CSCI 2100

Vectors

Announcements

- Test on Tuesday
  Review Monday
- HW due today
- Lab due today
- Next HW: cover Vectors,
  posted mid-next week

Last time: Queues

Simple data structure

limited access, fast

$O(1)$ time for everything

Today: Vectors

Similar to Lists in Python
(& we saw them in lab this week)

Our implementation:
- array based

Main functions –
see STL, because there are a' lot of them!

private: unsigned
int size;
int capacity;
Object* data;

data = new Object[capacity];
data[i]

To think through:

Vector<char> myvec;

added data

element

```
myvec.insert(2, 'c');
```

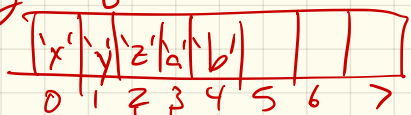
index

How to insert, if we don't want to lose data?

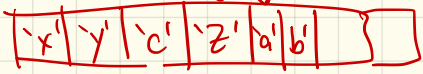
myvec: size = 5

capacity = 8

data:



goal →



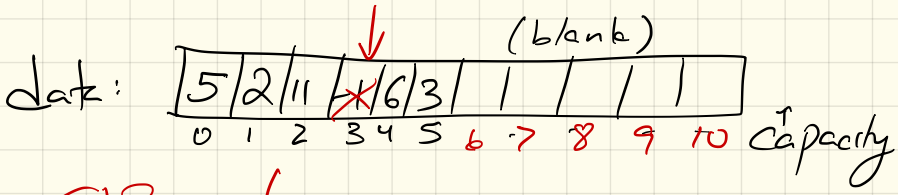

```
for (int i = size; i > index; i--)
    data[i] = data[i-1];
data[index] = element;
```

(throw error if ~~index~~ >= size)

Similarly, erase:

```
myvec.erase(3);
```

Underneath:



size = 6

capacity = 11

```
cout << myvec[3]
```

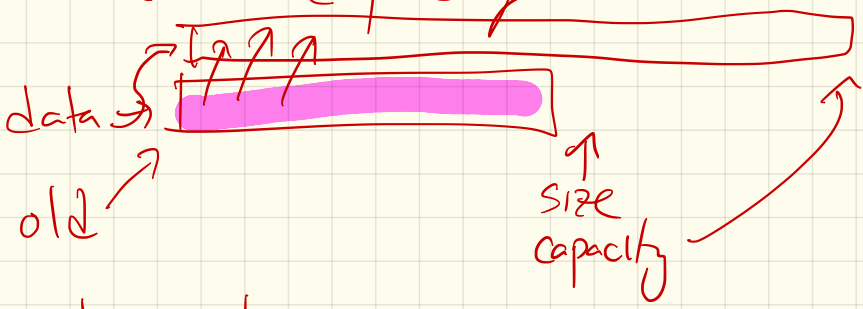
↑ 6 here

throw error if i >= size

```
for(int i=index; i<size-1; i++)
    data[i]=data[i+1];
size--;
```

Another issue:
- what if what if we exceed the capacity?

Suppose in insert,
- size == capacity:




```
//create a bigger array
Object * old = _data;
capacity *= 2;
_data = new Object[capacity];
//copy data over
for(int i=0; i<size; i++)
    data[i] = old[i];

//delete old array
delete[] old;

//continue w/ insert
```

Finally, don't forget housekeeping!

- already saw destructor
- copy cons + operator =

  need to allocate a new array + copy data