



# CS 225

## Data Structures

*February 22 – Stacks, Queues and Design*

*G Carl Evans*

## List.h

```
1 #pragma once
2
3 template <typename T>
4 class List {
5     public:
6         /* ... */
7
8     private:
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33 };
```



# Array Implementation

**insertAtFront:**

<b>C</b>	<b>S</b>	<b>2</b>	<b>2</b>	<b>5</b>
[0]	[1]	[2]	[3]	[4]

# Resize Strategy: +2 elements every time





Resize Strategy: +2 elements every time

# Resize Strategy: x2 elements every time





Resize Strategy: x2 elements every time

# Array Implementation

	Singly Linked List	Array
Insert/Remove at <b>front</b>		
Insert at <b>given</b> element		
Remove at <b>given</b> element		
Insert at <b>arbitrary</b> location		
Remove at <b>arbitrary</b> location		





# Queue ADT

- [Order]:
- [Implementation]:
- [Runtime]:



# Stack ADT

- [Order]:
- [Implementation]:
- [Runtime]:

## Queue.h

```
1 #pragma once
2
3 template <typename T>
4 class Queue {
5     public:
6         void enqueue(T e);
7         T dequeue();
8         bool isEmpty();
9
10    private:
11        T *items_;
12        unsigned capacity_;
13        unsigned size_;
14 };
15
16
17
18
19
20
21
22
```

What type of implementation is this Queue?

How is the data stored on this Queue?

## Queue.h

```
1 #pragma once
2
3 template <typename T>
4 class Queue {
5     public:
6         void enqueue(T e);
7         T dequeue();
8         bool isEmpty();
9
10    private:
11        T *items_;
12        unsigned capacity_;
13        unsigned size_;
14 };
15
16
17
18
19
20
21
22
```

What type of implementation is this Queue?

How is the data stored on this Queue?



```
Queue<int> q;
q.enqueue(3);
q.enqueue(8);
q.enqueue(4);
q.dequeue();
q.enqueue(7);
q.dequeue();
q.dequeue();
q.enqueue(2);
q.enqueue(1);
q.enqueue(3);
q.enqueue(5);
q.dequeue();
q.enqueue(9);
```

## Queue.h

```
1 #pragma once
2
3 template <typename T>
4 class Queue {
5     public:
6         void enqueue(T e);
7         T dequeue();
8         bool isEmpty();
9
10    private:
11        T *items_;
12        unsigned capacity_;
13        unsigned size_;
14 };
15
16
17
18
19
20
21
22
```



`Queue<char> q;`

...

`q.enqueue(m);`

`q.enqueue(o);`

`q.enqueue(n);`

...

`q.enqueue(d);`

`q.enqueue(a);`

`q.enqueue(y);`

`q.enqueue(i);`

`q.enqueue(s);`

`q.dequeue();`

`q.enqueue(h);`

`q.enqueue(a);`