# CS 180 - Lecture 9

Note Title 9/13/2009

## Announcements

- Program 1 - due Friday
- Midterm 1 will be Thurs., Sept. 24 in class
  - Review session will be Monday, & lab will be Wednesday the 23rd
- HW3 will come out this week.

Stacks: list of objects supporting the following operations: (LIFO)

- push(o): insert o at top of stack
  Input: object o  Output: None
- pop(): remove & return top object on stack
  Input: none  Output: Object

- size(): Return # of objects in stack
  Input: none  Output: Integer
- isEmpty(): Return boolean indicating if stack is empty
  Input: none  Output: bool
- top(): Return top object on stack without removing it
  Input: none  Output: Object

# Standard Template Library

Stacks are one of the built in class in the STL.

Functions: push, pop, top, size, & empty

Documentation is available online.

(We'll use this for lab soon...)

Our interface:

Always worth planning the functions ahead of time...

```
template <typename Object>
class Stack {
  void push(const Object& obj);   //mutator function
                                  // no const
  bool isEmpty() const;

  const Object& top() const;

  Object pop();

  int size() const;
};
```

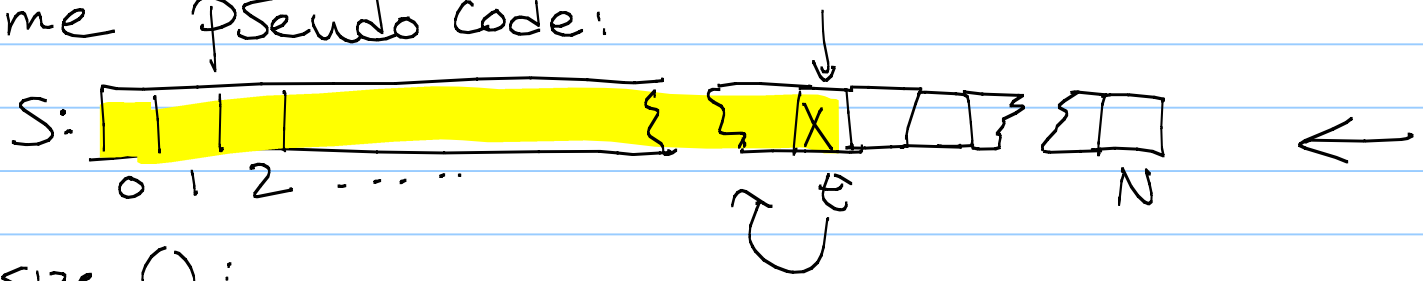
on an empty stack →

One complication: how should we return objects?

Should pop & top be different?

(see return types)

Some pseudo code:




```
size():
    return t+1
isEmpty():
    return (t<0)
top():
    if isEmpty()
        Throw "Stack empty error"
    return S[t]
pop():
    if isEmpty()
        throw Stack error
```

```
o ← S[t]
t ← t-1
return o
```

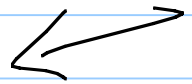
Our code: available on webpage

Based on code from text (p. 163)

(with a few changes).

Running Times:

| Function | Time |
|---|---|
| size | O(1) |
| isEmpty | O(1) |
| top | O(1) |
| push | O(1) |
| pop | O(1) |



Size usage? O(N)
(assuming Object is size O(1))

What is the main disadvantage?

predefined maximum size

other option: double array every time stack overflows
spend O(N) each time stack doubles, since we need to recopy

# Sec. 4.2.3 - Function calls + stacks

C++ actually keeps a private stack, called the run-time stack, to keep track of local variables.

Why is this data structure ideal?

(see p. 166-168 for more detail.)