Stacks
Lecture 24
Section 7.1 - 7.3

Robb T. Koether

Hampden-Sydney College

Fri, Mar 20, 2009
Outline

1. Stacks
2. The Stack Interface
3. Assignment
Definition (Stack)

A stack is a list that operates under the principle “last in, first out” (LIFO). New elements are pushed onto the stack. Old elements are popped off the stack.

To enforce the LIFO principle, we use a list and push and pop at the same end.
Stack Constructors

- `Stack();`
  Constructs an empty stack.

- `Stack(const Stack& s);`
  Constructs a copy of a specified stack.
Stack Inspectors

- **T top() const;**
  Gets a copy of the element at the top of the stack (but does not remove it).

- **int size() const;**
  Gets the number of elements in the stack.

- **bool isEmpty() const;**
  Determines whether the stack is empty.
Stack Mutators

- **void push(const T& value);**
  Pushes the specified value onto the top of the stack.

- **T pop();**
  Pops and returns the element off the top of the stack.

- **void makeEmpty();**
  Makes the stack empty.
Other Stack Member Functions

**bool isValid() const;**

Determines whether the stack has a valid structure.
Other Stack Functions

- `istream& operator>>(istream& in, Stack& s);`
  Reads a Stack object from the input stream.

- `ostream& operator<<(ostream& out, const Stack& s);`
  Writes a Stack object to the output stream.
implementation of Stacks

Which is more accurate?

- A stack *is* a list.
- A stack *has* a list.
Things will be simpler if we will say that a stack has a list.

Which List implementation should we use?
Which push and pop functions should we use?

pushFront() and popFront(), or
pushBack() and popBack().

Choose a List class for which pushing and popping at one end will be efficient.
One must be careful when reading a stack.

As the values are read, should they be pushed on the front end or the back end of the stack?

What does the List class input facilitator do?

Do we need to write new `input()` and `output()` functions?
Stack Implementation

- Download `arraystack.h`.
- `arraylist.h`.
- `linkedstack.h`.
- `linkedlist.h`.
- `linkedlistnode.h`.
- `StackTest.cpp`.
Assignment

Homework

- Read Section 7.1, pages 315 - 321.
- Read Section 7.2, pages 321 - 349.
- Read Section 7.3, pages 353 - 365.