CS2100

More Vectors

Recap

- HW due via git, Saturday (partner allowed)
- Lab this week (over Vectors)
  no prelab
  (due by Sunday)
- Midterm 1 next week
  review on Monday
  exam on Wed.
  no lab next Thursday
- Next HW- due after midterm, over vectors

Last lecture

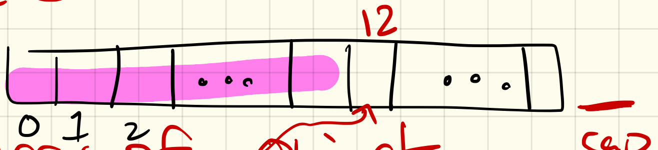
Vectors:

- basically a more robust array
- incorporates list-like functionality
- makes room automatically if it gets full

Picture:

Private data:

_data:



← array of objects

_size = one # larger than last entry in array

_capacity → max possible size

Today:

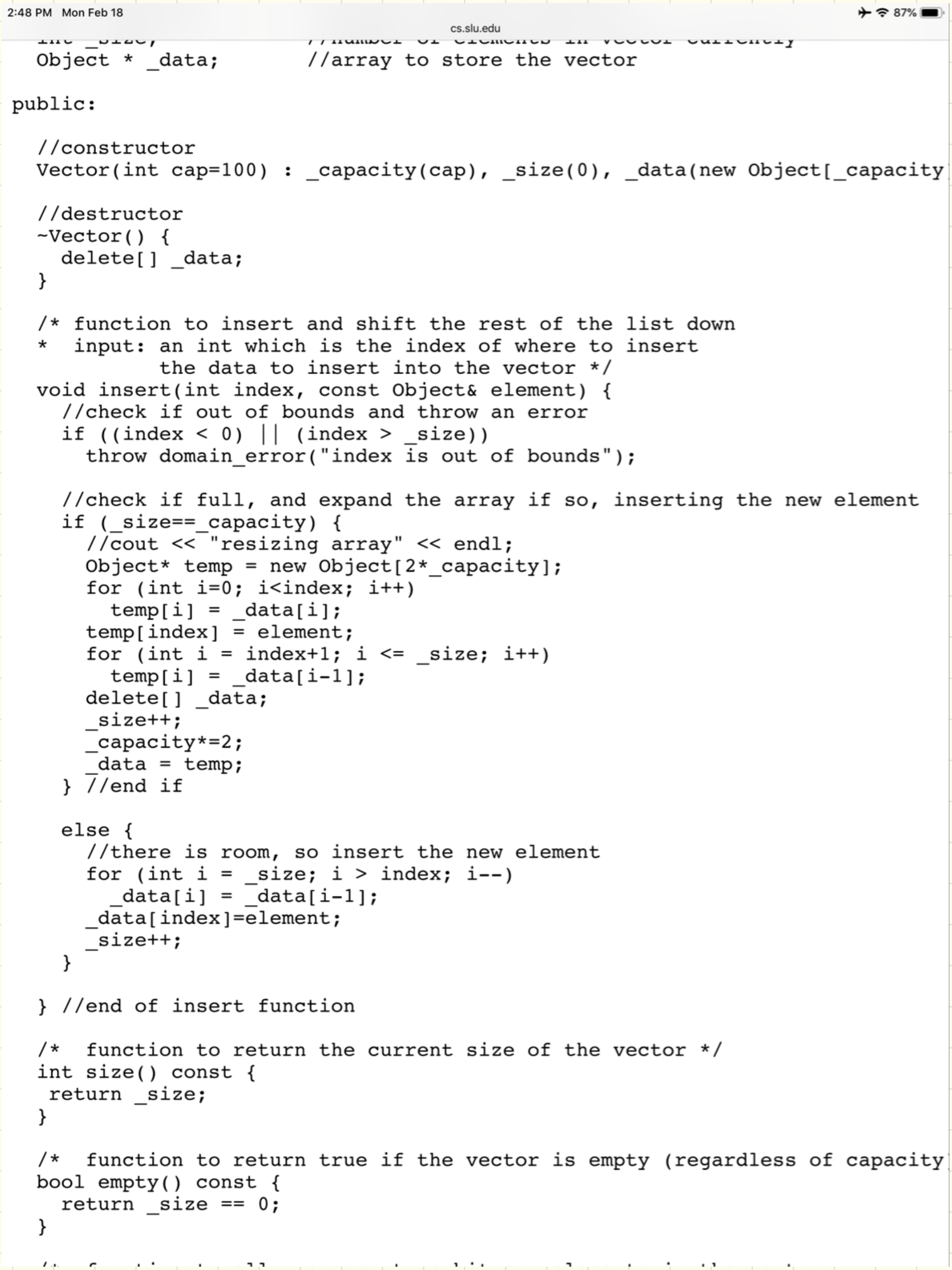
First, pen & paper coding to get started.

(~10 or 15 minutes)

If you finish on paper, try to implement!

2 cases:

- full : double capacity
- not full : move everything down one



```
  int _size;              //number of elements in vector currently
  Object * _data;         //array to store the vector

public:

  //constructor
  Vector(int cap=100) : _capacity(cap), _size(0), _data(new Object[_capacity

  //destructor
  ~Vector() {
    delete[] _data;
  }

  /* function to insert and shift the rest of the list down
  *  input: an int which is the index of where to insert
            the data to insert into the vector */
  void insert(int index, const Object& element) {
    //check if out of bounds and throw an error
    if ((index < 0) || (index > _size))
      throw domain_error("index is out of bounds");

    //check if full, and expand the array if so, inserting the new element
    if (_size==_capacity) {
      //cout << "resizing array" << endl;
      Object* temp = new Object[2*_capacity];
      for (int i=0; i<index; i++)
        temp[i] = _data[i];
      temp[index] = element;
      for (int i = index+1; i <= _size; i++)
        temp[i] = _data[i-1];
      delete[] _data;
      _size++;
      _capacity*=2;
      _data = temp;
    } //end if

    else {
      //there is room, so insert the new element
      for (int i = _size; i > index; i--)
        _data[i] = _data[i-1];
      _data[index]=element;
      _size++;
    }

  } //end of insert function

  /*  function to return the current size of the vector */
  int size() const {
   return _size;
  }

  /*  function to return true if the vector is empty (regardless of capacity
  bool empty() const {
    return _size == 0;
  }

  [illegible]
```

Then:

Still lots to code!

push_front + push_back: easy! (use insert)

erase(index i): (on HW)

pop_front + pop_back:

Housekeeping:

(Go over code - will post after class)

Next time:

Finish code

talk about big-O

insert: $O(n)$

amotized analysis:

push_back

if full: $O(n)$

if not: $O(1)$

"average" run time: $O(1)$