The Stack ADT

Monday, April 21, 2025 9:39 AM

LIFO – last in, first out

The most recently pushed (added/inserted) element is the first removed/popped

Data: a collection of items

Like a List!

Operations:

- push(item)
  - Put item on the top of the stack
- pop()
  - Pulls the item on the top of the stack off (removes that item from our stack)
  - In some implementations, pop() returns the item that was removed, other implementations it's a void function
- top()/peek()
  - Returns the item on the top of the stack (what would be removed next if pop() were called) -- don't actually remove the element
- size()
- empty()

Implementing Stacks

Monday, April 21, 2025 9:48 AM

- We can write a Stack class **composed** of a List class
  - Each stack has-a (contains) a List object (as a data member)
  - This List is private – clients can only access it in ways we define
  - Insert and remove on only one end of our List

Using Stacks

Monday, April 21, 2025 9:47 AM

- When functions call other functions (call stack, stack memory)
- Undo/redo -- back/forward buttons
- Easy to reverse a string/list/sequence
- Check balance of parentheses in an expression:
    - Initialize a Stack (let's call it stack), initially empty
    - Read in one character in the expression at a time
    - For each character:
        - If it's '(':
            - Push '(' onto the stack
        - If it's ')':
            - If the stack isn't empty: op the top element off the stack
            - If the stack is empty: we know this is not balanced!
        - If it's neither:
            - Ignore it!
    - At the end, if the stack is empty: it's balanced!
    - Otherwise, it's not balanced!

The Queue ADT

Wednesday, April 23, 2025 8:59 AM

FIFO – first in, first out

The elements wait patiently in line, the first one that enters the queue is the first taken off of it

Data: a collection of items

Like a List!

Operations:

- push(item)/enqueue(item)
  - Insert item at the back of the queue
- pop()/dequeue()
  - Remove the item at the front of the queue
- front()/peek()
  - Return the item at the front of the queue without actually removing it
- size()
- empty()

Implementing Queues

Friday, April 25, 2025 10:54 AM

- Like with Stacks:
  - We can write a Queue class **composed** of a List class
    - Each queue has-a (contains) a List object (as a data member)
    - This List is private – clients can only access it in ways we define
    - Insert at end and remove from the beginning
- For a linked list:
  - Make more efficient with a tail pointer
- For an array:
  - Make more efficient with a circular array

Standard Template Library

Friday, April 25, 2025 11:52 AM

- Contains class definitions for a lot of types of templated container classes
- vector
  - Array-based List
- list
  - Linked list-based List
- stack
- queue
- priority queue

The Priority Queue ADT

Monday, April 28, 2025 9:17 AM

Data: a collection of items

Like a List!

Operations:

- push(item)/enqueue(item)
  - Insert item into the queue
  - Position determined by some priority ordering
  - Elements with the same priority follow FIFO order
- pop()/dequeue()
  - Remove the item at the front of the queue (highest priority item)
- front()/peek()
  - Return the item at the front of the queue without actually removing it
- size()
- empty()

Composition vs. Inheritance

Monday, April 28, 2025 9:48 AM

- Composition of classes:
  - One class contains an instance of another class
  - Has-a relationship
  - A Stack object has-a List object
  - A Queue object has-a List object
- Inheritance:
  - One class (derived/child class) is derived from another class (base/parent class)
    - A Square class (derived/child class) might be derived from a Rectangle class (base/parent class)
  - Is-a relationship
  - Our derived/child class inherits all **public** members of the base/parent class
    - We don't get direct access to private members
    - We can extend the base class so that our derived class has specific methods or data the base class does not
    - We can **override** methods inherited from the parent class – provide a different definition of a method for objects of our child class

Implementing a Priority Queue

Monday, April 28, 2025 9:25 AM

- Define a SortedList class that inherits from List
  - Only inserts items in a sorted order (descending order)
- Our PriorityQueue class is composed of (contains) a SortedList object
  - Like how Queue class was composed of (contained) a List object