CS180 - Doubly Linked Lists + Stacks

Announcements

- Lab tomorrow
- Program will be posted in next 1-2 days
- First midterm in ~3 weeks.
- Book is available
Doubly Linked Lists

- \text{insert Back}
- \text{delete Back}

(see code on webpage)
Stack: a way to store a list of data

Ex: Web browser: Store history
Hit back, goes to most recent page

Ex: Text editors: Store previously executed commands
Undo will revert to most recent "history"
The Stack Abstract Data Type (ADT)

Supports 2 main functions:
- `push(o)`
  Insert object o at top of stack
- `pop()`
  Remove top object from stack and return it

```
stack = []
push(5)
push(11)
push(1)
push(2)
pop()
pop()
pop()
push(2)
```
Additional behaviors

- `size()` : Return # of objects in the stack

- `isEmpty()` : Returns true if stack is empty, false otherwise

- `top()` : Returns top object on stack without removing it
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...)
Class Stack &
private:

Notice:
I haven’t said what this is
made with!

Ideas?
- Array-based stack
- Linked list stack
One complication: how should we return objects?

Should `pop` and `top` be different?

STL does not return a popped element.

STL wins — `pop` will be void.
Our interface

Private variables:

private:

Object * - S;
int - capacity;
int - size;

(array) \rightarrow t

capacity

\[ \begin{array}{c}
0 & 1 & 2 & 3 & 4 \\
5 & 1 & 1 & - & 0 \\
\end{array} \]
Now - what is left??

- Constructor
- Destructor
- Copy Constructor
- Operator = \[ \text{deep copies} \]
Another way

Linked Stacks

(We can use SLinkedList class!)