August 31, 2015

Launchpad

Lecture - 15

Linked List

Anushray Gupta

Status of Assignment

Any doubts?

TA Sessions

What are data structures?

In computer science, a data structure is a particular way of organizing data in a computer so that it can be used efficiently. Different kinds of data structures are suited to different kinds of applications, and some are highly specialized to specific task.

Most used data structure is The List!

Important Operations on a List!

I. Insert Element
   I. In the beginning
   II. At the end
   III. After element X
   IV. At position K
II. Find element
III. Delete Element
   I. From the beginning
   II. From the end
   III. Before/After X
   IV. At Position K
IV. Get Element
   I. At the beginning
   II. At the end
   III. At position K
   IV. After/Before X

Array can be used to implement List!

Lets see cost of these operations in the list if it were to be implemented using Array.

Array as List!

I. Insert Element
   I. In the beginning – O(N)
   II. At the end – O(1)
   III. After element X – O(N)
   IV. At position K – O(N)
II. Find element – O(N)
III. Delete Element
   I. From the beginning – O(N)
   II. From the end – O(1)
   III. Before/After X – O(N)
   IV. At Position K – O(N)
IV. Get Element
   I. At the beginning – O(1)
   II. At the end – O(1)
   III. At position K – O(1)
   IV. After/Before X – O(1)

Advantages of using Array as List

I. Accessing Element is Faster – This is only because its contiguous allocation.
[BaseAddress + sizeof(type) * i]
II. Insertion and Deletion from end of list is faster.

Disadvantages of using Array as List

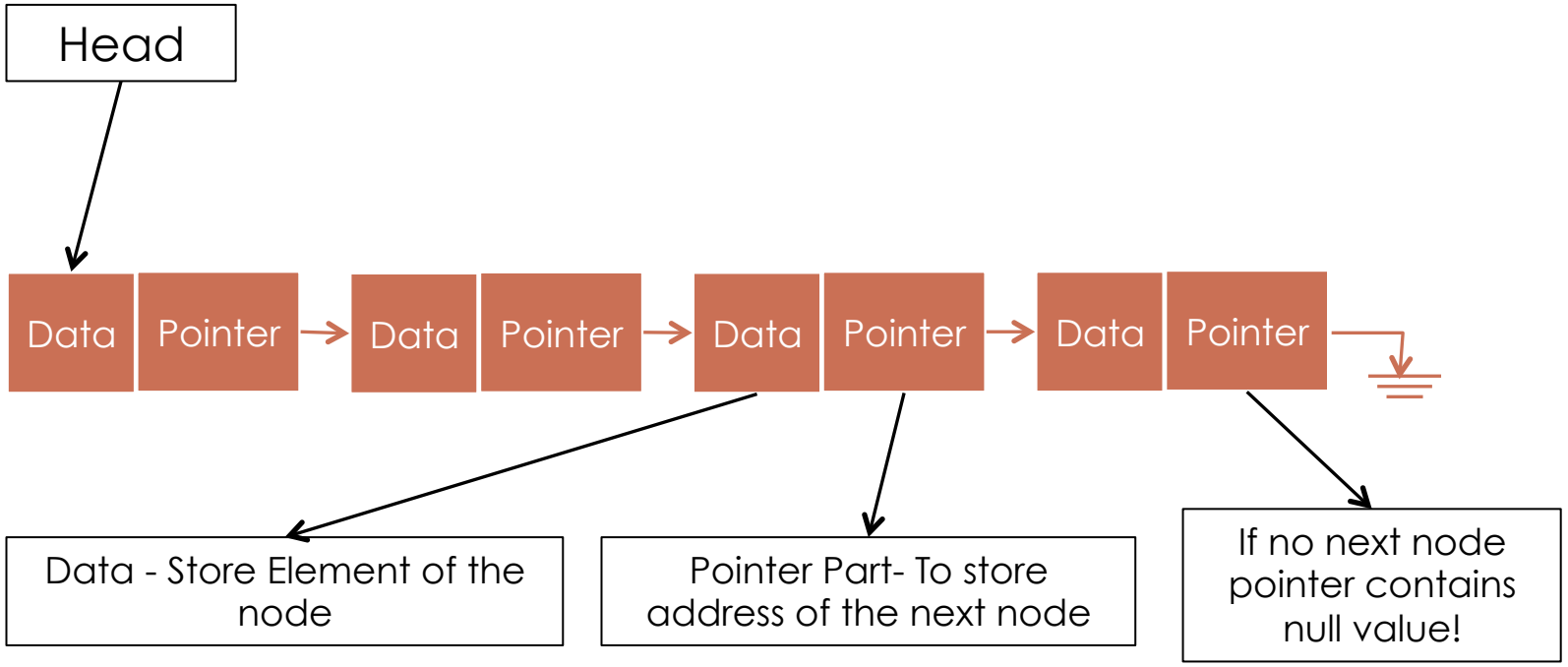
I. Fixed Size – This can lead to either unused memory or insufficient memory
II. Insertion and Deletion from between is expensive as this requires shifting of elements.

To minimize insertions and deletion we need to make sure items are not store continuously.

Linked List

I. Linked List is a Linear collection of self-referential structures called as “nodes”.
II. Each node has minimum two parts
    I. Data
    II. Pointer to next node called as “next” pointer

Linked List



CODING BLOCKS

How to represent a node?

```
struct Node {
        int data;
        Node * next;
};
```

How to represent a Linked List?

As seen before, a linked list is a linear collection of dynamically created self-referential nodes.

So what do we need to represent a Linked List? – Just address of the first Node which is conventionally called as head.

Node * head.

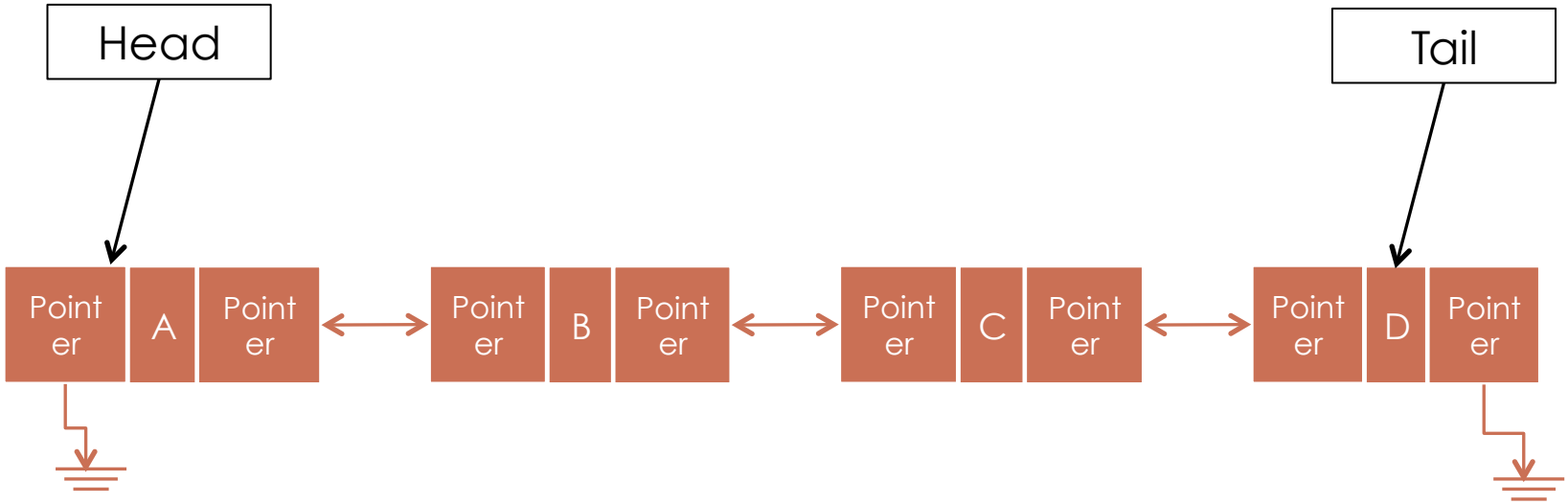
Lets see some operations on SLL.

I. Insert at the beginning
II. Printing the Linked List.
III. Reverse Print.
IV. Delete from the end.

Time to try?

I. Delete from the beginning
II. Insert at The end
III. Find Kth Element
IV. Delete at Kth
V. Insert at Kth
VI. Find Midpoint
VII. Swap i^{th} and j^{th} elements of the linked list

Doubly Linked List



CODING BLOCKS

Implementation?

```
struct node {
        int data;
        node * next;
        node * prev;
};
```

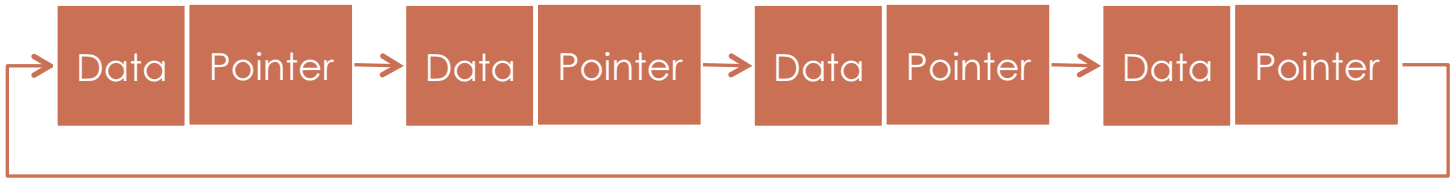
Doubly LL vs Singly LL

I. Faster to go back in the linked list
II. Uses more memory

Common Mistakes

I. Using variables which are uninitialized and pointing to garbage addresses!
II. When and when not to use “new” operator to get a new node.
III. Always delete memory for unused node using “delete” operator
IV. Make sure your pointers are pointing to right node after any modifications you make in the list
V. Keep your head safe and sane!

Circular Linked List



Lets solve few more problems?

I. Insertion Sort on Linked List
II. Sort a list which has two halves sorted.

Time to try?

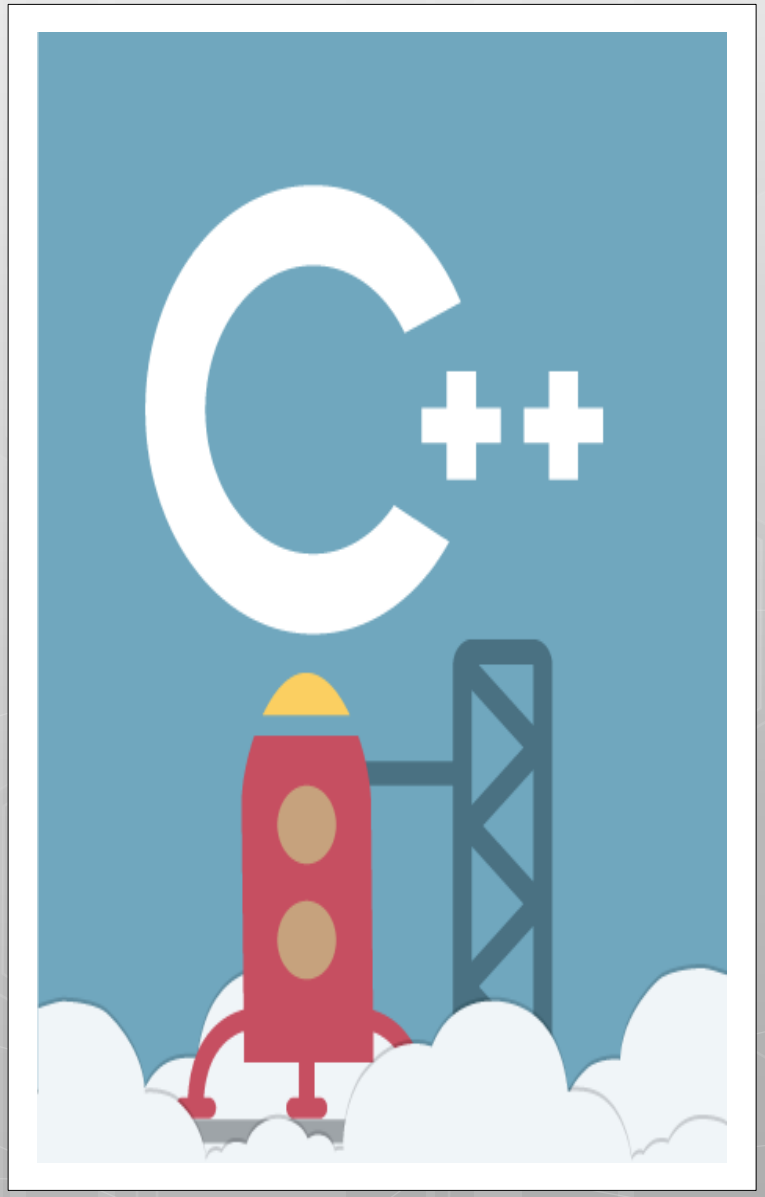
I. Given a list, copy it.
II. Reverse a Linked List.
III. Given a Linked List
a1→a2→a3….→an→b1→b2→…..→bn.
Convert it to a1→b1→a2→b2…→an→bn
IV. Implement Selection Sort.

Applications of Linked List

I. Polynomial Expressions.
II. Radix Sort

What is next class about?

I. Doubts + Problem Solving



Thank You!

Anushray Gupta

anushray@codingblocks.com
+91-9555567876