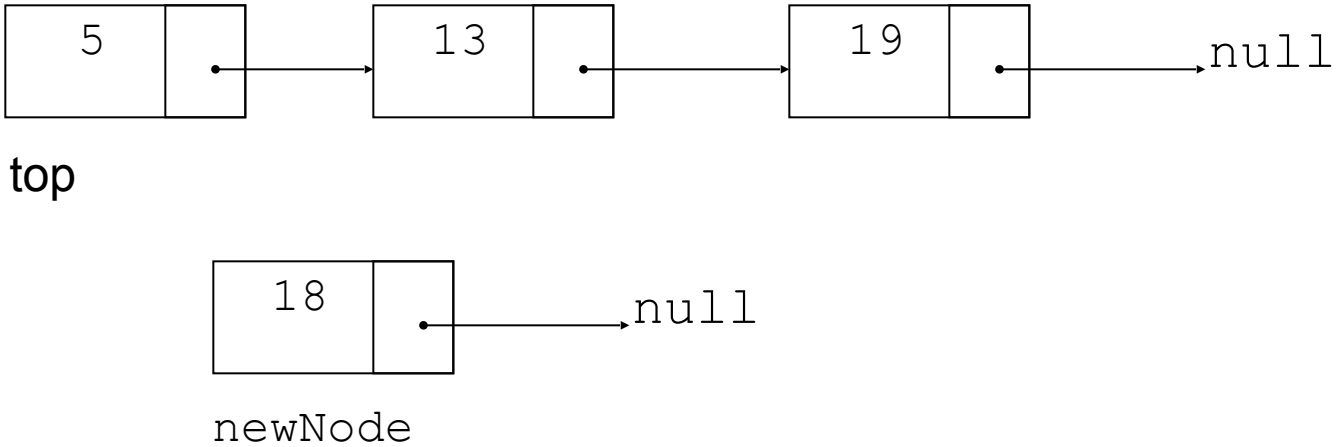
Chapter 19: Stacks and Queues

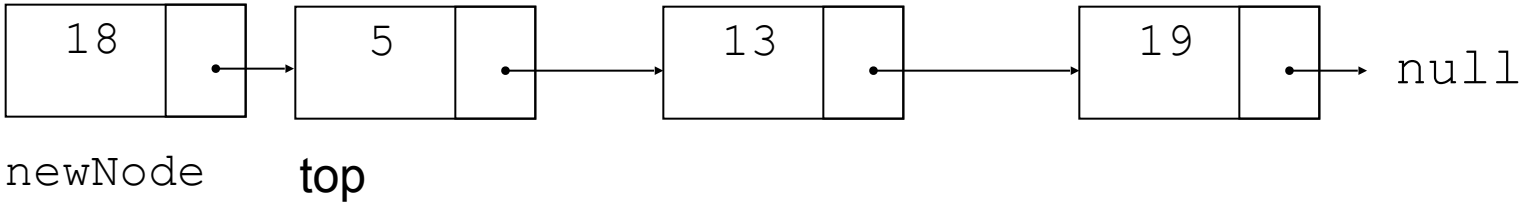
Kafi Rahman
Assistant Professor @ CS
Truman State University
// Distinct by Design

Dynamic Stack: adding an element



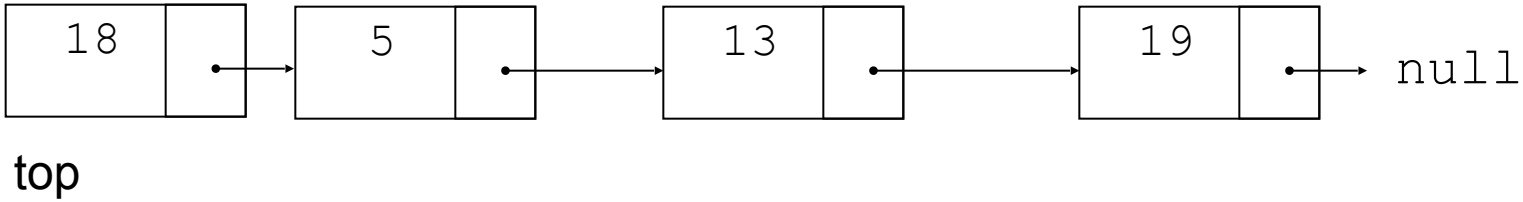
New node created

Dynamic Stack: adding an element



New node points to the current top

Dynamic Stack: adding an element



newNode becomes the new top

19.3

The STL stack Container

The STL stack container

- Stack template can be implemented as a vector, a linked list
- Implements push, pop, and empty member functions
- Implements other member functions:
  - size: number of elements on the stack
  - top: reference to element on top of the stack

Defining a stack

- Defining a stack of chars, named cstack, implemented using a vector:
  - stack< char, vector<char>> cstack;
- implemented using a list:
  - stack< char, list<char>> cstack;

- When using a compiler that is older than C++ 11, be sure to put spaces between the angled brackets that appear next to each other.

stack< char, vector<char> > cstack;

19.4

Introduction to the Queue ADT

Introduction to the Queue ADT

- Queue: a FIFO (first in, first out) data structure.
- Examples:
  - people in line at the theatre box office
  - print jobs sent to a printer
- Implementation:
  - static: fixed size, implemented as array
  - dynamic: variable size, implemented as linked list

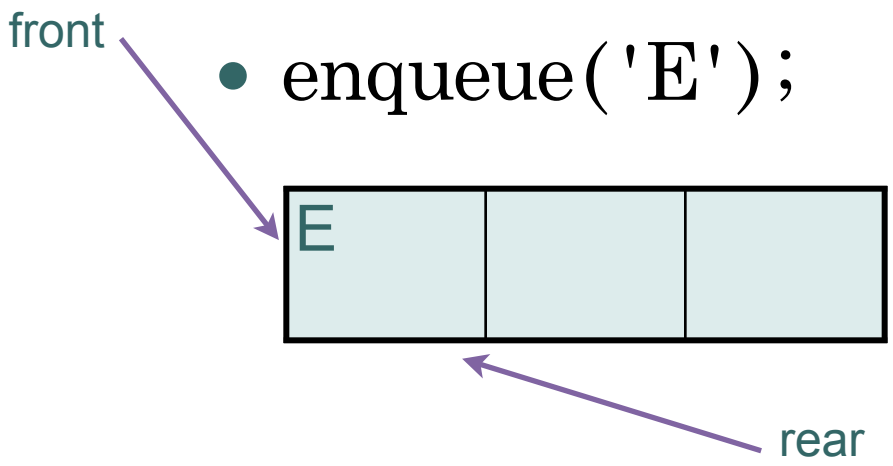
Queue Locations and Operations

- rear: position where elements are added
- front: position from which elements are removed
- enqueue: add an element to the rear of the queue
- dequeue: remove an element from the front of a queue

Queue Operations - Example

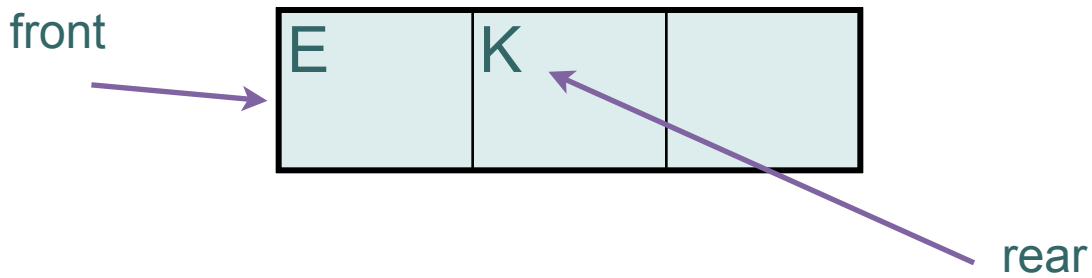
- A currently empty queue that can hold char values:

- `enqueue('E');`

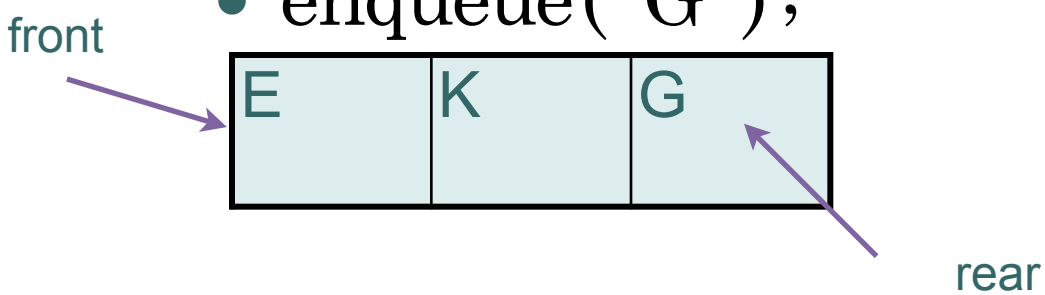


Queue Operations - Example

- enqueue('K');

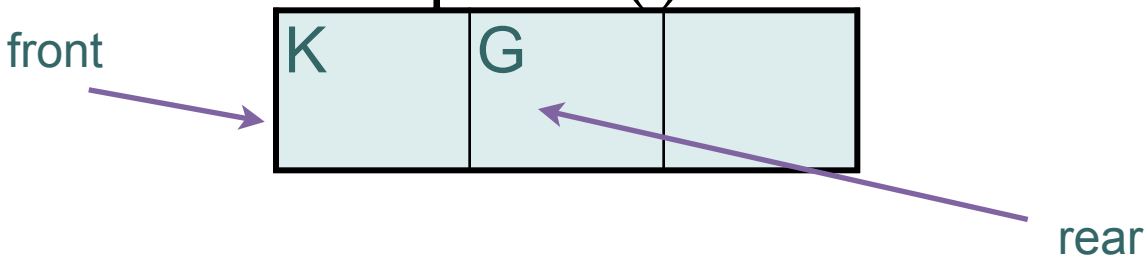


- enqueue('G');

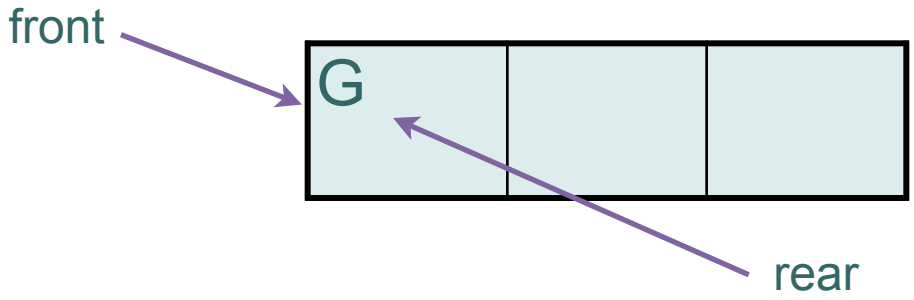


Queue Operations - Example

- dequeue(); // remove E



- dequeue(); // remove K



dequeue Issue, Solutions

- When removing an element from a queue, remaining elements must shift to front
- Solutions:
  - Let front index move as elements are removed (works as long as rear index is not at end of array)
  - Use above solution, and also let rear index "wrap around" to front of array, treating array as circular instead of linear (more complex enqueue, dequeue code)

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();
};
```