Chapter 19: Stacks and Queues

Kafi Rahman
Assistant Professor @ CS
Truman State University
// Distinct by Design

Int Queue: declaration

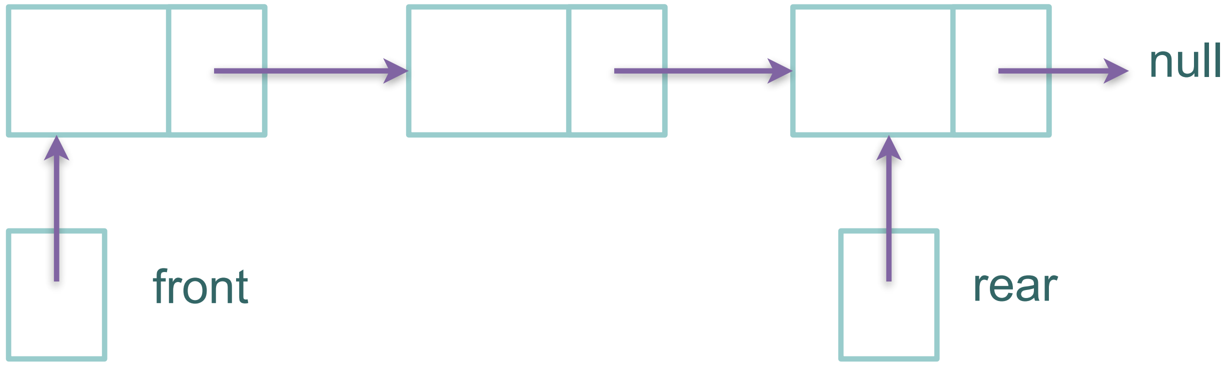
```
class IntQueue
{
private:
   int *queueArray;  // Points to the queue array
   int queueSize;    // The queue size
   int front;        // Subscript of the queue front
   int rear;         // Subscript of the queue rear
   int numItems;     // Number of items in the queue
public:
   // Constructor
   IntQueue(int);

   // Copy constructor
   IntQueue(const IntQueue &);

   // Destructor
   ~IntQueue();

   // Queue operations
   void enqueue(int);
   void dequeue(int &);
   bool isEmpty() const;
   bool isFull() const;
   void clear();
};
```

19.5

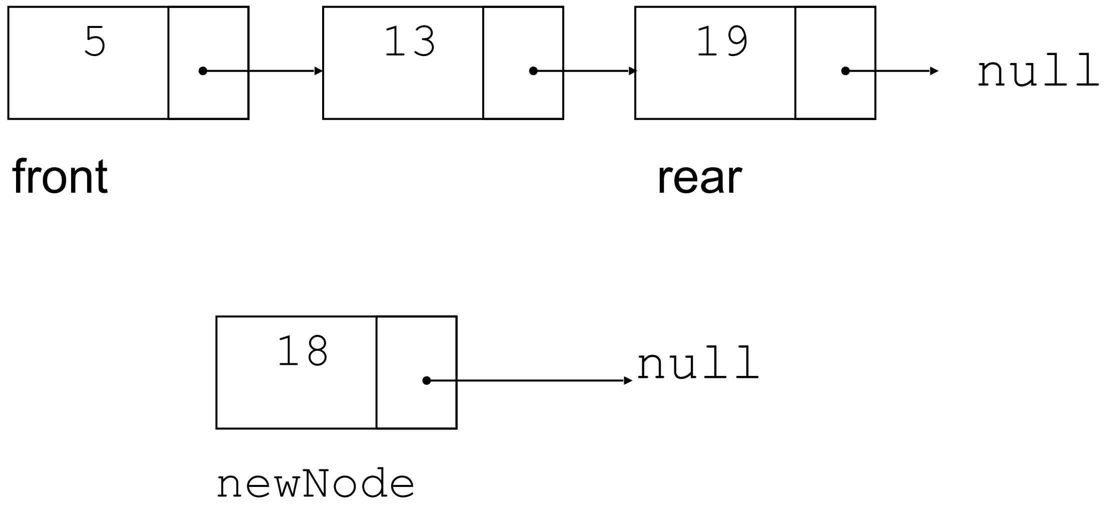
Dynamic Queues

Dynamic Queues

- Like a stack, a queue can be implemented using a linked list
- Allows dynamic sizing, avoids issue of shifting elements or wrapping indices

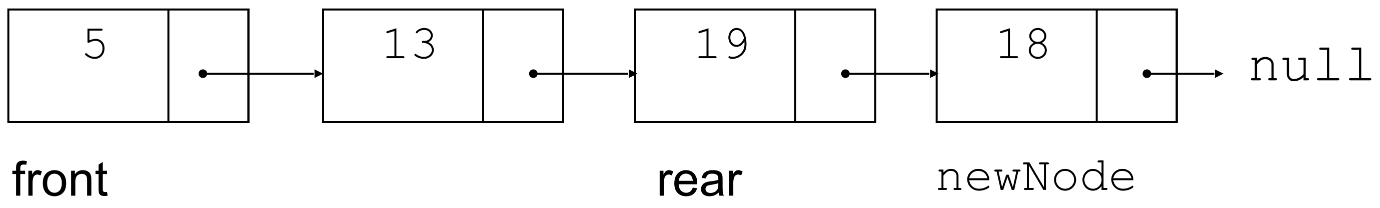


Dynamic Queue: adding an element



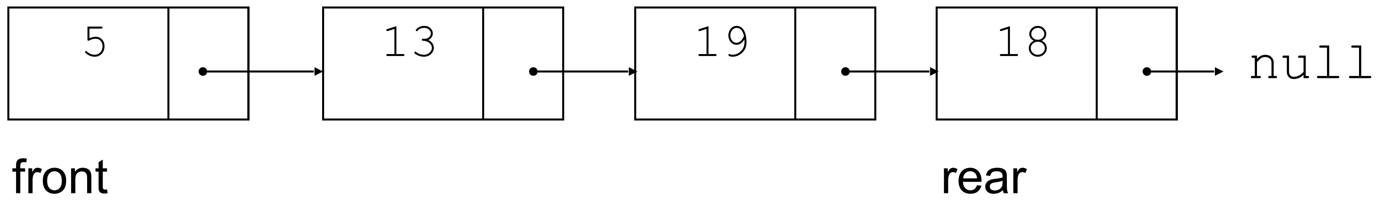
New node created

Dynamic Queue: adding an element



rear points to the newNode

Dynamic Queue: adding an element



newNode becomes the new rear

Implementing a Queue

- Programmers can program their own routines to implement queue operations
- We are going to review the DynIntQue class implementation a dynamic queue
- Lastly, there are existing implementation of queue available in the STL. We are going to discuss a couple of examples of existing STL classes

19.6

The STL deque and queue Containers

The STL deque and queue Containers

- deque: a double-ended queue.
  - Has member functions to enqueue (push_back) and dequeue (pop_front)
- queue: container ADT that can be used to provide queue as a vector, list, or deque.
  - Has member functions to enque (push) and dequeue (pop)

Defining a queue

- Defining a queue of chars, named cQueue

- implemented using a queue:
  - queue<char> cQueue;
- implemented using a list:
  - queue<char, list<char>> cQueue;