Lecture # 5

Abstract Data Type

- We have looked at four different implementations of the List data structures:
  - Using arrays
  - Singly linked list
  - Doubly linked list
  - Circular linked list.
- The interface to the List stayed the same, i.e., add(), get(), next(), start(), remove() etc.
- The list is thus an abstract data type; we use it without being concerned with how it is implemented.

Abstract Data Type

- What we care about is the methods that are available for use with the List ADT.
- We will follow this theme when we develop other ADT.
- We will publish the interface and keep the freedom to change the implementation of ADT without effecting users of the ADT.

- The C++ classes provide us the ability to create such ADTs.

Stack

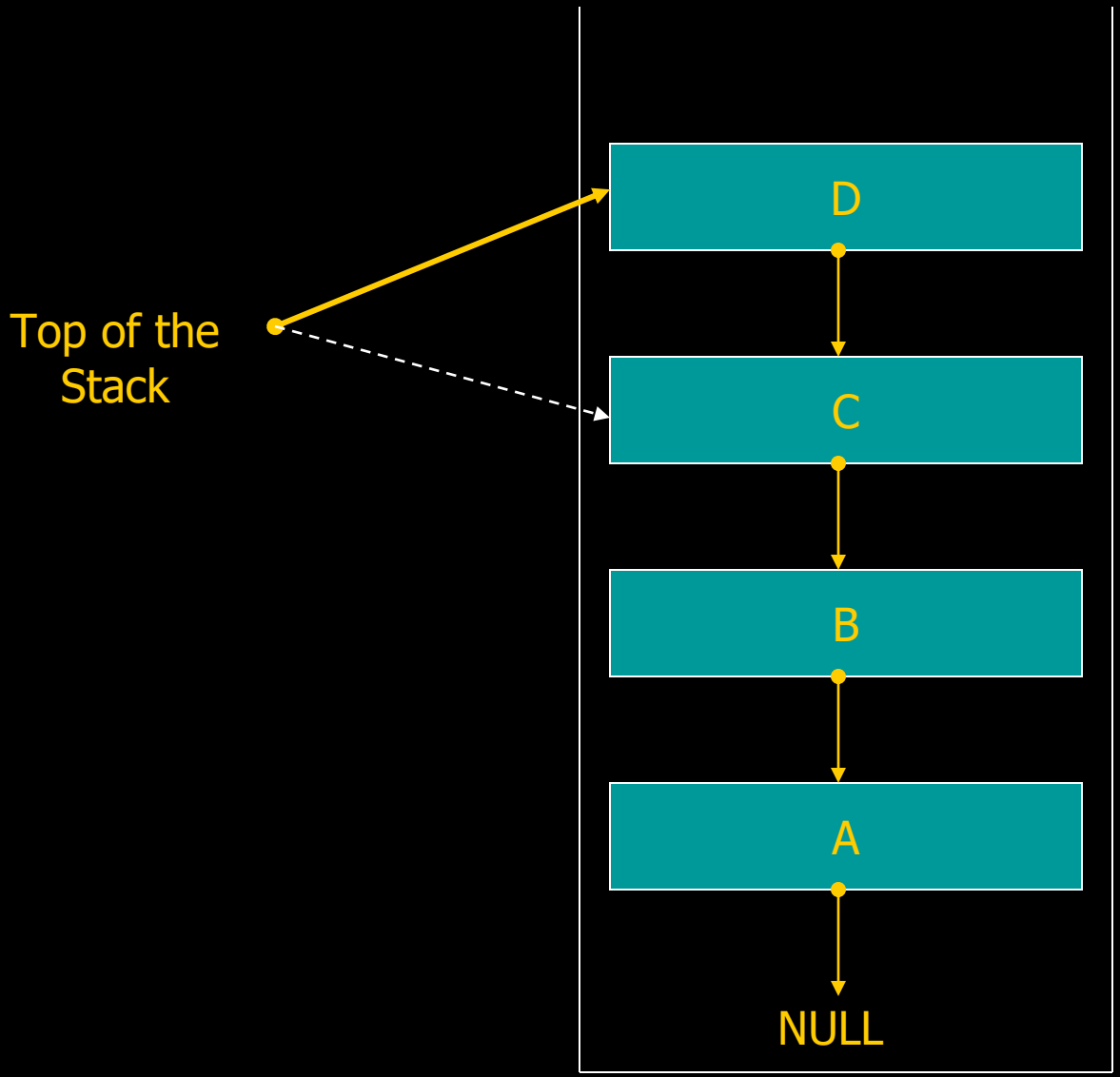
Stack

- Stacks in real life: stack of books, stack of plates
- Add new items at the top
- Remove an item at the top
- Stack data structure similar to real life: collection of elements arranged in a linear order.
- Can only access element at the top

- A stack is an ordered collection of items into which new items may be inserted and from which items may be deleted at one end only, called **top** of the stack. i.e. deletion/insertion can only be done from **top** of the stack.

- The insert operation on a stack is often called **Push** operation and delete operation is called **Pop** operation.

- Figure showing stack…..



- In a stack, the element deleted from the list is the one most recently inserted.
- Stack is also referred as Last-in First-out i.e LIFO.
- The example of stack is the stacks of plates used in cafeterias. The order in which plates are popped from the stack is the reverse of the order in which they were pushed onto the stack, since only the top plate is accessible.

- If an empty stack is popped, we say stack underflows, which is normally an error. and

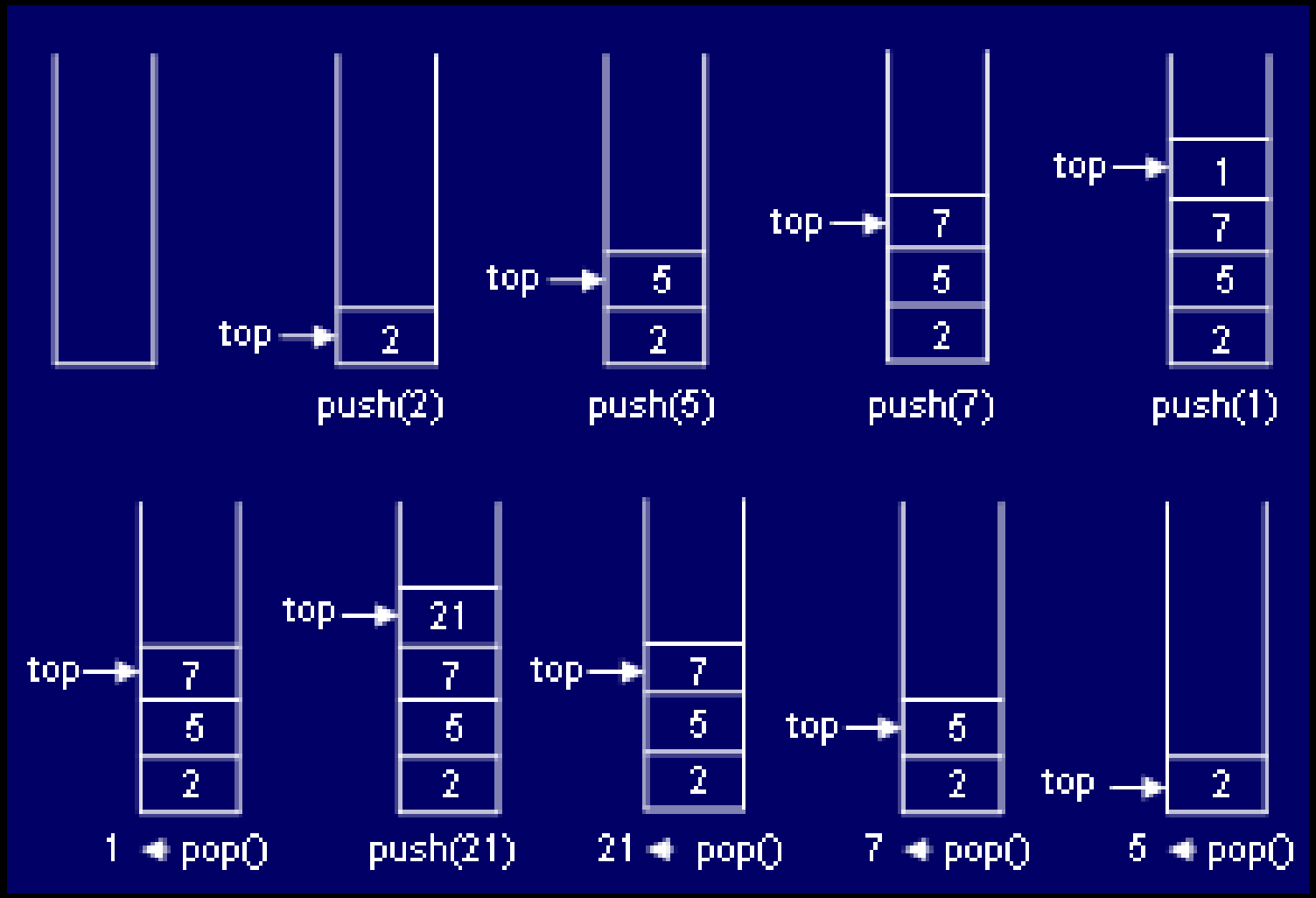
- If top of the stack exceeds n, the stack overflows.

Stack Operations

- Push(X) – insert X as the top element of the stack
- Pop() – remove the top element of the stack and return it.
- Top() – return the top element without removing it from the stack.

Stack Operation

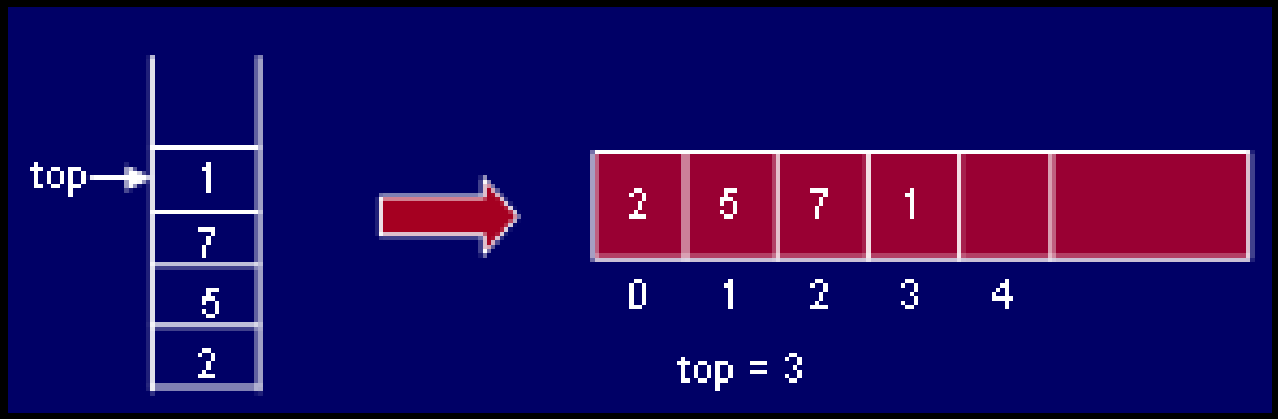
- The last element to go into the stack is the first to come out: *LIFO* – Last In First Out.
- What happens if we call pop() and there is no element?
- Have IsEmpty() boolean function that returns true if stack is empty, false otherwise.
- Throw StackEmpty exception: advanced C++ concept.


top
2
push(2)
top
5
2
push(5)
top
7
5
2
push(7)
top
1
7
5
2
push(1)
top
7
5
2
1
pop()
top
21
7
5
2
push(21)
top
7
5
2
21
pop()
top
5
2
7
pop()
top
2
5
pop()

Stack Implementation: Array

- Worst case for insertion and deletion from an array when insert and delete from the beginning: *shift elements to the left*.
- Best case for insert and delete is at the end of the array – *no need to shift any elements*.
- Implement push() and pop() by inserting and deleting at the end of an array

Stack Implementation: Array

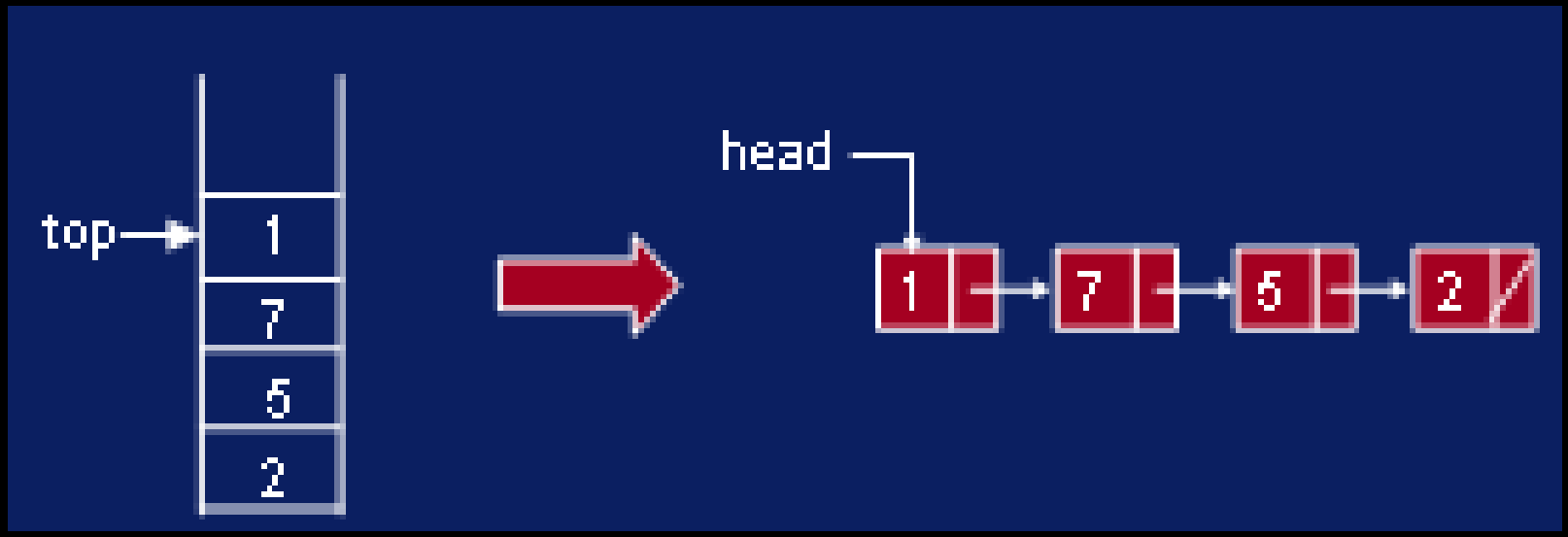


Stack using an Array

- In case of an array, it is possible that the array may "fill-up" if we push enough elements.
- Have a boolean function **IsFull()** which returns true if stack (array) is full, false otherwise.
- We would call this function before calling push(x).

Stack Using Linked List

- We can avoid the size limitation of a stack implemented with an array by using a linked list to hold the stack elements.
- As with array, however, we need to decide where to insert elements in the list and where to delete them so that push and pop will run the fastest.


top
1
7
5
2
head
1
7
5
2