Notes 1/21/2011

What is a stack? The first and probably easiest ADT (abstract data type.)

A stack goes by the acronym FILO: first in, last out. The first thing you stick into the stack is the last thing to come out – like a stack.

The stack has two major operations – “push” and “pop.” Push puts an item in and pushes all the other items down the stack, pop takes an item out and all the other items move up. It also has a couple other fairly common functions – isEmpty and top.

3  
4  
5  
6  
  
Stacktop = 4.

If you call pop(), it returns 6 and now the array reads

3  
4  
5  
  
Stacktop = 3.

Fairly simple and basic. isEmpty() checks the stacktop variable to see if it is 0, top deletes them all and sets the variable to 0, etc.

How about doing this with a linked list?

Class stack  
{  
 node \*head;  
 void push(node \*a);  
 node \*pop();  
 bool isEmpty();  
 node \*top();  
 stack();  
}  
  
Stack::Stack()  
{  
 head = NULL;  
}  
  
void Stack::push(node \*a)  
{  
 a->next = head;  
 head = a;  
}  
  
node\* Stack::pop()  
{  
 node\* temp = head;  
 if (head)  
 {  
 head = head -> next;  
 return temp;  
 }  
}  
  
bool Stack::isEmpty()  
{  
 return head == NULL;  
}  
  
Node\* Stack::top()  
{  
 return head;  
}

Stacks are really useful and you will use them forever. However there is another ADT called a queue that is also useful. It’s like a stack but follows FIFO instead – first in, first out.

The easiest way to implement this is a cyclical structure; run element 1, then put it on the end. Run element 2, then put it on the end. Etc.

Queues have the same basic functions like push and pop. They are very cumbersome to do with an array, however – you need a pointer to the last element, inserting elements at the end and moving pointers around, etc.

With a linked list this is a much more manageable problem.   
  
Stack::push()  
{  
 tail -> next = a; //remember you are pushing on the *back* of the list.  
 a->next = NULL;  
 tail = a;  
}