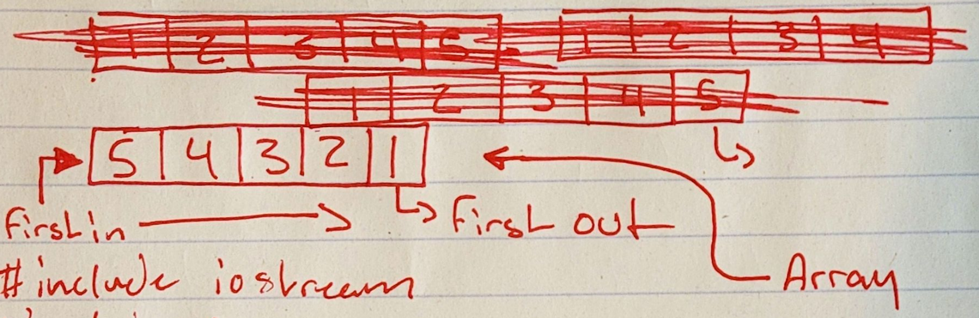
Queues Structure:

↳ First in first out data container.
 in→ out →

queue <data type> queue name;



```
#include iostream
#include queue
main ()
 // Declaration of queue w/ given name
 • queue <int> queue_name;
 // insert element in queue container
 • queue_name. push(1)
        }
 // removing / extracting the content from queue container
 • While queue_name not empty
  • Print whats in front of the queue
  • Pop () front of queue
```

Member types:
- Container_type: provides type of container adapted by queue
- Size_type: (int) Showing num of elements in queue
- Value_type: represents type of elements in queue.

Functions of queue:
- queue::empty → checks if empty
- queue::size → checks num of elements
- queue::front → Details about front
- back() → Details about back
- Push() → inserts new element into queue.
- Pop() → removes oldest
- Swap() → swaps contents of two queues, must be same data type
- emplace() → Adds new element to end.

linked list Structure:

// A linked list Node

- $\boxed{\text{int var}} \rightarrow \boxed{\text{Construct Node}} \rightarrow \boxed{\text{*Node next}}$

Push Node

Insert new node in front of list
- Inputs (Node** head, int node data)
  - Create and allocate node
  - assign data to node
  - set next of new node as head
  - Move head to point to new node.

Insert Node

Insert new node after given node
- Inputs (Node* prev_node, int node_data)
  - Check if prev_node is null
  - assign data to node
  - Make next of new node the next of prev_node
  - Move next of prev_node as new node.

Append Node

Insert new node at end of linked list
- Inputs (Node** head, node_data)
  - Create and allocate node
  - Assign data to node
  - Set next pointer of new node to null (is last)
  - if list is empty, new node becomes first
  - Else Move ~~Search~~ till the last node
  - Change next of last node

Display linked list contents