GeeksforGeeks
A computer science portal for geeks

Custom Search

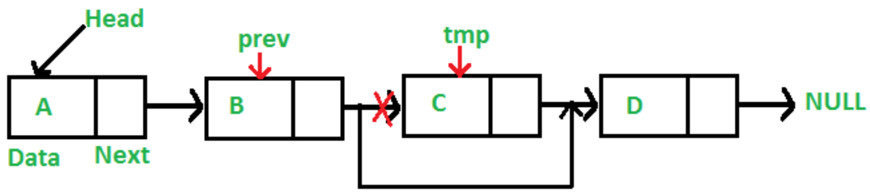
COURSES

HIRE WITH US

Linked List | Set 3 (Deleting a node)

We have discussed Linked List Introduction and Linked List Insertion in previous posts on singly linked list.

Let us formulate the problem statement to understand the deletion process. *Given a 'key', delete the first occurrence of this key in linked list.*
To delete a node from linked list, we need to do following steps.
1) Find previous node of the node to be deleted.
2) Change the next of previous node.
3) Free memory for the node to be deleted.



Recommended: Please solve it on "*PRACTICE*" first, before moving on to the solution.

Since every node of linked list is dynamically allocated using malloc() in C, we need to call free() for freeing memory allocated for the node to be deleted.

C/C++

```
// A complete working C program to demonstrate deletion in singly
// linked list
#include <stdio.h>
#include <stdlib.h>

// A linked list node
struct Node
{
    int data;
    struct Node *next;
};

/* Given a reference (pointer to pointer) to the head of a list
   and an int, inserts a new node on the front of the list. */
void push(struct Node** head_ref, int new_data)
{
    struct Node* new_node = (struct Node*) malloc(sizeof(struct Node));
    new_node->data  = new_data;
    new_node->next = (*head_ref);
    (*head_ref)    = new_node;
}

/* Given a reference (pointer to pointer) to the head of a list
   and a key, deletes the first occurrence of key in linked list */
```

```
void deleteNode(struct Node **head_ref, int key)
{
    // Store head node
    struct Node* temp = *head_ref, *prev;

    // If head node itself holds the key to be deleted
    if (temp != NULL && temp->data == key)
    {
        *head_ref = temp->next;   // Changed head
        free(temp);               // free old head
        return;
    }

    // Search for the key to be deleted, keep track of the
    // previous node as we need to change 'prev->next'
    while (temp != NULL && temp->data != key)
    {
        prev = temp;
        temp = temp->next;
    }

    // If key was not present in linked list
    if (temp == NULL) return;

    // Unlink the node from linked list
    prev->next = temp->next;

    free(temp);  // Free memory
}

// This function prints contents of linked list starting from
// the given node
void printList(struct Node *node)
{
    while (node != NULL)
    {
        printf(" %d ", node->data);
        node = node->next;
    }
}

/* Drier program to test above functions*/
int main()
{
    /* Start with the empty list */
    struct Node* head = NULL;

    push(&head, 7);
    push(&head, 1);
    push(&head, 3);
    push(&head, 2);

    puts("Created Linked List: ");
    printList(head);
    deleteNode(&head, 1);
    puts("\nLinked List after Deletion of 1: ");
    printList(head);
    return 0;
}
```

Java

```
// A complete working Java program to demonstrate deletion in singly
// linked list
class LinkedList
{
    Node head; // head of list

    /* Linked list Node*/
    class Node
    {
        int data;
        Node next;
        Node(int d)
        {
            data = d;
            next = null;
        }
    }
    /* Given a key, deletes the first occurrence of key in linked list */
    void deleteNode(int key)
    {
        // Store head node
        Node temp = head, prev = null;

        // If head node itself holds the key to be deleted
        if (temp != null && temp.data == key)
        {
            head = temp.next; // Changed head
            return;
        }

        // Search for the key to be deleted, keep track of the
        // previous node as we need to change temp.next
        while (temp != null && temp.data != key)
        {
            prev = temp;
            temp = temp.next;
        }

        // If key was not present in linked list
        if (temp == null) return;

        // Unlink the node from linked list
        prev.next = temp.next;
    }

    /* Inserts a new Node at front of the list. */
    public void push(int new_data)
    {
        Node new_node = new Node(new_data);
        new_node.next = head;
        head = new_node;
    }

    /* This function prints contents of linked list starting from
        the given node */
    public void printList()
    {
        Node tnode = head;
        while (tnode != null)
        {
            System.out.print(tnode.data+" ");
            tnode = tnode.next;
        }
    }

    /* Drier program to test above functions. Ideally this function
    should be in a separate user class. It is kept here to keep
    code compact */
    public static void main(String[] args)
    {
        LinkedList llist = new LinkedList();

        llist.push(7);
        llist.push(1);
        llist.push(3);
        llist.push(2);

        System.out.println("\nCreated Linked list is:");
        llist.printList();

        llist.deleteNode(1); // Delete node at position 4

        System.out.println("\nLinked List after Deletion at position 4:");
        llist.printList();
    }
}
```

Python

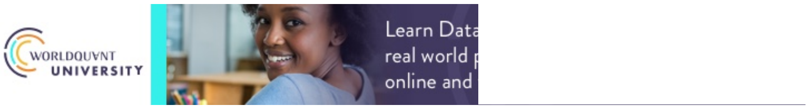
```
# Python program to delete a node from linked list

# Node class
class Node:
    # Constructor to initialize the node object
    def __init__(self, data):
        self.data = data
        self.next = None

class LinkedList:

    # Function to initialize head
    def __init__(self):
        self.head = None

    # Function to insert a new node at the beginning
    def push(self, new_data):
        new_node = Node(new_data)
        new_node.next = self.head
        self.head = new_node

    # Given a reference to the head of a list and a key,
    # delete the first occurence of key in linked list
    def deleteNode(self, key):

        # Store head node
        temp = self.head

        # If head node itself holds the key to be deleted
        if (temp is not None):
            if (temp.data == key):
                self.head = temp.next
                temp = None
                return

        # Search for the key to be deleted, keep track of the
        # previous node as we need to change 'prev.next'
        while(temp is not None):
            if temp.data == key:
                break
            prev = temp
            temp = temp.next

        # if key was not present in linked list
        if(temp == None):
            return

        # Unlink the node from linked list
        prev.next = temp.next

        temp = None


    # Utility function to print the linked LinkedList
    def printList(self):
        temp = self.head
        while(temp):
            print " %d" %(temp.data),
            temp = temp.next


# Driver program
llist = LinkedList()
llist.push(7)
llist.push(1)
llist.push(3)
llist.push(2)

print "Created Linked List: "
llist.printList()
llist.deleteNode(1)
print "\nLinked List after Deletion of 1:"
llist.printList()

# This code is contributed by Nikhil Kumar Singh (nickzuck_007)
```

Output:

```
Created Linked List:
 2  3  1  7
Linked List after Deletion of 1:
 2  3  7
```

Linked List | Set 3 (Deleting a node) | GeeksforGeeks

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.



Recommended Posts:

Find the middle of a given linked list in C and Java

Program for n'th node from the end of a Linked List

Write a function to get Nth node in a Linked List

Given only a pointer/reference to a node to be deleted in a singly linked list, how do you delete it?

Detect loop in a linked list

Write a function to delete a Linked List

Write a function that counts the number of times a given int occurs in a Linked List

Reverse a linked list

Given only a pointer to a node to be deleted in a singly linked list, how do you delete it?

Write a function to get the intersection point of two Linked Lists

Function to check if a singly linked list is palindrome

The Great Tree-List Recursion Problem.

Clone a linked list with next and random pointer | Set 1

Memory efficient doubly linked list

Given a linked list which is sorted, how will you insert in sorted way

Improved By : mlv

Article Tags : Linked List

Practice Tags : Linked List

53

□ To-do □ Done

1.7

Based on **192** vote(s)

Feedback/ Suggest Improvement Notes Improve Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

A computer science portal for geeks

5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305
feedback@geeksforgeeks.org

COMPANY
About Us
Careers
Privacy Policy
Contact Us

LEARN
Algorithms
Data Structures
Languages
CS Subjects
Video Tutorials

PRACTICE
Courses
Company-wise
Topic-wise
How to begin?

CONTRIBUTE
Write an Article
Write Interview Experience
Internships
Videos

@geeksforgeeks, Some rights reserved