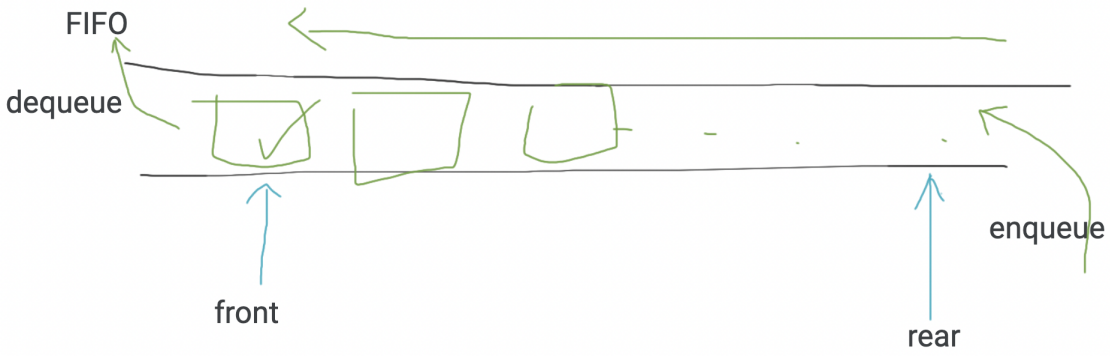
Phill-DS-1108

Queue

- FIFO (first in first out)
- enqueue, dequeue, front, rear (從左到右)
- 注意 array size
- -1 代表重置 初始狀態 → 避免元素搬動




```
#include <iostream>
#define MAX_SIZE 10
using namespace std;

class QueueArray{
  private:
    int front, rear; // index
    int arr[MAX_SIZE];

  public:
    QueueArray(): front(-1), rear(-1){}

    void enqueue(int data){
      if(rear >= MAX_SIZE-1){
        cout << "已經滿了喔" << endl;
        return;
      }

      arr[++rear] = data; // 有 element
```

```
      if(front==-1) front=0; // new element -> front 連動更新
    }

    int dequeue(){
      if(front ==-1){ //初始或被重置後的狀況 -> 空的queue
        cout << "已經空了喔"<< endl;
        return -9999; //錯誤碼
      }
      int value = arr[front];
      if(front==rear) // 為了 maintain front rear 機制
      {
        rear=front=-1;  //這種作法的好處是不用搬動 array
      }//取出最後一個元素 重置queue
      else{
        front++; //取出來了(變垃圾) front 向後指
      }
      return value;
    }

    bool isEmpty(){
      (front==-1)?true:false; //利用重置機制
    }
};


int main()
{
    QueueArray qu;
    qu.enqueue(94);
    cout << qu.dequeue() << endl;

    qu.enqueue(94);
    qu.enqueue(87);
    cout << qu.dequeue() << endl;
    cout << qu.dequeue() << endl;

    qu.enqueue(87);
    cout << qu.dequeue() << endl;
    cout << qu.dequeue() << endl;


    return 0;
}
```

Linked List

```
#include <iostream>
using namespace std;

struct Node{
  int data;
  Node* next;
};

class QueueList{
  private:
    Node* front;
    Node* rear;

  public:
    QueueList(): front(nullptr), rear(nullptr){}

    void enqueue(int data){
      Node* newNode = new Node; //產生實體
      newNode-> data = data;
      newNode-> next = nullptr;
      if(rear==nullptr){ //設定好重置 初始
        front = rear = newNode; //queue 只有我
        return;
      }

      rear->next = newNode;
      rear = newNode;
    }
};

int main()
{
    //cout << "Hello, World!";
    return 0;
}
```