CS 2100: Data Structures

Stacks

Recap

Bug in HW3!

operator= → copy constructor
(messed up tail)

redownload oh
(or copy those 2 functions from zyBook to yours)

- HW3 due Saturday
- Reading assigned for Wed. & Friday
- Lab Thurs.

Today: Stacks: a way to store a list

Key: Extremely limited ways to access the data.

However, despite simplicity, surprisingly useful! (very very fast)

Ex: Previously visited web pages

Ex: Previous changes to a word document

Ex: Program execution

Common behavior:

store most recent thing (can't go deeper in list)

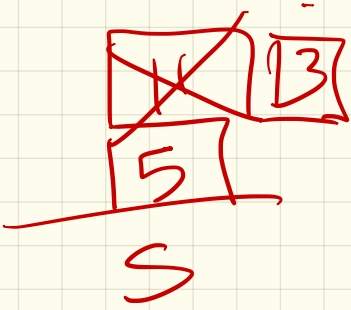
The stack ADT:

- push(e)
  - add to "top"
- pop():
  - remove from "top"
  - (no return value)

Also:

- size()
- empty()
- top() ← returns "top" element

top




```
push(5)
push(11)
pop()
push(3)
cout << top()
```

Example: (from a main)

```
#include <stack>
int main() {
    stack<int> mystack;
    for (int i=10; i<20; i+=2)
        mystack.push(i);
    mystack.pop();
    mystack.push(100);
    cout << mystack.top() << endl;
}
```

See cplusplus.com for lab this week on stacks.

Today, we'll code our own!

Implementation:

How should we store our data?

Essentially, what should our private class variables be?

```
private:
    SLinkedList S;
```

```
void push(T element) {
    S.addFront(element);
}
void pop() {
    S.removeFront();
}
T top() {
    return S.front();
}
```

Two versions:
(both on webpage)

Other: use an array!

private:

~~Object~~ * S;

// size + capacity

(see ArrayStack.h)

Runtimes:

either version:

push, pop, top, size and empty: O(1)

operator= & copy constructor: O(n)

(either copy array or duplicate nodes)

destructor:

array: O(1) - in C++

linked: O(n)

(but arrays had max size)