BINUS UNIVERSITY

People
Innovation
Excellence

Course : Data Structures

Effective Period : December 2018

Linked List I

Session 03

Learning Outcomes

At the end of this session, students will be able to:

- LO 1: Explain the concept of data structures and its usage in Computer Science
- LO 2: Illustrate any learned data structure and its usage in application
- LO 3: Apply data structures using C

Outline

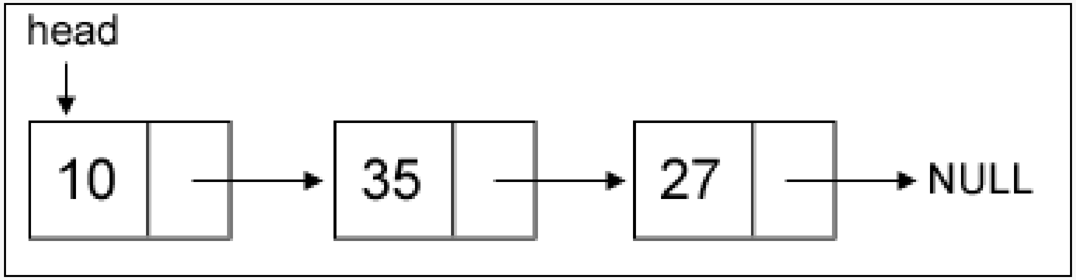
1. Linked List Introduction
2. Linked List vs Array
3. Single Linked List
4. Polynomial Representation

Linked List Introduction

- **Linked list** is a data structure that consists of a sequence of data records such that each record there is a field that contains a reference to the next record in the sequence.
- Linked list allows **insertion** and **deletion** of any element at any location.
- Linked list is used in many algorithms for **solving real-time problems,** when the number of elements to be stored is unpredictable and also during sequential access of elements.
- A linked list consists of two types:
  - Single Linked List
  - Double Linked List

Linked List Introduction

- Take a look at the following figure:



- Example of a list whose nodes contain two fields:
- an integer value and a link to the next node.
- Linked list which node contain only a single link to other node is called single linked list.

Linked List versus Array

Array:

- ❖ Linear collection of data elements
- ❖ Store value in consecutive memory locations
- ❖ Can be random in accessing of data

Linked List:

- ❖ Linear collection of nodes
- ❖ Doesn’t store its nodes in consecutive memory locations
- ❖ Can be accessed only in a sequential manner

BINUS
UNIVERSITY

Memory Allocation: Dynamic

- If you need to allocate memory dynamically (in runtime), you can use malloc in C/C++.
- To de-allocate you can use free.

```
int  *px = (int *) malloc(sizeof(int));
char *pc = (char *) malloc(sizeof(char));
*px = 205;
*pc = ‘A’;
printf( “%d %c\n”, *px, *pc );
free(px);
free(pc);
```

People
Innovation
Excellence

Single Linked List

- To create a linked list, we first need to define a node structure for the list.

- Supposed we want to create a list of integers.

```
struct tnode {
        int value;
        struct tnode *next;
};

struct tnode *head = 0;
```

head is the pointer to the **first element** in our linked list.

Single Linked List

- To create a list, we first need to define a node structure for the list.

- Supposed we want to create a list of integers.

```
struct tnode {
    int value;
    struct tnode *next;
};

struct tnode *head = 0;
```

head is the pointer to the **first element** in our linked list.

Single Linked List: Insert

- To insert a new value, first we should dynamically allocate a new node and assign the value to it and then connect it with the existing linked list.

- Supposed we want to append the new node **in front of the head.**

```
struct tnode *node =
    (struct tnode*) malloc(sizeof(struct tnode));
node->value = x;
node->next  = head;
head = node;
```

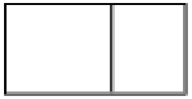
Operator -> has the same meaning as:

```
(*node).value = x;
(*node).next  = head;
```

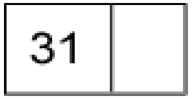
Single Linked List: Insert

Append a new node in front of head. Assuming there is already a linked list containing 10, 35, 27.

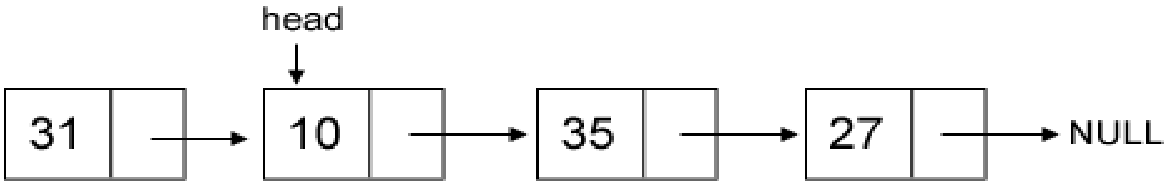
```
struct tnode *node = (struct tnode*) malloc(sizeof(struct tnode));
```



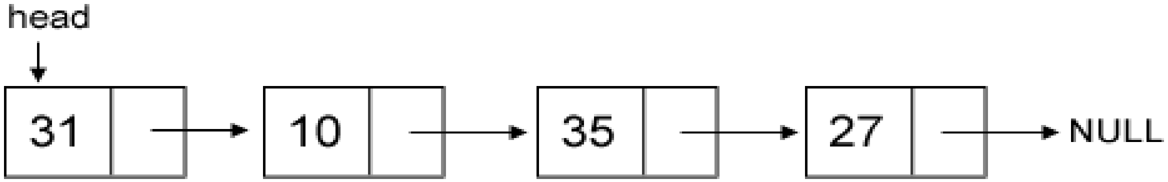
```
node->value = 31;
```



```
node->next = head;
```



```
head = next;
```



People
Innovation
Excellence

Single Linked List: Insert

How about the algorithm of inserting new node in the **middle** and in the **last** of the single linked list?

Can you try it by yourself?

You should try!

Single Linked List: Delete

- To delete a value, first we should find the location of node which store the value we want to delete, remove it, and connect the remaining linked list.

- Supposed the value we want to remove is x and assuming x is exist in linked list and it is unique.

- There are two conditions we should pay attention to:
  - ❖ if x is **in a head node** or
  - ❖ if x is **not in a head node**.

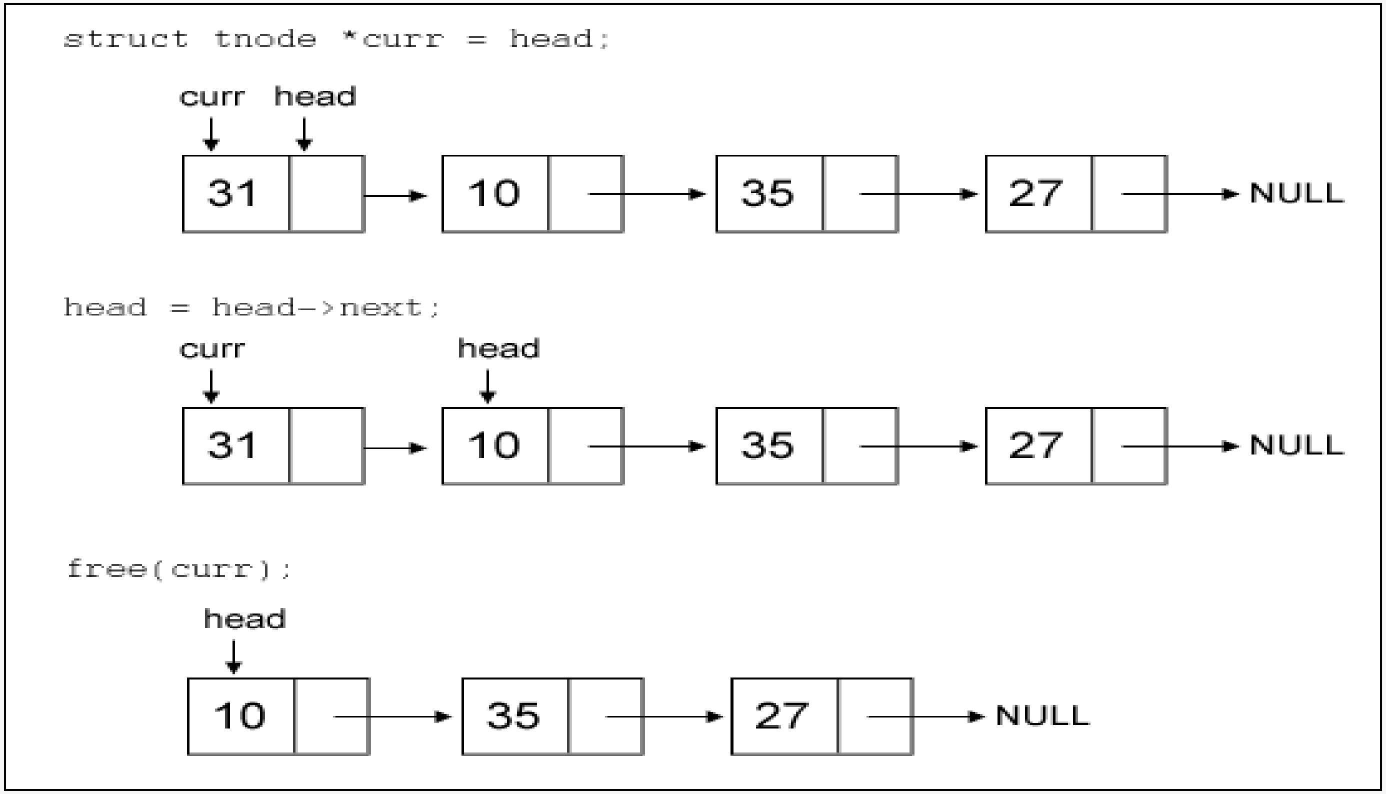
Single Linked List: Delete

```
struct tnode *curr = head;
// if x is in head node
if ( head->value == x ) {
   head = head->next;
   free(curr);
}
// if x is not in head node, find the location
else {
   while ( curr->next->value != x ) curr = curr->next;
   struct tnode *del = curr->next;
   curr->next = del->next;
   free(del);
}
```

People
Innovation
Excellence

Single Linked List: Delete

- Deleting 31 (located at head)



```
struct tnode *curr = head;
```

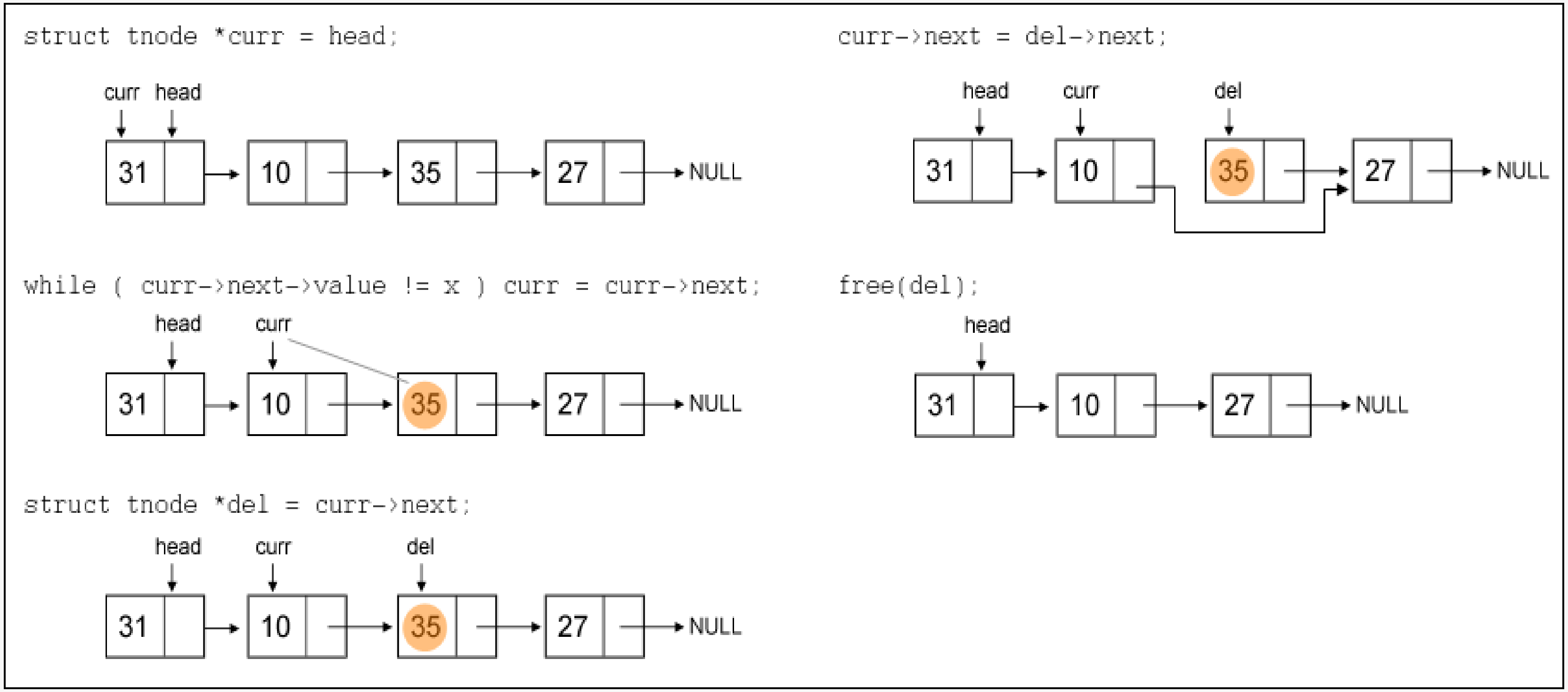
```
head = head->next;
```

```
free(curr);
```

People
Innovation
Excellence

Single Linked List: Delete

- Deleting 35 (not located at head)



```
struct tnode *curr = head;
```

```
while ( curr->next->value != x ) curr = curr->next;
```

```
struct tnode *del = curr->next;
```

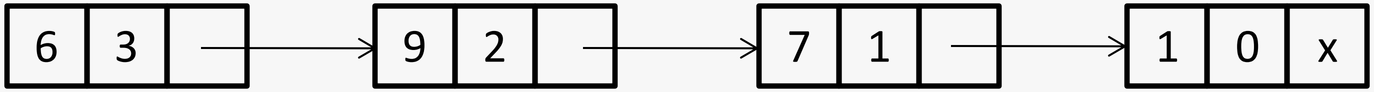
```
curr->next = del->next;
```

```
free(del);
```

People
Innovation
Excellence

Polynomial Representation

- Polynomial is given as $6x^3 + 9x^2 + 1$
- Every individual term in a polynomial consists of two parts, a coefficient and a power
- Here, 6, 9, 7, and 1 are the coefficients of the terms that have 3, 2, 1, and 0 as their power respectively.
- Every term of a polynomial can be represented as a node of the linked list.



6	3	→
9	2	→
7	1	→
1	0	x

Summary

- Linked list is useful, especially in solving real-time problems where the number of elements to be stored is unpredictable and also during sequential access of elements.
- Linked List has two types, single and double linked list.
- Single linked list is characterized by having a single one way link from a list pointing to another list

BINUS UNIVERSITY

People
Innovation
Excellence

References

- S. Sridhar. 2015. Design and Analysis of Algorithms. Oxford University Press. New Delhi. ISBN: 9780198093695. Chapter 5
- Reema Thareja. 2014. Data structures using C. Oxford University Press. New Delhi. ISBN:9780198099307. Chapter 6
- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, & Clifford Stein. (2009). Introduction to Algorithms. 03. The MIT Press. London. ISBN: 9780262033848. Chapter 10
- Linked List, https://visualgo.net/en/list?slide=3