Phill-DS-1122

使用原來的class Queue

股票 moving average

- 固定時間區間 window =5
- 平均值

a sequence of data → int data[] ={10,50,40,20,100,80,60,70};

每走一格, 印出移動平均值

→ 更改 main, class 也可些許更改

→ 加一個 MovingAverage class

```
#include <iostream>

// 定義節點結構
struct Node {
    int data;
    Node* next;
};

class QueueList {
private:
    Node* front;  // 指向queue頭部的指標
    Node* rear;   // 指向queue尾部的指標

public:
    QueueList() : front(nullptr), rear(nullptr) {}  // 建構子，初始化front和rear為nullptr

    int size(){
      int mySize=0;
      Node* temp=front;
      while(temp!=nullptr){
        mySize++;
        temp= temp->next;
      }
      return mySize;
    }

    // 加入元素到queue的尾部
    void enqueue(int val) {
        Node* newNode = new Node;
```

```
        newNode->data = val;
        newNode->next = nullptr;
        if (rear == nullptr) {
            front = rear = newNode; // 如果是第一次加入，設定front和rear都指向新節點
            return;
        }
        rear->next = newNode;  // 將新節點加入到尾部
        rear = newNode;        // 更新rear指標
    }

    // 從queue的頭部取出元素
    int dequeue() {
        if (front == nullptr) {
            std::cout << "Queue underflow!" << std::endl;
            return -1;
        }
        Node* temp = front;
        int dequeuedValue = temp->data;
        front = front->next;  // 更新front指標
        if (front == nullptr) rear = nullptr; // 如果取出後queue為空，重設rear
        delete temp;
        return dequeuedValue;
    }

    // 檢查queue是否為空
    bool isEmpty() {
        return front == nullptr;
    }

    // 解構子，釋放記憶體
    ~QueueList() {
        while (!isEmpty()) {
            dequeue();
        }
    }
};

class MovingAverage {
  private:
    QueueList data;
    int windowSize;
    int movingSum;

  public:
    MovingAverage(int size) : windowSize(size), movingSum(0) {}

    double appendValue(int val){
      if(data.size()==windowSize){
        movingSum -=data.dequeue(); //總和扣掉front
      }
      data.enqueue(val);
      movingSum+=val;
```

```
      return (double)(movingSum/data.size());
    }
};

int main(){
  MovingAverage abc;
  double showVal;
  int seq[]={10,......};
  for(int i=0; i<len(seq);i++){
    showVal=MovingAverage.appendValue(seq[i]);
    cout << "now moving average = " << showVal << endl;
  }
}
```