Quick Links for Linked List

- Quizzes
- Videos on Linked List

Singly Linked List

- Introduction to Linked List
- Linked List vs Array
- A Programmer's approach of looking at Array vs. Linked List
- Linked List Insertion
- Linked List Deletion (Deleting a given key)
- Linked List Deletion (Deleting a key at given position)
- How to write C functions that modify head pointer of a Linked List?

Linked List | Set 2 (Inserting a node)

1.7

We have introduced Linked Lists in the previous post. We also created a simple linked list with 3 nodes and discussed linked list traversal.

All programs discussed in this post consider following representations of linked list .

C Java Python

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

Run on IDE

In this post, methods to insert a new node in linked list are discussed. A node can be added in three ways

1) At the front of the linked list

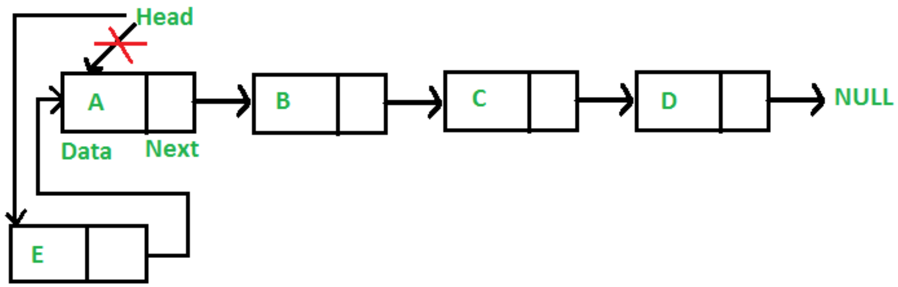
2) After a given node.

3) At the end of the linked list.

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

Add a node at the front: (A 4 steps process)

The new node is always added before the head of the given Linked List. And newly added node becomes the new head of the Linked List. For example if the given Linked List is 10->15->20->25 and we add an item 5 at the front, then the Linked List becomes 5->10->15->20->25. Let us call the function that adds at the front of the list is push(). The push() must receive a pointer to the head pointer, because push must change the head pointer to point to the new node (See this)



Following are the 4 steps to add node at the front.

C Java Python

ascending and descending orders?

Point arbit pointer to greatest value right side node in a linked list

Flattening a Linked List

Explore More...

Circular Linked List

Circular Linked List Introduction and Applications,

Circular Linked List Traversal

Split a Circular Linked List into two halves

Sorted insert for circular linked list

Doubly Linked List

Doubly Linked List Introduction and Insertion

Delete a node in a Doubly Linked Lis

Reverse a Doubly Linked List

The Great Tree-List Recursion Problem

Clone a linked list with next and random pointer

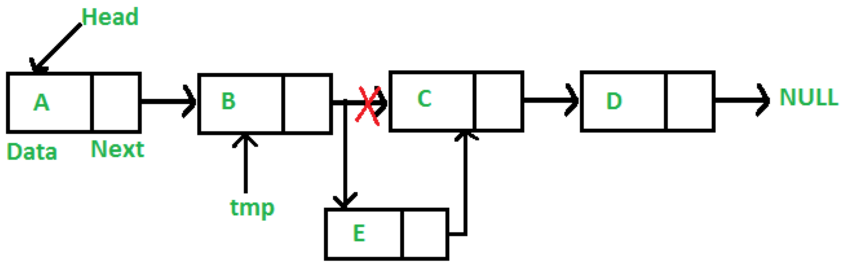
```
/* Given a reference (pointer to pointer) to the head of
   and an int,  inserts a new node on the front of the li
void push(struct Node** head_ref, int new_data)
{
    /* 1. allocate node */
    struct Node* new_node = (struct Node*) malloc(sizeof(

    /* 2. put in the data  */
    new_node->data  = new_data;

    /* 3. Make next of new node as head */
    new_node->next = (*head_ref);

    /* 4. move the head to point to the new node */
    (*head_ref)    = new_node;
}
```

Run on IDE

Time complexity of push() is O(1) as it does constant amount of work.

Add a node after a given node: (5 steps process)

We are given pointer to a node, and the new node is inserted after the given node.



Longest Common Subsequence

Breadth First Traversal or BFS

School Programming

Longest Repeated Subsequence

Longest Palindromic Subsequence

Detect a negative cycle in a Graph | (Bellman Ford)

0-1 Knapsack Problem

GATE CS Notes

Depth First Traversal or DFS for a Graph

Reverse a linked list

Most Visited Posts

Top 10 Algorithms and Data Structures for Competitive Programming

Top 10 algorithms in Interview Questions

How to begin with Competitive Programming?

Step by Step Guide for Placement Preparation

How to prepare for ACM-ICPC?

Insertion Sort , Binary Search , QuickSort , MergeSort , HeapSort

Popular Categories

QuickSort on Doubly Linked List

Swap Kth node from beginning with Kth node from end in a Linked List

Merge Sort for Doubly Linked List

Arrays

Segregate 0s and 1s in an array

Rearrange positive and negative numbers in O(n) time and O(1) extra space

Maximum Product Sub-array

More...

Strings

Return maximum occurring character in an input string

Check whether two strings are anagram of each other

Remove all duplicates from a given string

More...

Stack

Implement two stacks in an array

Infix to Postfix Conversion

C Java Python

```
/* Given a node prev_node, insert a new node after the gi
   prev_node */
void insertAfter(struct Node* prev_node, int new_data)
{
    /*1. check if the given prev_node is NULL */
    if (prev_node == NULL)
    {
       printf("the given previous node cannot be NULL");
       return;
    }

    /* 2. allocate new node */
    struct Node* new_node =(struct Node*) malloc(sizeof(s

    /* 3. put in the data  */
    new_node->data  = new_data;

    /* 4. Make next of new node as next of prev_node */
    new_node->next = prev_node->next;

    /* 5. move the next of prev_node as new_node */
    prev_node->next = new_node;
}
```

Run on IDE

Time complexity of insertAfter() is O(1) as it does constant amount of work.

Add a node at the end: (6 steps process)

The new node is always added after the last node of the given Linked List. For example if the given Linked List is 5->10->15->20->25 and we add an item 30 at the end, then the Linked List becomes 5->10->15->20->25->30.

Interview Experiences

Advanced Data Structures

Dynamic Programming

Greedy Algorithms

Backtracking

Pattern Searching

Divide & Conquer

Geometric Algorithms

Searching

Sorting

Analysis of Algorithms

Mathematical Algorithms

Randomized Algorithms

Recursion

Game Theory

Statistical Algorithms

Tags

Advanced Data Structure Amazon Aptitude Aptitude Arrays Articles Bit Magic C C C++ C++ Quiz Computer Networks CPP-Library C Quiz Data Structures Data Structures DBMS Dynamic Programming Experienced GBlog Geometric Graph Hash Internship Interview using Stack

The Stock Span Problem

More...

Queue

Implement Queue using Stacks

Implementation of Deque using circular array

How to efficiently implement k Queues in a single array?

More...

Binary Tree

Print level order traversal line by line | Set 1

Diagonal Traversal of Binary Tree

Boundary Traversal of binary tree

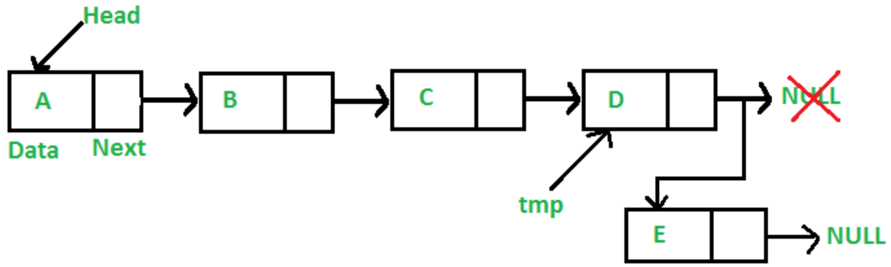
More...

Binary Search Tree

Lowest Common Ancestor in a Binary Search Tree

Check if each internal node of a BST has exactly one child

Since a Linked List is typically represented by the head of it, we have to traverse the list till end and then change the next of last node to new node.



Following are the 6 steps to add node at the end.

C Java Python

```
/* Given a reference (pointer to pointer) to the head
   of a list and an int, appends a new node at the end  *
void append(struct Node** head_ref, int new_data)
{
    /* 1. allocate node */
    struct Node* new_node = (struct Node*) malloc(sizeof(

    struct Node *last = *head_ref;  /* used in step 5*/

    /* 2. put in the data  */
    new_node->data  = new_data;

    /* 3. This new node is going to be the last node, so
          of it as NULL*/
    new_node->next = NULL;

    /* 4. If the Linked List is empty, then make the new
    if (*head_ref == NULL)
    {
       *head_ref = new_node;
```

Experiences Java Linked Lists Mathematical Matrix Microsoft number-digits Program Output Project Puzzles Python QA - Placement Quizzes QA - Placement Quizzes School Programming Searching series Sorting STL Strings Technical Scripter Trees

Recent Comments

Merge Two Balanced Binary Search Trees

More...

Heap

Binomial Heap

Fibonacci Heap

More...

Graph

Detect Cycle in an Undirected Graph

Longest Path in a Directed Acyclic Graph

More...

Hashing

Count distinct elements in every window of size k

Largest subarray with equal number of 0s and 1s

More...

Searching

Linear Search

Binary Search

Fibonacci Search

```
        return;
    }

    /* 5. Else traverse till the last node */
    while (last->next != NULL)
        last = last->next;

    /* 6. Change the next of last node */
    last->next = new_node;
    return;
}
```

Run on IDE

Time complexity of append is O(n) where n is the number of nodes in linked list. Since there is a loop from head to end, the function does O(n) work.
This method can also be optimized to work in O(1) by keeping an extra pointer to tail of linked list/

Following is a complete program that uses all of the above methods to create a linked list.

C Java Python

```
// A complete working C program to demonstrate all insert
// on Linked List
#include <stdio.h>
#include <stdlib.h>

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

More...

Sorting

Bubble Sort

Insertion Sort

Quick Sort

More...

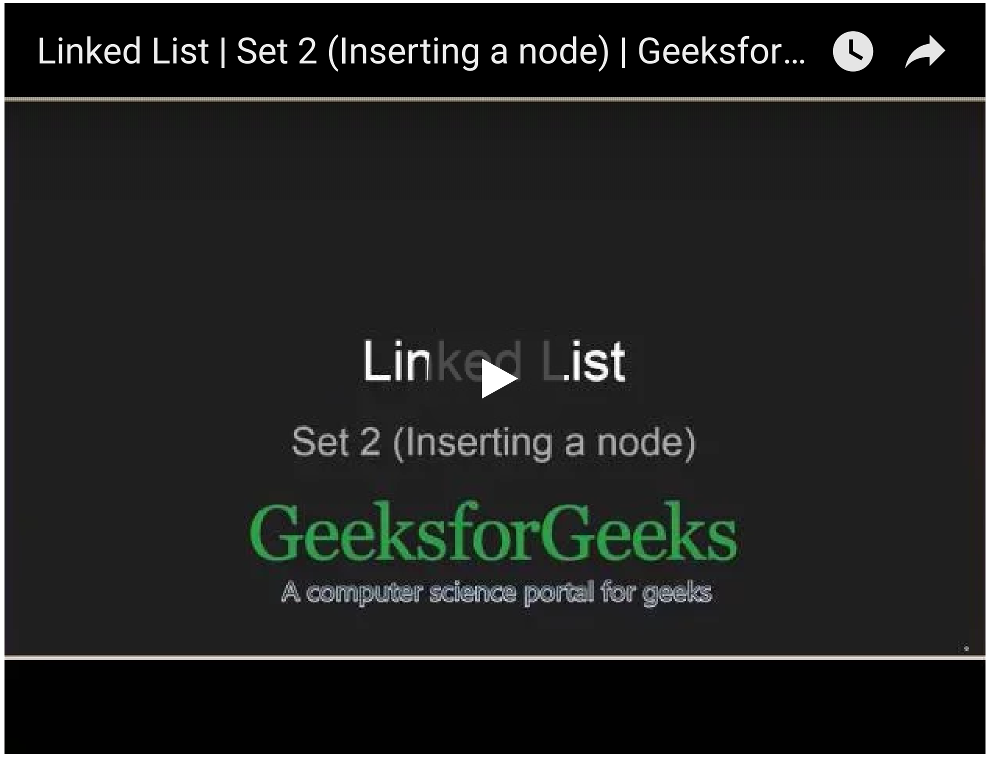
```
/* Given a reference (pointer to pointer) to the head of 
   an int, inserts a new node on the front of the list. *
void push(struct Node** head_ref, int new_data)
{
    /* 1. allocate node */
    struct Node* new_node = (struct Node*) malloc(sizeof(

    /* 2. put in the data  */
    new_node->data  = new_data;

    /* 3. Make next of new node as head */
    new_node->next = (*head_ref);

    /* 4. move the head to point to the new node */
    (*head_ref)    = new_node;
}

/* Given a node prev_node, insert a new node after the gi
   prev_node */
void insertAfter(struct Node* prev_node, int new_data)
{
    /*1. check if the given prev_node is NULL */
    if (prev_node == NULL)
    {
      printf("the given previous node cannot be NULL");
      return;
    }

    /* 2. allocate new node */
    struct Node* new_node =(struct Node*) malloc(sizeof(s

    /* 3. put in the data  */
    new_node->data  = new_data;

    /* 4. Make next of new node as next of prev_node */
    new_node->next = prev_node->next;

    /* 5. move the next of prev_node as new_node */
    prev_node->next = new_node;
}

/* Given a reference (pointer to pointer) to the head
   of a list and an int, appends a new node at the end  *
void append(struct Node** head_ref, int new_data)
{
    /* 1. allocate node */
    struct Node* new_node = (struct Node*) malloc(sizeof(
    struct Node *last = *head_ref;  /* used in step 5*/

    /* 2. put in the data  */
    new_node->data  = new_data;

    /* 3. This new node is going to be the last node, so
          it as NULL*/
    new_node->next = NULL;

    /* 4. If the Linked List is empty, then make the new
    if (*head_ref == NULL)
    {
       *head_ref = new_node;
       return;
    }

    /* 5. Else traverse till the last node */
    while (last->next != NULL)
        last = last->next;

    /* 6. Change the next of last node */
    last->next = new_node;
    return;
}

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

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

  // Insert 6.  So linked list becomes 6->NULL
  append(&head, 6);

  // Insert 7 at the beginning. So linked list becomes 7-
  push(&head, 7);

  // Insert 1 at the beginning. So linked list becomes 1-
```

```
 push(&head, 1);

 // Insert 4 at the end. So linked list becomes 1->7->6-
 append(&head, 4);

 // Insert 8, after 7. So linked list becomes 1->7->8->6
 insertAfter(head->next, 8);

 printf("\n Created Linked list is: ");
 printList(head);

 return 0;
}
```

Run on IDE

Output:

```
Created Linked list is:  1  7  8  6  4
```


Linked List | Set 2 (Inserting a node) | Geeksfor...
Linked List
Set 2 (Inserting a node)
GeeksforGeeks
A computer science portal for geeks

You may like to try **Practice MCQ Questions on Linked List**

Asked in: TCS, Wipro

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

GATE CS Corner **Company Wise Coding Practice**

Linked Lists

Login to Improve this Article

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

Recommended Posts:

Linked List | Set 3 (Deleting a node)
Linked List | Set 1 (Introduction)
QuickSort on Doubly Linked List
Delete a Linked List node at a given position
Linked List vs Array
Alternate sorting of Linked list
Multiply two numbers represented as linked lists into a third list
Linked List Sum of Nodes Between 0s
Recursive insertion and traversal linked list
Exchange first and last nodes in Circular Linked List

<< Previous Post

Next Post >>

(Login to Rate)

1.7 Average Difficulty : **1.7/5.0**
Based on **136** vote(s)

Basic Easy Medium Hard Expert

🟩 Add to TODO List

🟩 Mark as DONE

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

Load Comments Share this post!