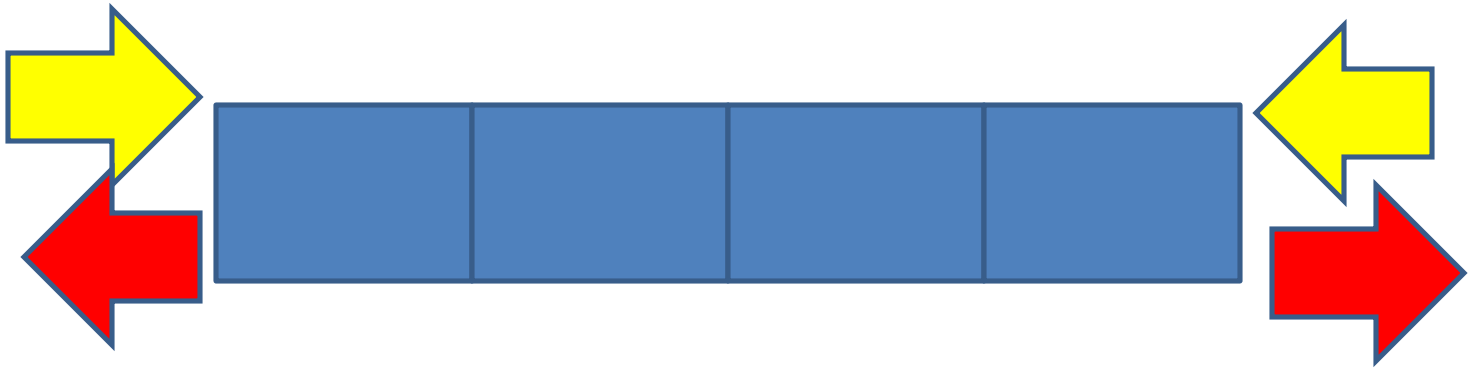
Deque (double ended queue) usage

- ใช้เก็บ sequence ของดาต้า
- โตเองได้
- เอาของใส่/ออก ได้จากปลายทั้งสองข้าง



- ไม่เหมือนเวคเตอร์ ตรงที่ มันไม่เก็บข้อมูลในส่วนของ memory ที่ติดกัน (data จริงๆอาจกระจายทั่วไปใน memory) ดังนั้นใช้ pointer access แล้ว ++ เอาโดยตรงไม่ได้
- เหมาะกับการเก็บ data sequence ยาวๆ เพราะกระจายหาที่เก็บได้ง่าย
- ใช้ [] ดูของภายในได้เหมือนเวคเตอร์และอาร์เรย์ (แต่ถ้าเกิน range จะรีเทิร์นค่ามั่วออกมาให้ใช้ได้)
- at(i) ใช้ได้ และจะ throw exception ให้ถ้าเกินrange ของ deque
- front() รีเทิร์นของที่เก็บข้างหน้าสุด
- back() รีเทิร์นของที่เก็บข้างหลังสุด

fields

- มี iterator ตามปกติ และมี reverse_iterator ด้วย
- std::reverse_iterator

```
for (std::deque<int>::iterator it = myQ.begin(); it!=myQ.end();
    ++it)
        std::cout << ' ' << *it;
```

constructors

```
std::deque<int> first; // empty deque of ints
std::deque<int> first(5); // ใส่ 0 ไป 5 ตัว

std::deque<int> second (4,100); // four ints with value 100

std::deque<int> third (second.begin(),second.end());
// iterating through second

std::deque<int> fourth (third); // a copy of third
```

```
int myints[] = {16,2,77,29};
std::deque<int> fifth (myints, myints + sizeof(myints)
    /sizeof(int) );


std::deque<std::string> words1 {"the", "frogurt", "is", "also",
    "cursed"};
std::cout << "words1: " << words1 << '\n';
```

destructor

~deque();

เป็นการคืนหน่วยความจำที่จองไว้ (แต่ถ้าสมาชิกของคิวเป็นพอยเตอร์ ออบเจ็กต์ที่ถูกชี้จะไม่คืนนะ)

=

- ใช้ copy ของจากคิวขวามาใส่คิวซ้าย และเปลี่ยนขนาดของคิวซ้ายด้วย

std::deque<*int*> first (3); *// deque with 3 zero-initialized ints*

std::deque<*int*> second (5); *// deque with 5 zero-initialized ints*

second = first; //ตอนนี้มีขนาด 3

first = std::deque<*int*>(); //ตอนนี้มีขนาด 0

std::deque<int> nums1 {3, 1, 4, 6, 5, 9}; เวอร์ชันเก่าใช้ไม่ได้

std::deque<int> nums2;

std::deque<int> nums3;

nums3 = std::move(nums1); //เอาของจาก num1 มาใส่ num3 เลย ทำให้ num1 ไม่มีของเหลือเลย

iterators

begin()

end()

rbegin() // reverse direction ถ้า ++ มันจะเลื่อนไปทางข้อมูลชิ้นแรก

rend()

ฟังก์ชันเกี่ยวกับ size

- size() จำนวนของที่เก็บใน deque
- max_size() จำนวนของที่เก็บได้มากสุด (ขึ้นกับระบบ)
- resize(n) ใช้ขยายหรือย่อขนาด deque ได้ ถ้าย่อจะตัดข้างหลังทิ้งไปเลย ถ้าขยายจะเติมค่า default ให้ หรือเราบอกค่าให้ก็ได้ โดยใส่เป็นพารามิเตอร์ที่สอง
- empty() คือ isEmpty นั่นเอง

ฟังก์ชันที่ใช้เปลี่ยนค่าต่างๆที่เก็บใน deque

- assign ใช้เปลี่ยน deque ให้เป็น deque ใหม่เลย ทำลายข้อมูลเก่าทิ้งหมด
- assign (7,100) //เลข 100 7 ตัว
- assign (iterator1,iterator2) //ใช้ iterator กำหนดช่วงของข้อมูลที่จะใส่ deque ได้
- ใช้ initializer list ได้ด้วย

Int myints[] = {1776,7,4};

mydeque.assign (myints,myints+3);

- push_back (data) เติมของที่ด้านหลัง
- push_front (data) เติมของที่ด้านหน้า
- pop_back() เอาของที่อยู่ท้ายสุดออก ไม่ได้รีเทิร์นค่าอะไร
- pop_front() เอาของที่อยู่หน้าสุดออก ไม่ได้รีเทิร์นค่าอะไร
- Insert แทรกสมาชิกเข้าไป ณ ตำแหน่งที่กำหนด
  - insert (iterator,data) ใส่ข้อมูล 1 ตัว
  - insert (iterator,n,data) ใส่ data ไป n ครั้งที่ตำแหน่ง iterator
  - insert (iterator, c.begin(), c.end()) ใส่ของจาก c เข้ามา

- erase คือการเอาของออกโดยบอกเป็นช่วงหรือบอกตำแหน่ง ตัวชี้ตำแหน่งต้องเป็น iterator ของ deque นั้นๆ
  - erase (iterator) ลบของออกตำแหน่งเดียว
  - erase (iterator1, iterator2) ลบของออกเป็นช่วง
- swap(anotherDeque) สลับของกันระหว่าง deque เลย iterator ที่ชี้สมาชิกไหนก็ยังชี้ที่เดิม แต่เป็นในคิวใหม่แล้ว
- clear() ลบของออกจากคิวทั้งหมดและทำให้ size เป็น 0

การเปรียบเทียบ deque

- ==, != คิวสองคิวเทียบกันโดยดู size ก่อน ถ้าเท่ากันจะไปเทียบสมาชิกข้างใน เรียงทีละตัว ถ้าเรียงไม่เหมือนกันหรือมีสิ่งที่ต่างกันก็ไม่เท่า
- <,>,<=,>= เปรียบเทียบสมาชิกเรียงตัว แต่ละตัวตรวจด้วยเครื่องหมาย < ถ้าได้คำตอบตอนไหนก็จะเอาไปตอบเลย

แบบฝึกหัด

- สร้าง deque ที่มี 1,2,3,4,5 แล้ว เขียนฟังก์ชัน

divisible(deque<int> q, int n) ซึ่งจะลบค่าที่หารด้วย n ไม่ลงตัวทิ้งไป เหลือแต่ค่าที่หารด้วย n ลงตัว ในการเขียนฟังก์ชันนี้ ให้ใช้เฉพาะ size(), front(), back(), push_back(..), pop_front() เท่านั้น โดยสมาชิกที่เหลือของคิวต้องยังเรียงตามลำดับเดิม

- เขียนฟังก์ชัน sortStack(deque<int> q) ซึ่ง sort ข้อมูลมากไปน้อยเรียงจากตัวแรกไปตัวสุดท้าย (สร้าง deque อีกตัวหนึ่งมาช่วยได้) โดย deque เหล่านี้ให้ใช้เฉพาะ back(), push_back(..), pop_back(), empty() เท่านั้น ถ้า deque ว่าง ต้องรีเทิร์นตัวว่างด้วย

Code สำหรับเทส

```
deque<int> q1;
for(int i =1; i<=12; i++)
    q1.push_back(i);
printDeque(q1);
deque<int> q2 = divisible(q1,4);
printDeque(q2);
deque<int> q3;
q3.push_back(6);
q3.push_back(1);
q3.push_back(3);
q3.push_back(5);
q3.push_back(4);
q3.push_back(0);
q3.push_back(7);
printDeque(q3);
```

```
void printDeque(deque<int>
```