CS 225

Data Structures

February 22 – Stacks, Queues and Design

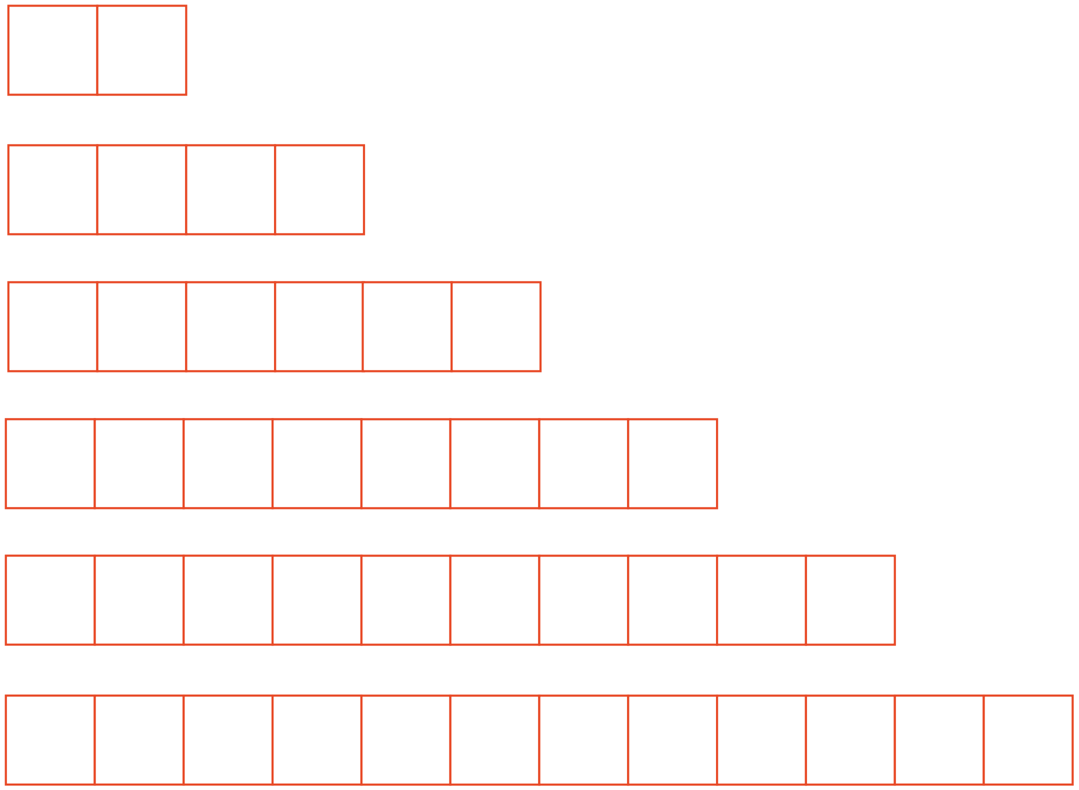
G Carl Evans

Array Implementation

insertAtFront:

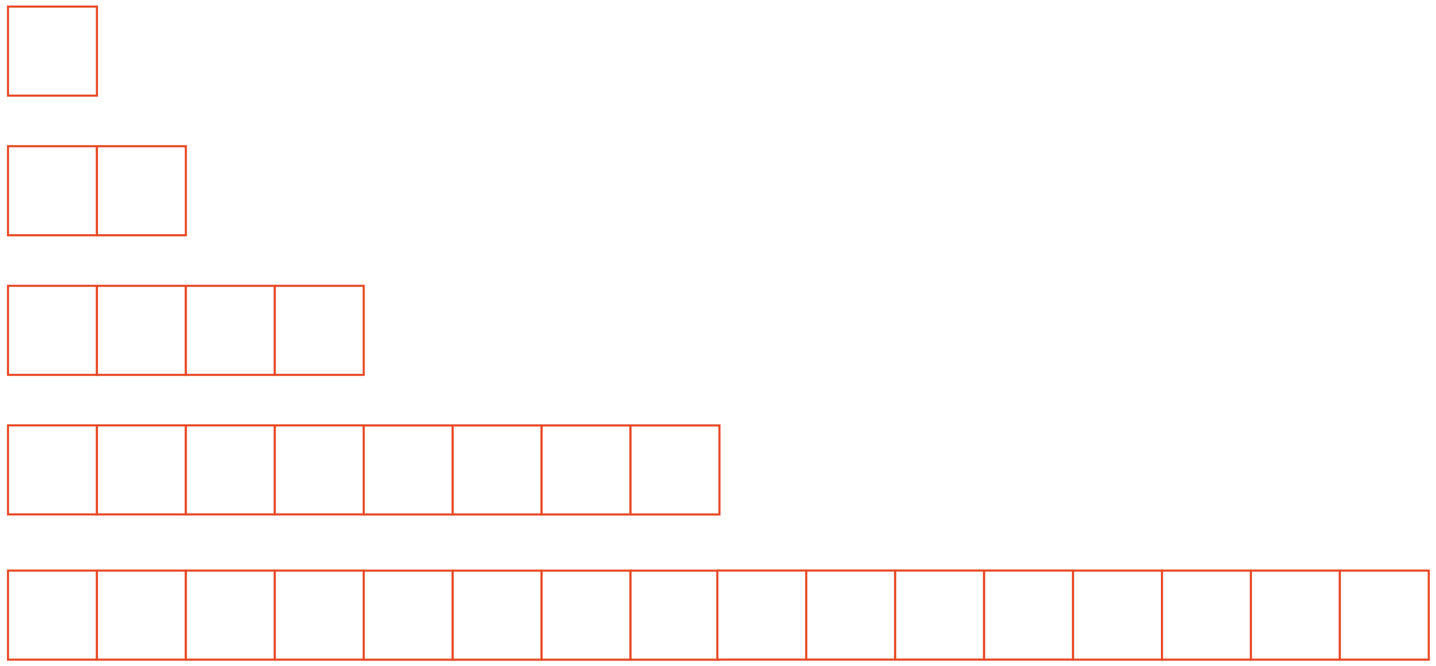
C	**S**	**2**	**2**	**5**
[0]	[1]	[2]	[3]	[4]

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 **front**		
Insert at **given** element		
Remove at **given** element		
Insert at **arbitrary** location		
Remove at **arbitrary** location		

Queue ADT

- [Order]:

- [Implementation]:

- [Runtime]:

Stack ADT

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

Queue.h

```
#pragma once

template <typename T>
class Queue {
  public:
    void enqueue(T e);
    T dequeue();
    bool isEmpty();

  private:
    T *items_;
    unsigned capacity_;
    unsigned size_;
};
```

What type of implementation is this Queue?

How is the data stored on this Queue?

Queue.h

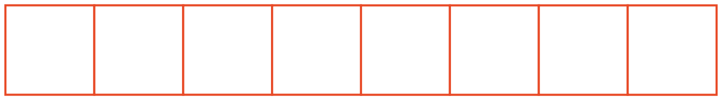
```
#pragma once

template <typename T>
class Queue {
  public:
    void enqueue(T e);
    T dequeue();
    bool isEmpty();

  private:
    T *items_;
    unsigned capacity_;
    unsigned size_;
};
```

What type of implementation is this Queue?

How is the data stored on this Queue?



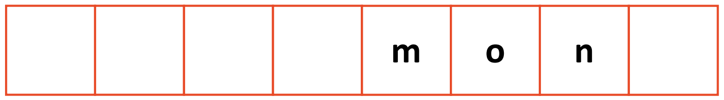
```
Queue<int> q;
q.enqueue(3);
q.enqueue(8);
q.enqueue(4);
q.dequeue();
q.enqueue(7);
q.dequeue();
q.dequeue();
q.enqueue(2);
q.enqueue(1);
q.enqueue(3);
q.enqueue(5);
q.dequeue();
q.enqueue(9);
```

Queue.h

```cpp
#pragma once

template <typename T>
class Queue {
  public:
    void enqueue(T e);
    T dequeue();
    bool isEmpty();

  private:
    T *items_;
    unsigned capacity_;
    unsigned size_;
};
```



```
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);
```