของข้อมูล (Sorted) โดยสามารถทำการ Search ได้ใน $O(\log n)$)
    - Set
    - Multiset
    - Map
  - Unordered Associative Containers
    - Unordered Set
    - Unordered Map

Flowchart of Adaptive Containers and Unordered Containers



Flowchart of Sequence conatiners and ordered containers



Iterators

- ใช้สำหรับเป็น Pointer ชี้ไปยัง Memory Addresses ของ STL Containers ต่างๆ
- Operations:
  - `begin()` คือฟังก์ชันที่จะคืนค่าตำแหน่ง**เริ่มต้น**ของ Container
  - `end()` คือฟังก์ชันที่จะคืนค่าตำแหน่ง**หลัง**สุดท้ายของ Container

Pairs

- เป็น Container ที่ใช้แทนคู่อันดับ (E1, E2) โดยที่ E1 และ E2 คือชนิดข้อมูล
- make_pair() คืนค่า pair ที่ถูก initialized ด้วยค่าที่ส่งเป็นพารามิเตอร์
- References:
  - first อ้างถึงสมาชิกตัวแรก (E1)
  - Second อ้างถึงสมาชิกตัวที่สอง (E2)

Vector

- Vector เป็น Dynamic array โดยที่ Size ของมันสามารถปรับเปลี่ยนไปได้เมื่อมีการ เพิ่ม ลบ โดยที่เราไม่จำเป็นต้องกังวลถึงการจองพื้นที่
- สามารถเข้าถึงได้ด้วย [] เหมือน array ทั่วไป (Constant time)
- Operations:
  - push_back() เพิ่มสมาชิกใหม่ โดยจะถูกเพิ่มไปท้ายสุดของ Array (Constant time)
  - pop_back() ลบสมาชิกตัวหลังสุด (Constant time)
  - size() รีเทิร์นค่าความยาวของ array ปัจจุบัน (Constant time)
  - erase() ลบสมาชิกตำแหน่งที่ระบุไว้ในพารามิเตอร์ (Linear time)
  - front(), back() คืนค่าสมาชิกตัวแรก (front) และตัวสุดท้าย (end)
  - อื่นๆ
- Vector มีลักษณะการจองพื้นที่แบบติดกัน (Contiguous memory) ดังนั้นการเข้าถึงสมาชิกจะสามารถทำได้รวดเร็วในขณะที่ Container ที่มีลักษณะคล้ายกันคือ List มีลักษณะการจองพื้นที่แบบไม่ติดกัน (Non-contiguous memory) เวลาที่เข้าถึงสมาชิก vector จะทำได้รวดเร็วกว่า แต่ถ้าต้องการลบหรือเพิ่มสมาชิกอาจทำได้เร็วกว่า

ตัวอย่าง

- ให้ดูในไฟล์ my_vector.cpp

แบบฝึกหัด 1

ให้นักเรียนเขียนโปรแกรมเพื่อแก้ปัญหาต่อไปนี้

- กำหนดให้ข้อมูลเข้าเป็น $a_1, a_2, \ldots, a_n$ โดยที่ $n \geq 2$ เสมอ และ a_i ใดๆ เป็นเลขโดด
- ให้แสกนลิสท์จากตำแหน่งซ้ายสุดไปทางขวา โดยให้ตรวจสอบว่าค่าที่กำลังเช็คดังกล่าวมีค่าเท่ากับค่าในตำแหน่งสุดท้ายในลิสท์หรือไม่ ถ้าไม่ใช่ให้แสกนต่อ ถ้าใช่ให้ลบตำแหน่งสุดท้ายออกและตรวจสอบเลขตัวสุดท้ายลำดับถัดไป ทำการตรวจสอบต่อไปเรื่อยๆจนตำแหน่งที่ตรวจสอบคือตำแหน่งสุดท้ายในลิสท์พอดี และ
- ให้ตอบว่าผลรวมของลิสท์ที่เหลือมีค่าเท่าใด
- ตัวอย่าง ข้อมูลนำเข้า 1 2 3 4 1 2 3 4
  ข้อมูลส่งออก 16

Queue (แถวคอย)

- เป็น Container ที่มีลักษณะการดำเนินการแบบ “เข้าก่อนออกก่อน” หรือ “First In First Out” หรือ FIFO
- มีลักษณะการเพิ่มสมาชิกไว้ที่ท้ายคิวและเอาออกที่หัวคิว
- Operations:
  - push() เพิ่มสมาชิกเข้าไปในคิว
  - pop() ป๊อปสมาชิกหัวแถวออก
  - front() คืนค่าสมาชิกหัวแถว
  - empty() คืนค่าจริงเมื่อคิวเป็นคิวว่าง คืนเท็จเมื่อคิวมีสมาชิกอย่างน้อย 1 ตัว
  - size() คืนค่าจำนวนสมาชิกที่อยู่ในแถวคอย

Stack

- เป็น Container ที่มีลักษณะการดำเนินการแบบ “เข้าก่อนออกทีหลัง” หรือ “Last In First Out” หรือ LIFO
- มีลักษณะการเพิ่มสมาชิกไว้ที่ท้ายคิวและเอาออกที่หัวคิว
- Operations:
  - push() เพิ่มสมาชิกเข้าไปในคิว
  - pop() ป๊อปสมาชิกหัวแถวออก
  - front() คืนค่าสมาชิกหัวแถว
  - empty() คืนค่าจริงเมื่อคิวเป็นคิวว่าง คืนเท็จเมื่อคิวมีสมาชิกอย่างน้อย 1 ตัว
  - size() คืนค่าจำนวนสมาชิกทั้งหมด

แบบฝึกหัด 2

ให้นักเรียนเขียนโปรแกรมเพื่อแก้ปัญหาต่อไปนี้

- กำหนดให้ข้อมูลเข้าเป็น $a_1, a_2, \ldots, a_n$ โดยที่ $n \geq 2$ เสมอ และ a_i ใดๆ เป็นเลขโดด
- ให้แสกนลิสท์จากตำแหน่งซ้ายสุดไปทางขวา โดยให้ตรวจสอบว่าค่าที่กำลังเช็คดังกล่าวมีค่าเท่ากับค่าในตำแหน่งสุดท้ายในลิสท์หรือไม่ ถ้าไม่ใช่ให้แสกนต่อ ถ้าใช่ให้ลบตำแหน่งสุดท้ายออกและนำไปเพิ่มไว้ต้นลิสท์ และตรวจสอบเลขตัวสุดท้ายลำดับถัดไป ทำการตรวจสอบต่อไปเรื่อยๆจนตำแหน่งที่ตรวจสอบคือตำแหน่งสุดท้ายในลิสท์พอดี และ
- ให้ตอบว่าผลรวมตำแหน่งแรกและตำแหน่งสุดท้ายของลิสท์มีค่าเท่ากับเท่าใด
- ตัวอย่าง ข้อมูลนำเข้า 1 2 3 4 1 2 3 4
  ข้อมูลส่งออก 7

Priority Queue

- Priority Queue เป็น Container ที่ถูกออกแบบมาให้หัวแถวจะต้องมีค่ามากที่สุดเสมอ (เทียบกับสมาชิกตัวอื่นในคิว)
- Operators:
  - push() เพิ่มสมาชิกเข้าไปในคิว
  - pop() ป๊อปสมาชิกหัวแถวออก
  - top() คืนค่าสมาชิกหัวแถว
  - empty() คืนค่าจริงเมื่อคิวเป็นคิวว่าง คืนเท็จเมื่อคิวมีสมาชิกอย่างน้อย 1 ตัว
  - size() คืนค่าจำนวนสมาชิกที่อยู่ในแถวคอย
- เราสามารถ customize ตัวเปรียบเทียบ (Comparator) หรือกำหนด priority ของ queue เองได้

ตัวอย่าง

- ให้ดูในไฟล์ queue_stack_pq.cpp

Sets (เซต)

- เซตคือ Container ที่สมาชิกแต่ละตัวต้องมีความแตกต่างกัน (Unique)
- สมาชิกของ Set ไม่สามารถถูกแก้ไขตรงๆได้ อาจแก้ไขได้ด้วยการลบสมาชิกตัวที่ต้องการแก้ไขออกและเพิ่มค่าที่แก้ไขแล้วลงไปใหม่
- Operation:
  - begin() คืนค่า Iterator ชี้ไปยังสมาชิกตัวแรกของเซต
  - end() คืนค่า Iterator ชี้ไปยังสมาชิกตัวหลังตัวสุดท้ายของเซต
  - size() คืนค่าจำนวนสมาชิกที่อยู่ในเซต
  - insert() เพิ่มสมาชิกลงไปใน Set
  - upper_bound() คืนค่า iterator ไปยังสมาชิกตัวแรกที่เท่ากับค่าพามิเตอร์ที่ใส่เข้าไป หรือ สมาชิกตัวที่จะไม่**มาก่อน**ค่าดังกล่าว
  - lower_bound() คืนค่า iterator ไปยังสมาชิกตัวแรกที่เท่ากับค่าพามิเตอร์ที่ใส่เข้าไป หรือ สมาชิกตัวที่จะไม่**มาทีหลัง**ค่าดังกล่าว
- ใช้ Iterator ในการเข้าถึงสมาชิกภายในเซต

ตัวอย่าง

- ให้ดูในไฟล์ sets.cpp

Map

- Map เป็น container ที่มีโครงสร้างแบบ Key แมพไปยัง Value โดย Keys จะต้องไม่ซ้ำกัน
- ใช้ Pair เพื่อแทนคู่อันดับการแมพ (Key, Value)
- Operations ที่สำคัญ:
  - pair_insert() เพิ่มสมาชิกใหม่เป็นคู่อันดับ (Key, Value)
  - erase() ลบสมาชิกโดยใช้ Key หรือ Iterator ในตำแหน่งที่ต้องการลบเป็นพารามิเตอร์
  - clear() ลบสมาชิกทุกตัวใน map
  - find() คืนค่า iterator ของคู่อันดับที่ค่า Key ตรงกับพารามิเตอร์
- สามารถเข้าถึง/เปลี่ยนแปลงค่า Value ด้วยการใช้เครื่องหมาย [ ] เช่น mymap[key]
- unordered_map จะคล้ายกับ map แต่การเก็บค่าไม่ได้เป็นแบบเรียงลำดับ เพราะฉะนั้นจึงไม่อาจจะทำ operations บางอย่างเช่น upper_bound หรือ lower_bound ได้

ตัวอย่าง

- ให้ดูในไฟล์ maps.cpp

Algorithms

- sort(first_iterator, last_iterator, [comparator]) รับพารามิเตอร์เข้ามาเป็น iterators 2 ตัว เป็นจุดเริ่มต้นและจุดสิ้นสุดของตำแหน่งที่ต้องการจะทำการเรียงลำดับ
- reverse(first_iterator, last_iterator) รับพารามิเตอร์เข้ามาเป็น iterators 2 ตัว เป็นจุดเริ่มต้นและจุดสิ้นสุดของตำแหน่งที่ต้องการจะทำการกลับด้านลิสท์
- max_element(first_iterator, last_iterator)/min_element(first_iterator, last_iterator) รับพารามิเตอร์เข้ามาเป็น iterators 2 ตัว เป็นจุดเริ่มต้นและจุดสิ้นสุดของตำแหน่งที่ต้องการหาค่าสูง/ต่ำสุด (ค่าที่คืนจะเป็น iterator ชี้ไปยังตำแหน่งผลลัพธ์)
- binary_search(first_iterator, last_iterator, x) รับพารามิเตอร์เข้ามาเป็น iterators 2 ตัว เป็นจุดเริ่มต้นและจุดสิ้นสุดของตำแหน่งที่ต้องการทดสอบว่า x นั้นปรากฏอยู่ในลิสท์หรือไม่
- lower_bound(first_iterator, last_iterator, x) รับพารามิเตอร์เข้ามาเป็น iterators 2 ตัว เป็นจุดเริ่มต้นและจุดสิ้นสุดของตำแหน่งที่ต้องการคืนค่า iterator ที่เป็น lower_bound
- upper_bound(first_iterator, last_iterator, x) รับพารามิเตอร์เข้ามาเป็น iterators 2 ตัว เป็นจุดเริ่มต้นและจุดสิ้นสุดของตำแหน่งที่ต้องการคืนค่า iterator ที่เป็น upper_bound

แบบฝึกหัด 3

- กำหนดให้ List ต่อไปนี้ 10, 12, 15, 19, 20, 1, -3, -7, 8, 14, 12, 30, 58, -100, 44
- ให้เขียนโปรแกรมเพื่อตอบคำถามต่อไปนี้
  - ให้พิมพ์ค่า List แบบเรียงลำดับจากน้อยไปหามาก
  - ให้พิมพ์ค่า List แบบเรียงลำดับจากมากไปหาน้อย
  - หาค่าสูงสุด/ต่ำสุด ของ List ดังกล่าว
  - ให้หาอย่างมีประสิทธิภาพว่าใน List มีค่า 7 อยู่หรือไม่
  - ให้หาอย่างมีประสิทธิภาพว่าใน List มีค่า -7 อยู่หรือไม่
  - ให้พิมพ์ค่าแบบเรียงลำดับจากน้อยไปหามากตั้งแต่ตำแหน่งที่มีค่ามากกว่า 0 (เรียกใช้ lower_bound)
  - ให้พิมพ์ค่าแบบเรียงลำดับจากน้อยไปหามากตั้งแต่ตำแหน่งที่มีค่าน้อยกว่า 40 (เรียกใช้ upper_bound)

References

- https://www.geeksforgeeks.org/the-c-standard-template-library-stl/
- https://www.codechef.com/problems/school/
- https://beta.programming.in.th/tasks
- https://www.programming.in.th/