Stack and Queue

Inst. Nguyễn Minh Huy

Contents

- Stack.
- Queue.

Contents
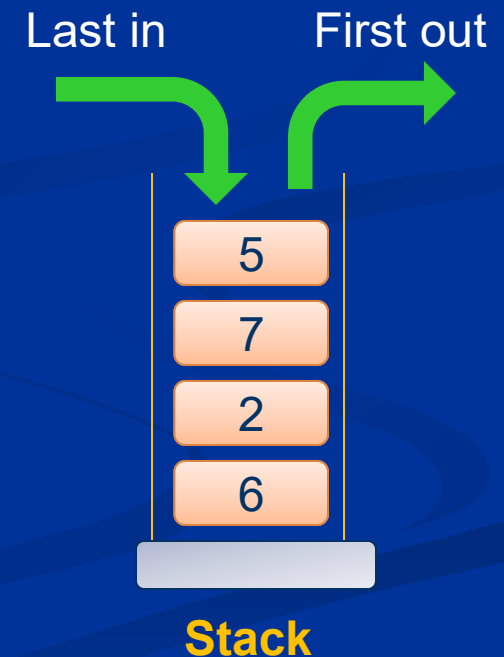
- **Stack.**
- Queue.

Stack

- ## Stack concept:
  - Collection of elements accessed by LIFO method.
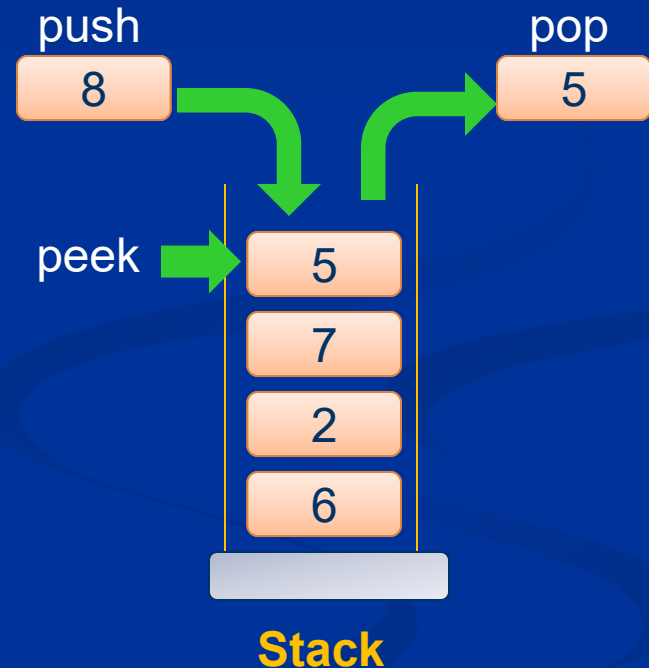 - LIFO (**L**ast **I**n **F**irst **O**ut):
    - Last insert, first removed.

Last in   First out

5
7
2
6

Stack

Stack

- Operations on stack:
  - init: initialize stack.
  - isEmpty: check empty.
  - isFull: check full.
  - push: insert element.
  - pop: remove element.
  - peek: read element.

Stack

Stack

- ## Stack implementation:
  - ### Declaration:

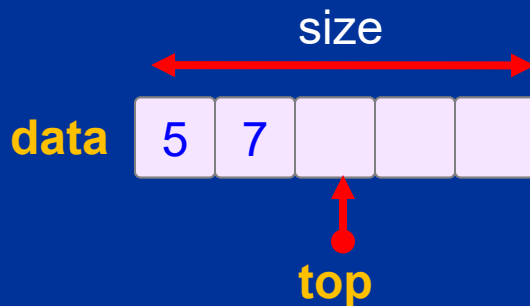*// Use dynamic array.*
```
struct Stack
{
    int  *data;
    int   size;
    int   top;
};
```
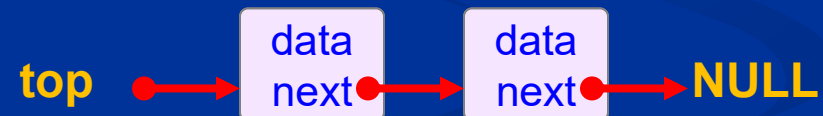
// Use linked list.
```
struct Stack
{
    Node  *top;
};
```
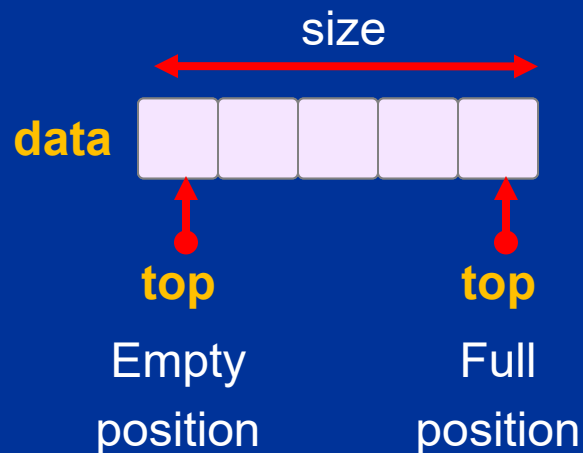
Stack

■ **Stack implementation:**

  ■ init: initialize empty stack.

  ■ isEmpty: check top position.

  ■ isFull: check top position.

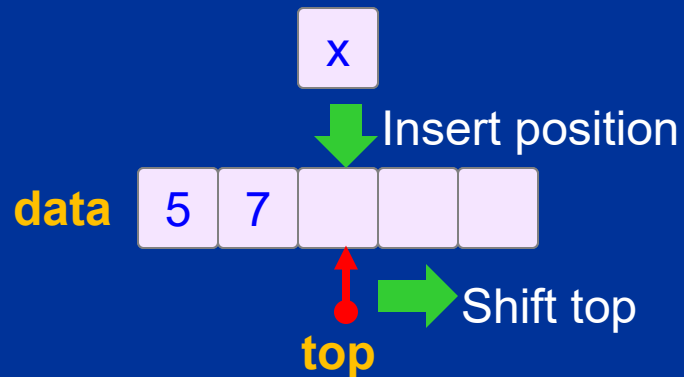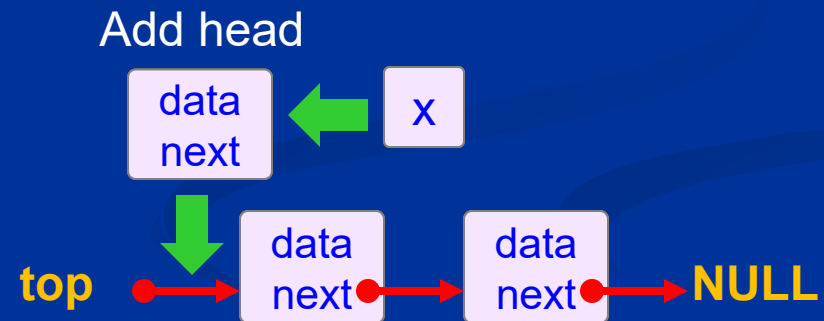**Use dynamic array**

Use linked list

size

data

top
Empty
position

top
Full
position

top → **NULL**
Empty
position

Stack

- ## Stack implementation:
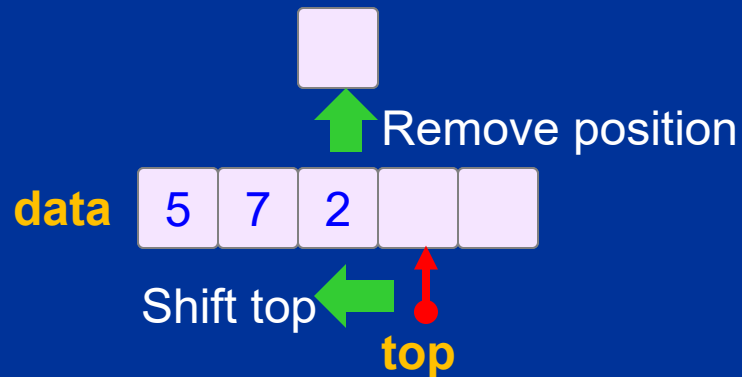 - Push: insert element into stack.

Use dynamic array

Use linked list

Stack

- ## Stack implementation:
  - Pop: remove element from stack.
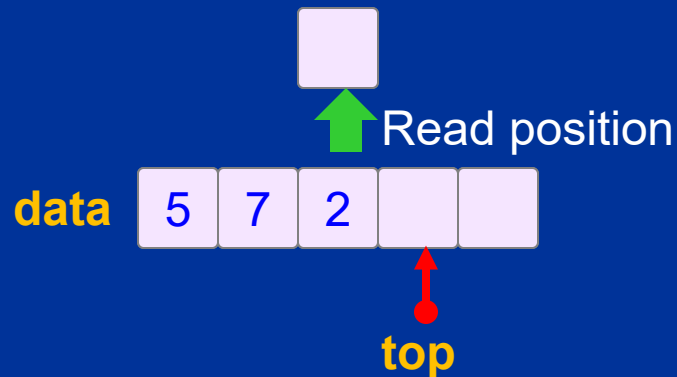
Use dynamic array

Use linked list

Remove position

Remove head

data | 5 | 7 | 2 | | |

Shift top

top

top

data next

data next

NULL
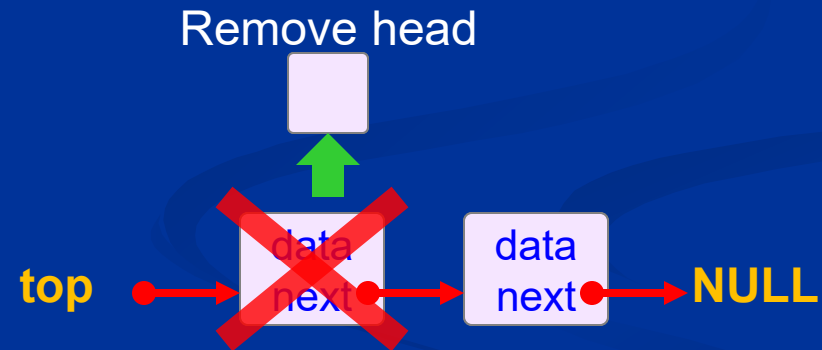
Stack

- # Stack implementation:
  - Peek: read element from stack, do not remove.

Use dynamic array

Use linked list

Read position

Read top

data | 5 | 7 | 2 | | |
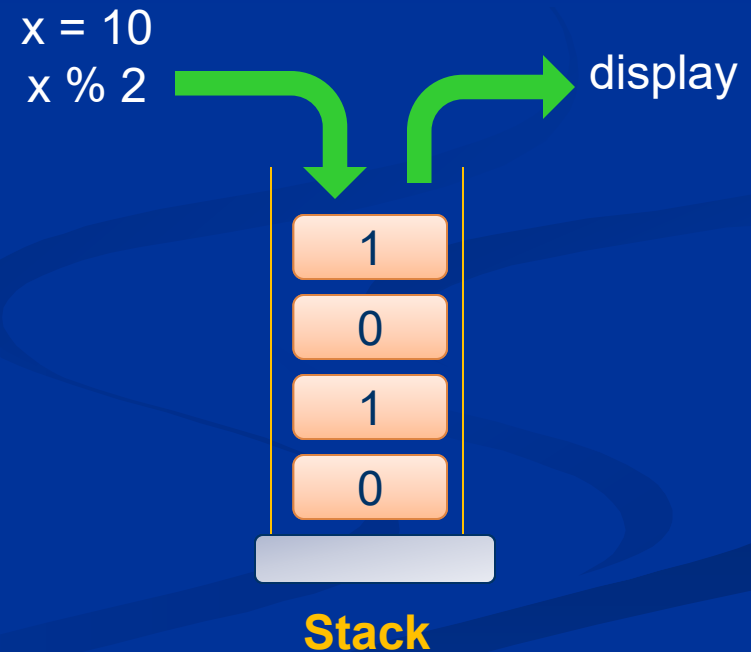
top

top → data / next → data / next → **NULL**
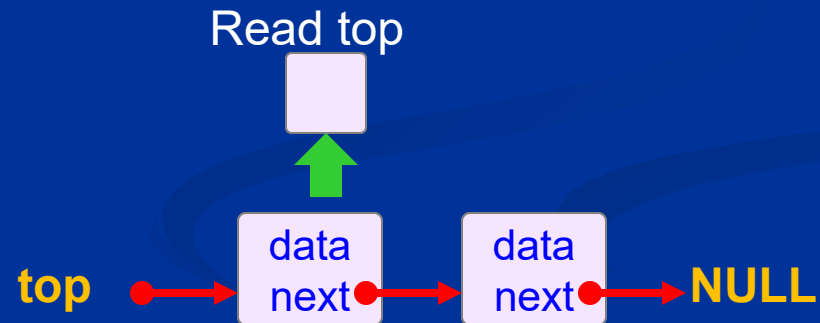
Stack

- **Stack applications:**
  - Perform reversed operations:
    - Convert decimal to binary.
  - Process expression:
    - Reversed Polish Notation.
  - Simulate recursion.

x = 10
x % 2

display

| 1 |
| 0 |
| 1 |
| 0 |

Stack

Contents

- Stack.
- **Queue.**

Queue

- ## Queue concept:
  - Collection of elements accessed by FIFO method.
  - FIFO (**F**irst **I**n **F**irst **O**ut):
    - First come first serve.
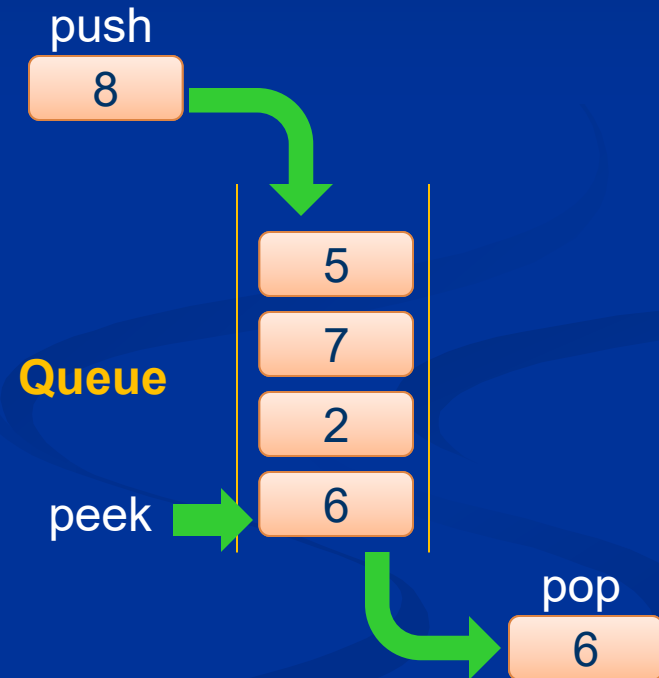   - First insert first remove.

Last insert last remove

Queue

| 5 |
| 7 |
| 2 |
| 6 |

First insert first remove

Queue

- **Operations on queue:**
  - init: initialize queue.
  - isEmpty: check empty.
  - isFull: check full.
  - push: insert element.
  - pop: pop element.
  - peek: read element.

push

8

Queue

5

7

2

peek

6

pop

6

Queue

■ Queue implementation:

■ Declaration:

// Use dynamic array
```
struct Queue
{
        int  *data;
        int  size;
        int  in;
        int  out;
};
```
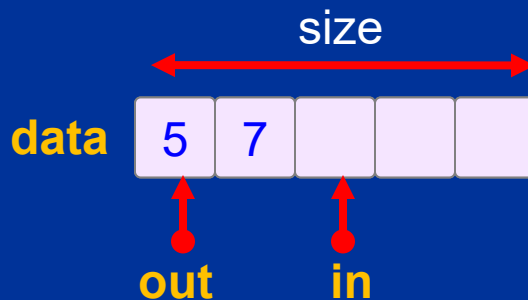
// Use linked list
```
struct Queue
{
        Node  *head;
        Node  *tail;
};
```
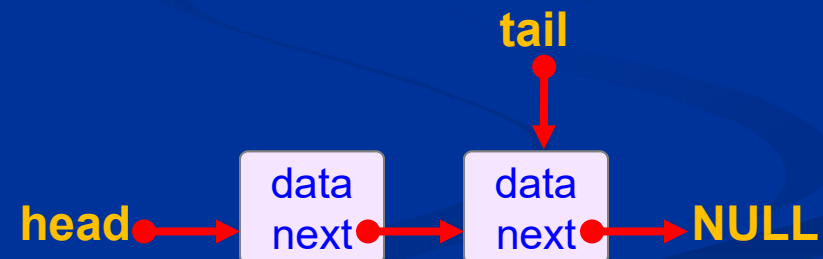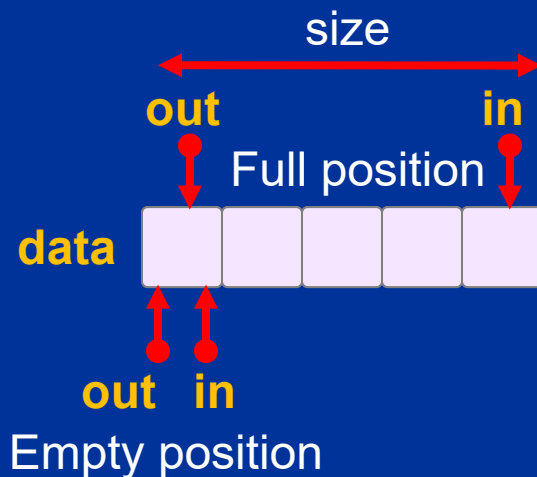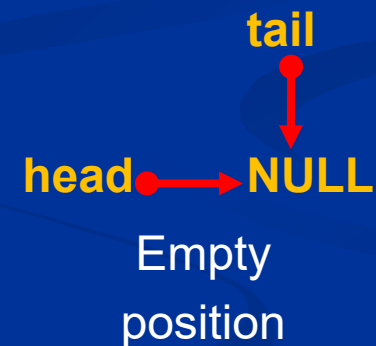
Queue

- **Queue implementation:**
  - init: initialize empty queue.
  - isEmpty: check in and out position.
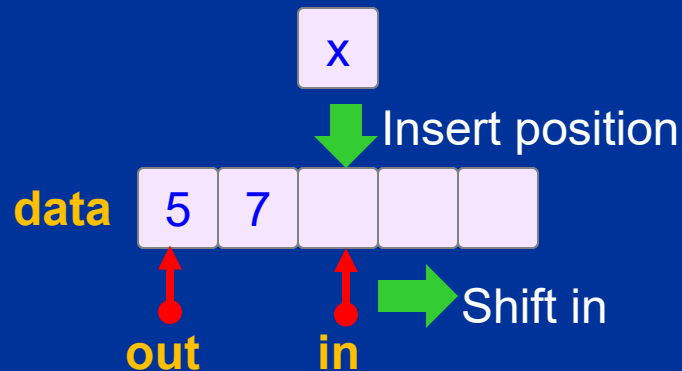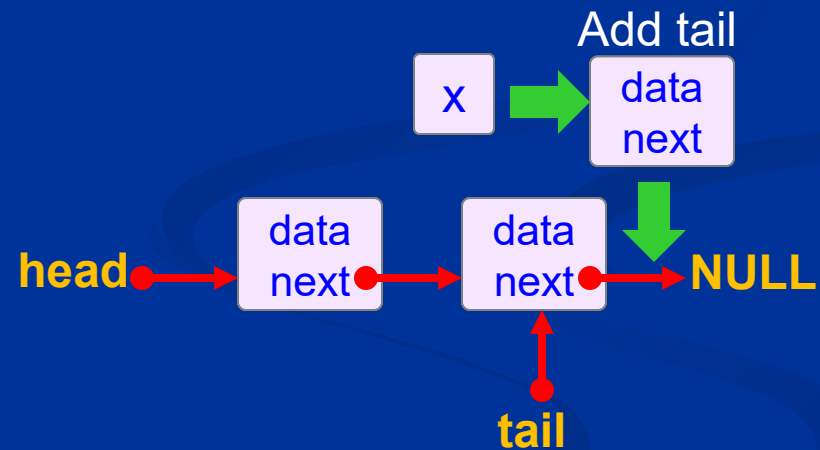 - isFull: check in and out position.

Use dynamic array

size

out   in

Full position

data

out   in

Empty position

Use linked list

tail

head → NULL

Empty
position

Queue

■ **Queue implementation:**

  ■ Push: insert element into queue.

Use dynamic array

Use linked list

Insert position

data | 5 | 7 | | | |

out   in

Shift in

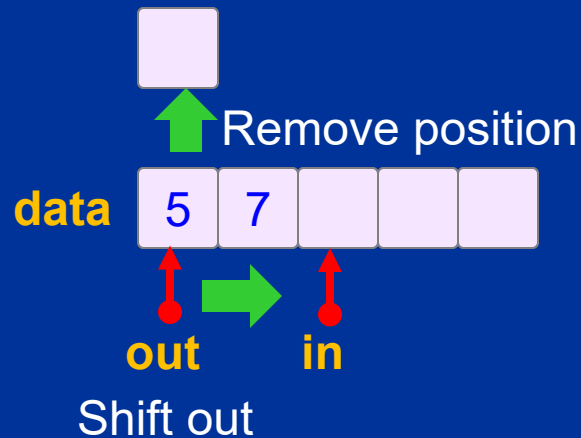Add tail

x

data
next

head

data
next

data
next

NULL

tail

Queue

- ## Queue implementation:
  - Pop: remove element from queue.

Use dynamic array

Use linked list

Remove position

data | 5 | 7 | | | |

out

in

Shift out

Remove head

head → ☒ → data next → NULL

tail

Queue

- ## Queue implementation:
  - ### Peek: read element from queue.

Use dynamic array

Use linked list

Read head

Read position

data | 5 | 7 | | | |

out   in

head → data next → data next → **NULL**

tail
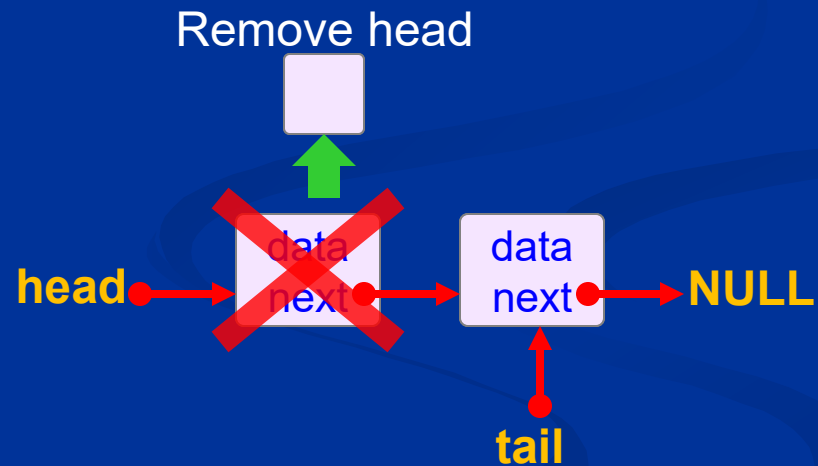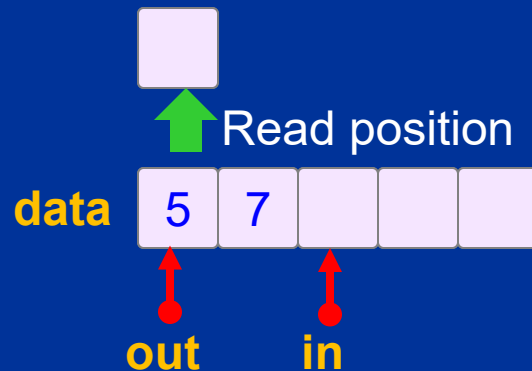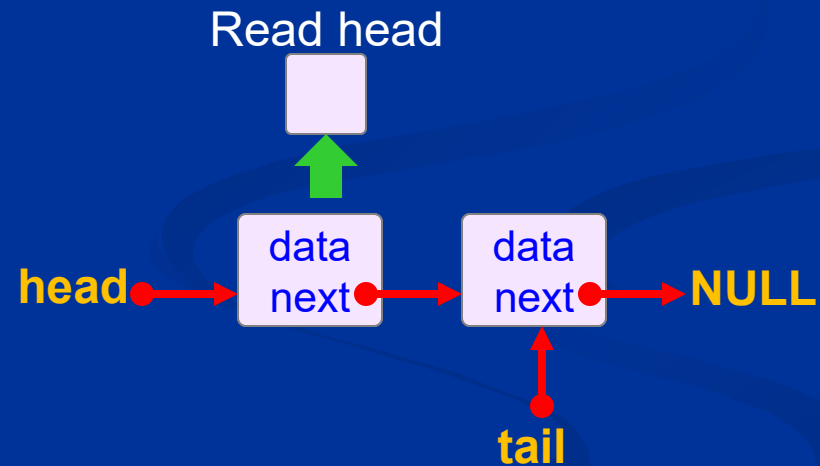
Queue

- **Queue applications:**
  - Breadth-first search in tree.
  - System queue.

- Concept:
  - Stack: LIFO – Last In First Out.
  - Queue: FIFO – First In First Out.
- Operations:
  - init, isEmpty, isFull.
  - push, pop, peek.
- Implementations:
  - Dynamic array.
  - Singly linked list.