UTM
UNIVERSITI TEKNOLOGI MALAYSIA

DATA STRUCTURE & ALGORITHM

Chapter 09

Queue

Part 2 -Queue Implementation Linked List

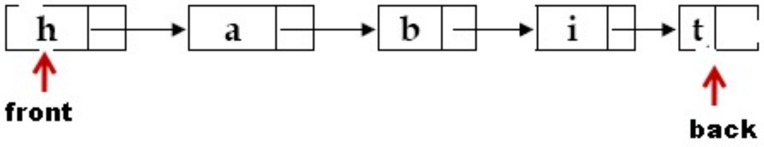
Nor Bahiah Hj Ahmad & Dayang Norhayati A.Jawawi
School of Computing

Queue Implementation Linked List

Pointer-Based Implementation

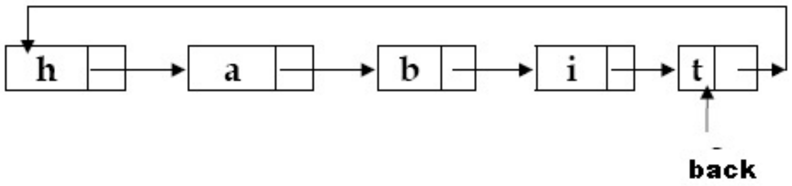
- More straightforward than array-based
- Possible options:
  - Linear linked list

    Two external pointer (Front & Back)



  - Circular linked list

    One pointer will be enough (Back)



Queue Implementation Linked List

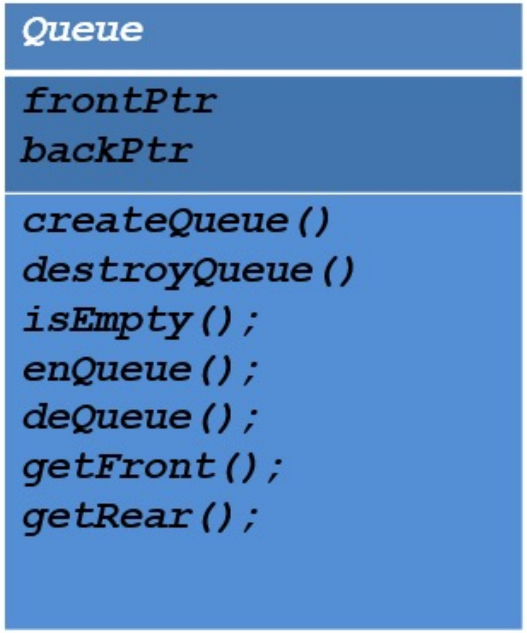
Need 2 structure

- Declaration of the node

```
struct nodeQ {
        char item;
        nodeQ * next;
}
```

- Declaration of the queue

```
class queue
{public:
       nodeQ *backPtr, * frontPtr;
       // operations for queue
};
```



Queue Implementation Linked List

createQueue()

```
backPtr = Null; frontPtr = NULL;
```

destroyQueue()

Destroy the whole nodes in the queue

```
nodeQ *temp = frontPtr;
while (temp){
frontPtr = temp->next;
delete temp; temp=frontPtr; }
```

isEmpty()

```
backPtr == Null && frontPtr == NULL
```

Insert to a linear queue

- Inserting a new node at the Back needs 3 pointer changes

  Change Next pointer in the new node
  Change the Next pointer in the Back node
  Change the External pointer

- Special case:
  - If the queue is empty

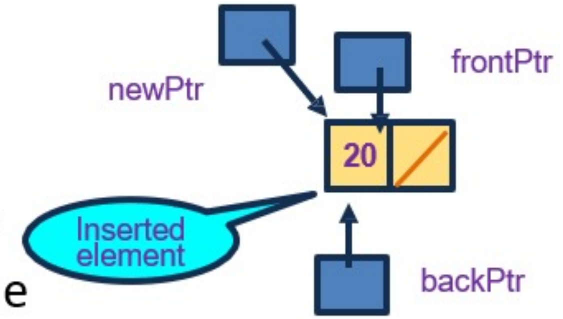
Queue Implementation Linked List

Linear linked list with 2 external pointers

1. Create a new node -> newPtr
2. Insert to an empty queue

    newPtr -> next = NULL

    `frontPtr=backPtr = newPtr`

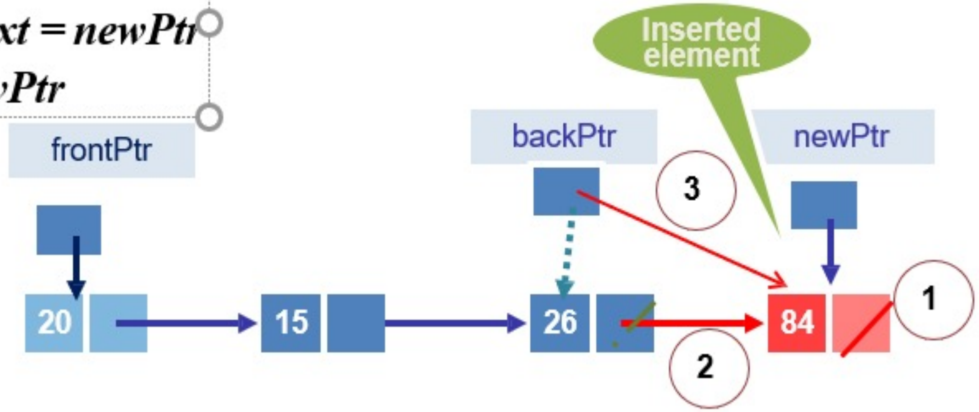


3. Insertion to a non-empty queue

    1 *newPtr -> next = NULL*

    2 *backPtr -> next = newPtr*

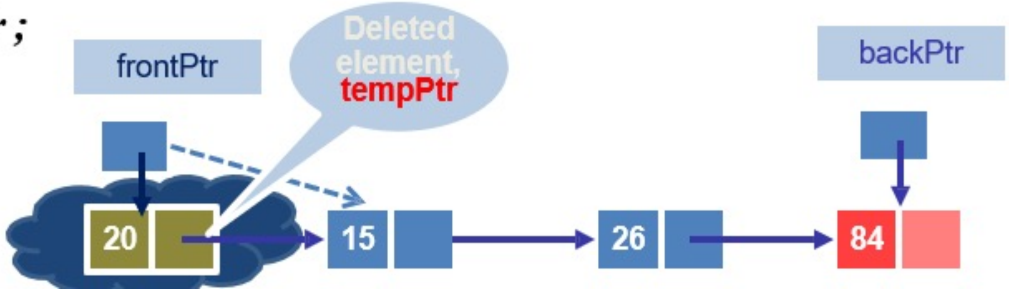
    3 *backPtr = newPtr*



Delete from Linear queue

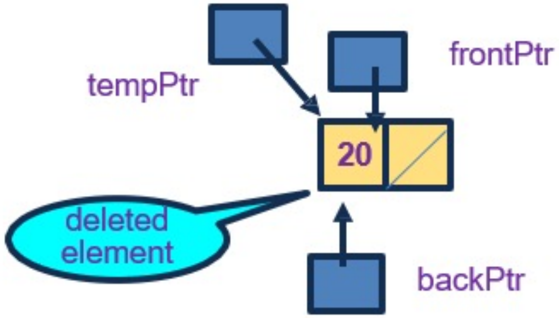
- Deletion
  - Delete from the Front
  - Only one pointer change is needed
  - Special case:
    - If the queue contains one item only
- Deletion Code

```
tempPtr = frontPtr
frontPtr = frontPtr -> next
tempPtr -> next = NULL
delete tempPtr ;
```



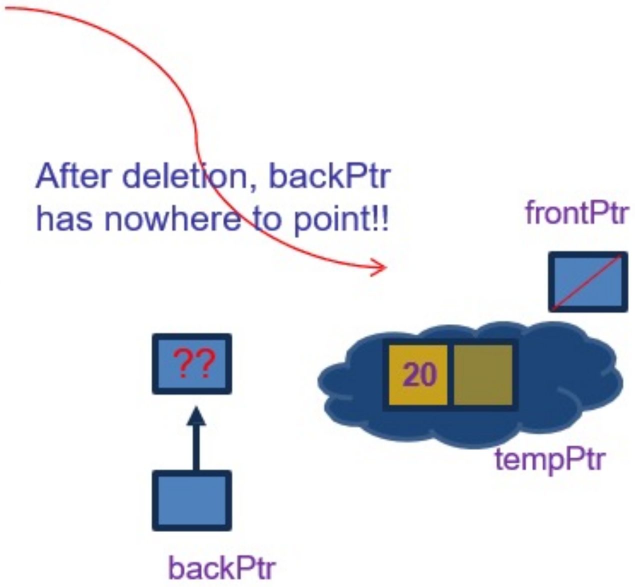
Delete from Linear queue

If the queue contains one item only,



Need to add this statement:

```
If (!frontPtr)
    backPtr = NULL;
```



After deletion, backPtr has nowhere to point!!

Circular Queue Implementation

Circular linear linked list with one external pointer

– Insertion

- Into an empty queue

  NewPtr -> Next = NewPtr

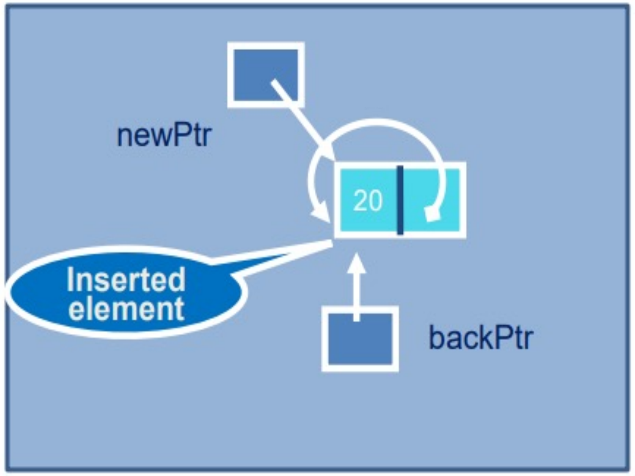
  BackPtr = NewPtr

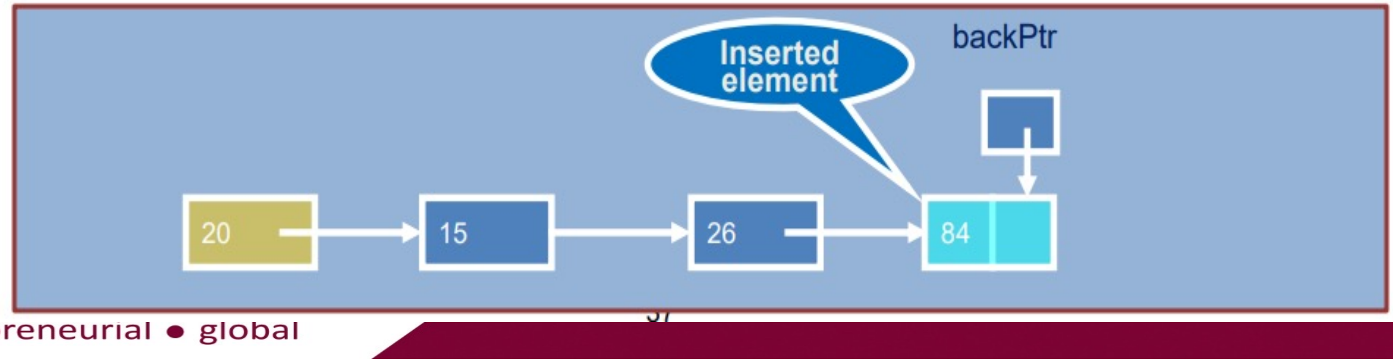
- Into a non-empty queue

  NewPtr -> Next = BackPtr-> Next

  BackPtr -> Next = NewPtr

  BackPtr = NewPtr



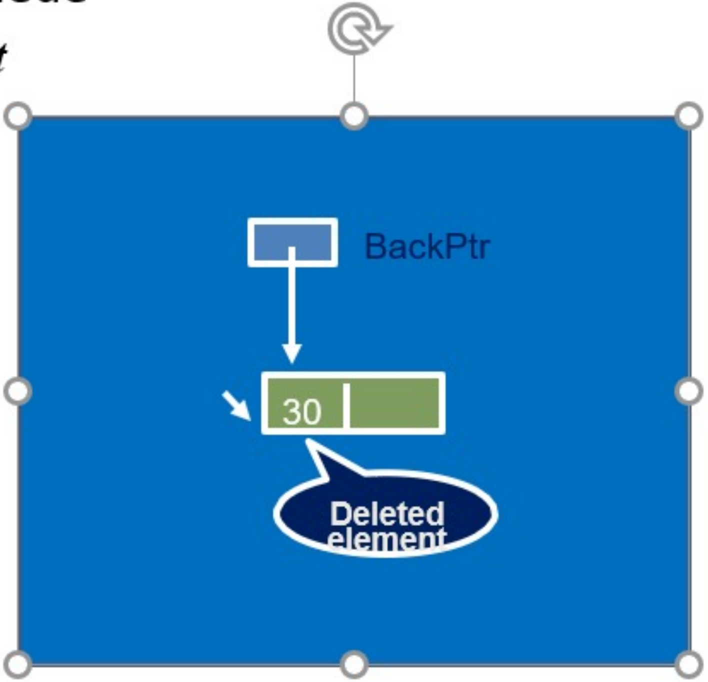


Circular Queue Implementation

Deletion

- From a one-node (one item) queue

```
deletePtr  = BackPtr -> Next
If (deletePtr = BackPtr)
        BackPtr = NULL
delete deletePtr
```

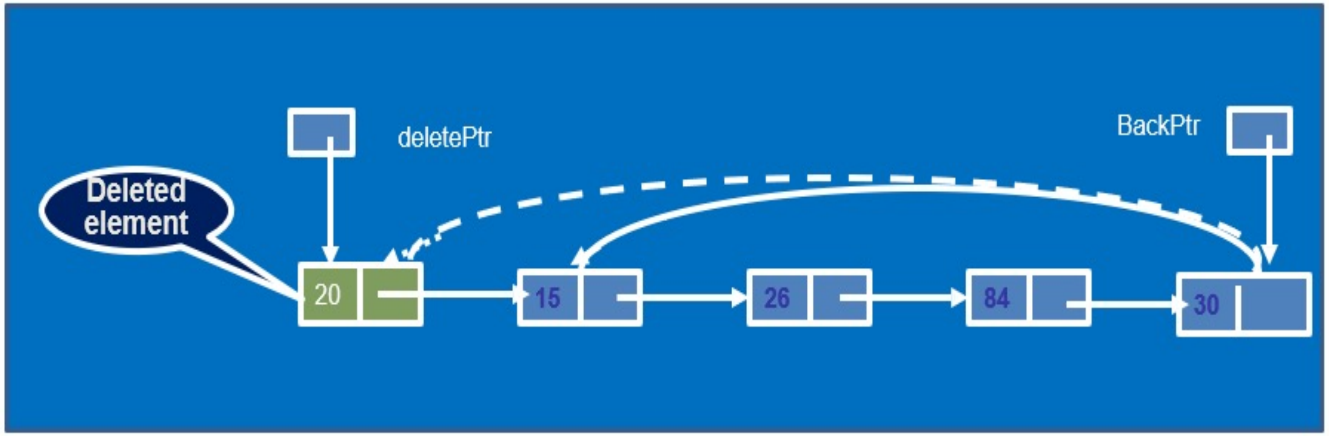


Circular Queue Implementation

Deletion

- From a non-empty, more than one item queue

```
deletePtr  = BackPtr -> Next
BackPtr -> Next =  deletePtr -> Next
delete deletePtr
```



Comparing Implementations

- Fixed size versus dynamic size
  - – A statically allocated array
    - Prevents the *enqueue* operation from adding an item to the queue if the array is full
  - – A resizable array or a reference-based implementation
    - Does not impose this restriction on the *enqueue* operation
- Pointer-based implementations
  - – A linked list implementation
    - More efficient

Comparing Implementations

- Pointer-Based
  - More complicated than ADT List
  - Most flexible No size restrictions
- Array-Based
  - No overhead of pointer manipulation
  - Prevents adding elements if the array is full

Summary of Position-Oriented ADTs

Stacks

- Operations are defined in terms of position of data items
- Position is restricted to the Top of the stack. Only one end position can be accessed
- Operations:
  - *create:*
    - Creates an empty ADT of the Stack type
  - *isEmpty:*
    - Determines whether an item exists in the ADT
  - *push:*
    - Inserts a new item into the Top position
  - *pop:*
    - Deletes an item from the Top position
  - *peek:*
    - Retrieves the item from the Top position

Summary of Position-Oriented ADTs

Queues

- Operations are defined in terms of position of data items
- Position is restricted to the front & back of the queue. Only the end positions can be accessed
- Operations:
  - *create:*
    - Creates an empty ADT of the Queue type
  - *isEmpty:*
    - Determines whether an item exists in the ADT
  - *enqueue:*
    - Inserts a new item in the Back position
  - *dequeue:*
    - Deletes an item from the Front position
  - *peek:*
    - Retrieves the item from the Front position

Summary of Position-Oriented ADTs

- Stacks and queues are very similar
  - Operations of stacks and queues can be paired off as
    - *createStack* and *createQueue*
    - Stack *isEmpty* and queue *isEmpty*
    - *push* and *enqueue*
    - *pop* and *dequeue*
    - Stack *getTop* and queue *getFront*

References

- Frank M. Carano, Janet J Prichard. “*Data Abstraction and problem solving with C++” Walls and Mirrors*. 5th edition (2007). Addision Wesley.
- Nor Bahiah et al. “*Struktur data & algoritma menggunakan C++”. Penerbit UTM. 2005.*