CS 225

Data Structures

January 30 – List <vector>

G Carl Evans

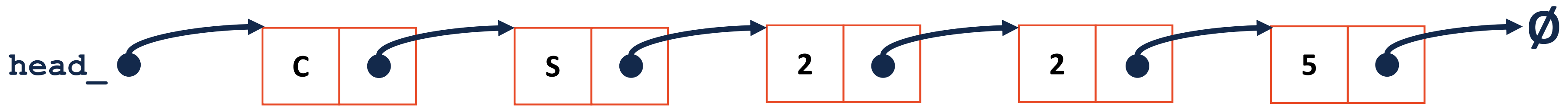
Honors Starts Today

- 1214 Siebel Center at 5pm

Lecture Code Repo

https://github.com/cs225-illinois/lecture-sp23.git

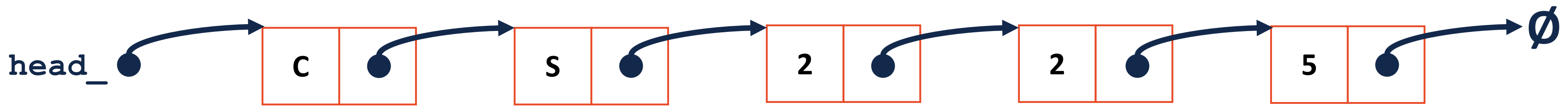
Linked Memory: `operator[]`



List.hpp

```
template <typename T>
T & List<T>::operator[](unsigned index) {
…






}
```

Linked Memory: `remove`



List.hpp

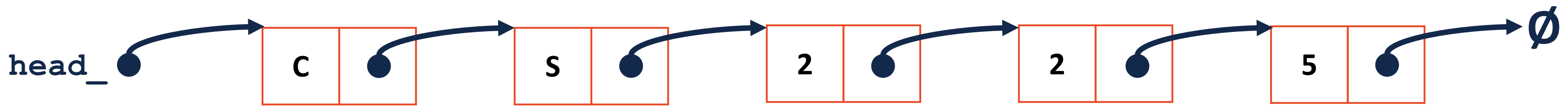
```
template <typename T>
void List<T>::remove(unsigned index) {







}
```

List.hpp

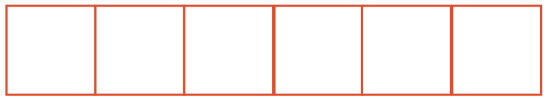
```
template <typename T>
T List<T>::remove(unsigned index) {
  ListNode *& node = _index(index);
  return _remove(node);
}
```

```
template <typename T>
T List<T>::_remove(ListNode *& node) {
  ListNode * temp = node;
  node = node->next;

  T data = temp->data;
  delete temp;

  return data;
}
```

Linked Memory Runtimes

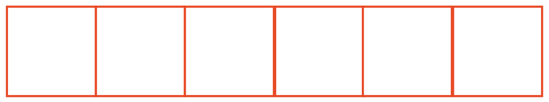


Array Implementation



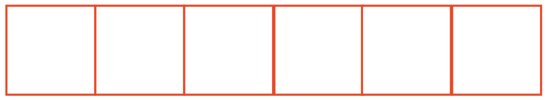
List.h

```
#pragma once

template <typename T>
class List {
public:
    /* --- */
private:






};
```



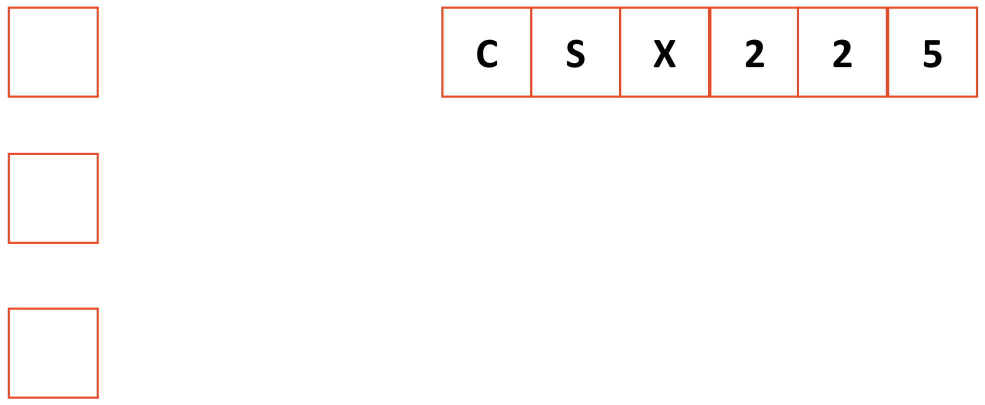
Array Implementation

_addspace:



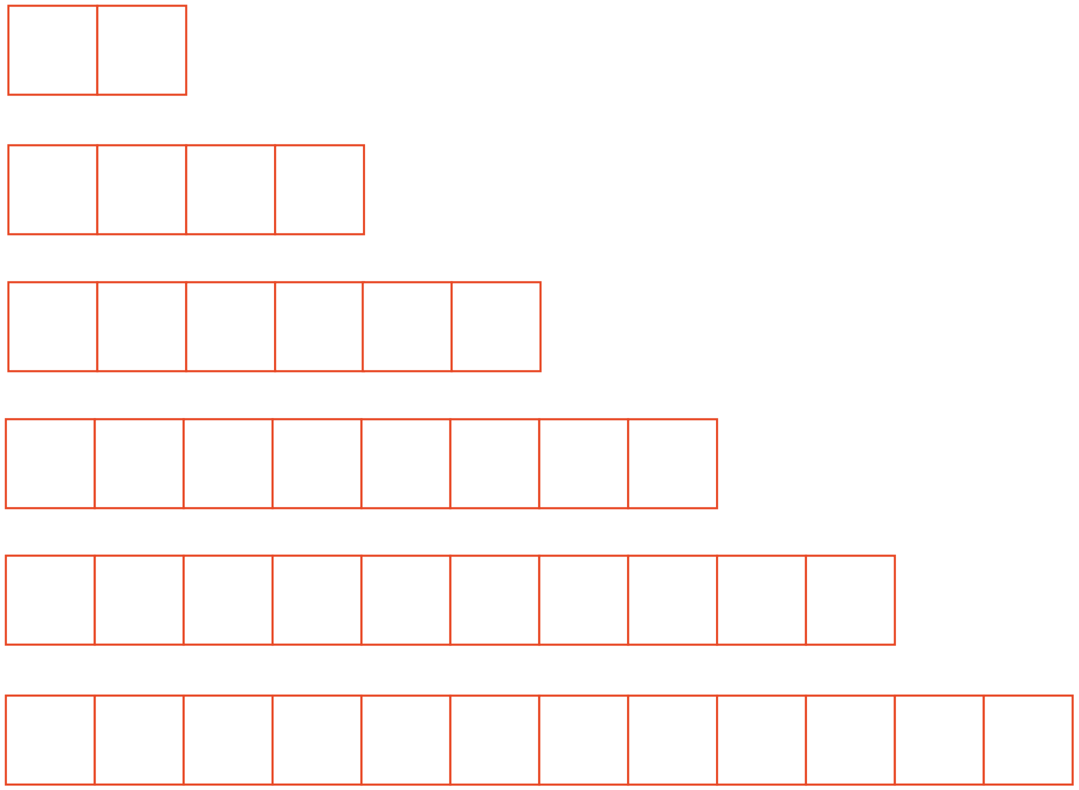
Array Implementation

_addspace():



Amortized Analysis

Resize Strategy: +2 elements every time



Resize Strategy: +2 elements every time

Resize Strategy: Can We Do Better

Queue ADT

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

Stack ADT

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

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_;
};
```

What type of implementation is this Queue?

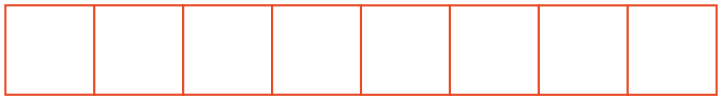
How is the data stored on this Queue?

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?

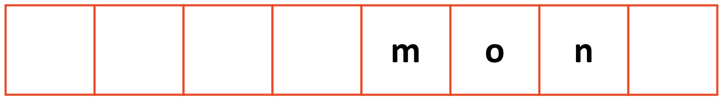
How is the data stored on this Queue?



```
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

```
#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);
```