Nth node from the end of a Linked List

Given a Linked List and a number n, write a function that returns the value at the nth node from end of the Linked List.

Method 1 (Use length of linked list)
1) Calculate the length of Linked List. Let the length be len.
2) Print the (len – n + 1)th node from the begining of the Linked List.

```
#include<stdio.h>
#include<stdlib.h>
```

```
/* Link list node */
struct node
{
  int data;
  struct node* next;
};

/* Function to get the nth node from the last of a linked list*/
void printNthFromLast(struct node* head, int n)
{
    int len = 0, i;
    struct node *temp = head;

    // 1) count the number of nodes in Linked List
    while (temp != NULL)
    {
        temp = temp->next;
        len++;
    }

    // check if value of n is not more than length of the linked list
    if (len < n)
      return;

    temp = head;

    // 2) get the (n-len+1)th node from the begining
    for (i = 1; i < len-n+1; i++)
       temp = temp->next;

    printf ("%d", temp->data);

    return;
}

void push(struct node** head_ref, int new_data)
{
  /* allocate node */
  struct node* new_node =
          (struct node*) malloc(sizeof(struct node));

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

  /* link the old list off the new node */
  new_node->next = (*head_ref);

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

/* Drier program to test above function*/
int main()
```

```
{
  /* Start with the empty list */
  struct node* head = NULL;

  // create linked 35->15->4->20
  push(&head, 20);
  push(&head, 4);
  push(&head, 15);
  push(&head, 35);

  printNthFromLast(head, 5);
  getchar();
  return 0;
}
```

Following is a recursive C code for the same method. Thanks to Anuj Bansal for providing following code.

```
void printNthFromLast(struct node* head, int n)
{
    static int i = 0;
    if(head == NULL)
       return;
    printNthFromLast(head->next, n);
    if(++i == n)
       printf("%d", head->data);
}
```

Time Complexity: O(n) where n is the length of linked list.

Method 2 (Use two pointers)
Maintain two pointers – reference pointer and main pointer. Initialize both reference and main pointers to head. First move reference pointer to n nodes from head. Now move both pointers one by one until reference pointer reaches end. Now main pointer will point to nth node from the end. Return main pointer.

Implementation:

```
#include<stdio.h>
#include<stdlib.h>

/* Link list node */
struct node
{
  int data;
  struct node* next;
};

/* Function to get the nth node from the last of a linked list*/
void printNthFromLast(struct node *head, int n)
{
```

```
    struct node *main_ptr = head;
    struct node *ref_ptr = head;

    int count = 0;
    if(head != NULL)
    {
       while( count < n )
       {
          if(ref_ptr == NULL)
          {
             printf("%d is greater than the no. of "
                       "nodes in list", n);
             return;
          }
          ref_ptr = ref_ptr->next;
          count++;
       } /* End of while*/

       while(ref_ptr != NULL)
       {
          main_ptr = main_ptr->next;
          ref_ptr  = ref_ptr->next;
       }
       printf("Node no. %d from last is %d ",
                 n, main_ptr->data);
    }
}

void push(struct node** head_ref, int new_data)
{
    /* allocate node */
    struct node* new_node =
              (struct node*) malloc(sizeof(struct node));

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

    /* link the old list off the new node */
    new_node->next = (*head_ref);

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

/* Drier program to test above function*/
int main()
{
    /* Start with the empty list */
    struct node* head = NULL;
    push(&head, 20);
    push(&head, 4);
    push(&head, 15);
```

```
  printNthFromLast(head, 3);
  getchar();
}
```

Time Complexity: O(n) where n is the length of linked list.

Please write comments if you find the above codes/algorithms incorrect, or find other ways to solve the same problem.

Related Topics:

- Clone a linked list with next and random pointer | Set 2
- Given a linked list of line segments, remove middle points
- Construct a Maximum Sum Linked List out of two Sorted Linked Lists having some Common nodes
- Given a linked list, reverse alternate nodes and append at the end
- Pairwise swap elements of a given linked list by changing links
- Self Organizing List | Set 1 (Introduction)
- Merge a linked list into another linked list at alternate positions
- QuickSort on Singly Linked List

Tags: Linked Lists

Writing code in comment? Please use **ideone.com** and share the link here.

Yeah true, recursion is the most wonderful concept in programming.... provided you choose to use it correctly and understand it.

Unlike the recursive code which is mostly simple it is real hard to develop your mind to think recursively.

I tried putting up a simple explanation on how to recursion, if someone is interested.
http://techieme.in/recursion/

• Reply • Share ›

kaushik Lele • a month ago

2nd option is really smart one.
I had thought of creating a stack adding all the nodes in stack and then popping n times from stack

• Reply • Share ›

satheesh • a month ago

```
private Node kthFromLast(Node head, int k) {
Node kNode = head;
int kIndex = 0, i = 0;

while (head != null) {
if ((i - kIndex) == k) {
kNode = kNode.next;
kIndex++;
}
head = head.next;
i++;
}

return kNode;

}
```

• Reply • Share ›

Utkarsh Mishra • a month ago

```
void endprint(node* head,int n)
{
node *mid,*loop;
int i;
mid=loop=head;
int c=0,middle=1;
while(loop->link!=NULL)
{
c++;
```

```
if(c%2!=0)
{
mid=mid->link;
middle++;
}
loop=loop->link;
}
if(n<=middle)
{
for(i=0;i<c 2+1-n;i++)="" mid="mid-">link;
printf("%d",mid->data);
}
else
{
for(i=0;i<c-n+1;i++) head="head-">link;
printf("%d",head->data);
}
}
```

• Reply • Share ›

Arman Ghassemi • 3 months ago

What would be the downsides to using a stack to push all the data then do n pops, returning the nth pop? I think this would take O(n) as well.

• Reply • Share ›

Goku ➦ Arman Ghassemi • 2 months ago

That would take O(n) memory.Here we are doing it in O(1) memory.

2 • Reply • Share ›

rishabh kushwaha • 4 months ago

in second case....

• Reply • Share ›

rishabh kushwaha • 4 months ago

is you print nth element from the last, given the length of the linked list is n, it will print wrong...

3 • Reply • Share ›

codecrecker • 6 months ago

```
#include<stdio.h>

#include<stdlib.h>

typedef struct nd node;
```

```
struct nd{

int d;

node *n;

};

node *head,*ptr;

node *createNode(int d)

{

node *tmp;
```

see more

• Reply • Share ›

The_Geek • 6 months ago

@geeksforgeeks, In the 2nd method instead of condition while(ref_ptr != NULL), it should be while(ref_ptr->next != NULL).
Please check it once, or explain it once if not wrong.

• Reply • Share ›

Siya ➚ The_Geek • 5 months ago

I find its correct,the difference between ref_ptr and main_ptr is n.If we will do ref->next!=NULL then i will stop one element before the end element.Then i will give n+1 th element from end not n.

Correct me if i am thinking wrong !!

2 • Reply • Share ›

The_Geek ➚ Siya • 5 months ago

Take a linked list having elements 1,2,3,4,5,6. Now we have to find 2nd element frm last i.e. 5.
Denote main_ptr as M and Ref_ptr as R. Initially M and R are pointing at 1. Move R only once, bcz condition is count<2, i.e. we can only move R once. Now it will point to 2. Now M is pointing at 1 and R is pointing at 2. Move M and R one by one, you will see that R will reach at 6, when M will point at 5, which is our desired element. So we need to stop moving R further, thats y we need this condition (R->next!=NULL).
Hope this is correct.

• Reply • Share ›

Siya ➚ The_Geek • 5 months ago

No !! R will point to 3 because count is initialized to 0.So in first iteration count=0 and <2 true so R=2;in second iteration count=1 and <2 true so R points to 3 and count =2;in third iteration count=2<2 false so stop.Then the same steps u have used...
Hope this is correct :P
3 • Reply • Share ›

The_Geek ➚ Siya • 5 months ago
ohh.. my bad ***... i was making ths count mistake..
Thanks fr clarifying it.
2 • Reply • Share ›

HATIM ALI • 7 months ago
IN 2ND METHOD ,THE EXPRESSION OF SECOND WHILE LOOP WILL BE ref_ptr->next != NULL , otherwise it will give wrong output
1 • Reply • Share ›

Vaishal Shah ➚ HATIM ALI • 6 months ago
yes you are right please change here.
1 • Reply • Share ›

Preethi • 7 months ago
Use recursion like,

```
count=0;
nNode(node){
if(node==NULL)
return;
nNode(node->next);
count+=1;
if(count==n)
printf("node->data");
}
```

Time complexity=O(n)
• Reply • Share ›

Gowtham • 8 months ago
In the second method we are checking whether the head is null..then why we have to check the ref ptr
• Reply • Share ›

Gaurav ➚ Gowtham • 8 months ago
ref ptr is checked when head is not null but value of 'n' is greater than total no. of nodes in linked list. In that case printing nth node from last is not possible, hence the check

1 • Reply • Share ›

Vikas Bansal • 8 months ago

liked the recursive technique and method 2

3 • Reply • Share ›

Abhinav Bhardwaj • 8 months ago

Simple Recursion.

```
public int getNthFromtheLast(Node node,int k)
{

if(node==null)
return 0;
int count=getNthFromtheLast(node.next, k)+1;
if(count==k)
System.out.println(node.data);

return count;
}
```

13 • Reply • Share ›

Pranav Kumar Jha ➦ Abhinav Bhardwaj • 8 months ago

thumbs up (y)

• Reply • Share ›

Ramesh BG • 8 months ago

In the second method ...assume there is a list of 10 elements, if i need to get 3rd element element from the end how it is possible ?

• Reply • Share ›

Kim Jong-il ➦ Ramesh BG • 7 months ago

Thats simply work on the logic of first method, "Len-N+1"

One pointer is going to traverse complete Len and another start late by N, hence it will go upto the Len-N, and since it start from 1 then it became Len-N+1.

1 • Reply • Share ›

DS+Algo=Placement ➦ Ramesh BG • 8 months ago

move ref_ptr to 3rd node from head and then move both main_ptr and ref_ptr until ref_ptr becomes NULL. Now main_ptr is our required node

• Reply • Share ›

Guest • 9 months ago

keep two counters integer and remainder

keep two counters integer and remainder..
determine integer N/n by incrementing counter integer at every nth node during 1st traversal..
store quoteint=integer
lly determine remainder during 1st traversal..
In next traversal first traverse to the node number=remainder..
and then start the integer counter till we reach node with integer counter =quoteint-1

• Reply • Share ›

DS+Algo=Placement Guest • 8 months ago
Elaborate please

• Reply • Share ›

kinshuk chandra • 10 months ago
I really like the 2nd approach. But there is a recursive solution to this as well as written here - http://k2code.blogspot.in/2010/04/return-nth-node-from-end-of-linked-list.html:

```
node* findNthNode (node* head, int find, int& found){
    if(!head) {
        found = 1;
        return 0;
    }
    node* retval = findNthNode(head->next, find, found);
    if(found==find)
        retval = head;
    found = found + 1;
    return retval;
}
```

Thanks.

• Reply • Share ›

MK kinshuk chandra • 10 months ago
Didn't get your logic. Please explain

• Reply • Share ›

Sriram Ganesh MK • 9 months ago
Its a recursive function. "node* retval = findNthNode(head->next, find, found);" this line means proceed to next node no matter what the next line is (its pushed to system's stack). This happens till the end condition i.e. the null next to last node is reached. So the if(found==find) and the rest following this line are not at all executed till the end of list is reached. When the end is reached, head now points to null. So if(!head) is true and found is set to 1. now the statements after that recursive call is executed one by one (popped out from stack). Since last address is popped out, it now points to last node(prev of null). Now lets say you need 2nd from last (find=2). Now found is 1 so if(found==find) fails n found is ++

need 2nd from last (find=2). Now found is 1 so if(found==find) fails n found is 1 + 1 to 2. Function ends. So pop out the next address from stack which is the prev of last. now if(found==find) is true, so return the current pointer of the node.

^ | v • Reply • Share ›

Abdur ➚ kinshuk chandra • 10 months ago

Could you please explain its logic?

^ | v • Reply • Share ›

Sriram Ganesh ➚ Abdur • 9 months ago

Refer to the above comment

^ | v • Reply • Share ›

Arvind • a year ago

seems second method gives nth node from beginning instead of end. am i wrong? tried to run it on ideone.

^ | v • Reply • Share ›

VeridisQuo ➚ Arvind • a year ago

the program is correct
Note: the function push adds an element to the beginning of the linked list.

^ | v • Reply • Share ›

AMIT JAMBOTKAR ➚ VeridisQuo • a year ago

can you explain it.How it make diff adding elements from beginning

^ | v • Reply • Share ›

Alex ➚ AMIT JAMBOTKAR • 10 months ago

we are taking a difference of n between first and second pointer..
and when second pointer will reach to null/last node..
first pointer will be pointing to n th node from last.

1 ^ | v • Reply • Share ›

Guest • a year ago

although trivial, but for ppl looking code in java

```
public Link getNthNode(int N){
Link nthNode = null;
Link firstNode = head;
int cnt = 1;
while(firstNode != null){
if(cnt == N){
nthNode = head;
}
```

```
if(cnt > N){
nthNode = nthNode.getNext();
}
firstNode = firstNode.getNext();
++cnt;
}
return nthNode;
}
```

• Reply • Share ›

neo • a year ago

Recursive solution

```
typedef struct node n;

n* getn(n* head,int *k)
{
//Base case
if(head==NULL)
return NULL;
else
{
//tail recursion
n* t = getn(head->next,k);
if(t!=NULL)
return t;
if(*k==1)
return head;
else
*k=*k-1;
return NULL;
}
}
```

2 • Reply • Share ›

deepuanand ➚ neo • a year ago

wondering can we really do it via tail recursion as @neo mentioned ?

by the way @neo its not a tail recursion

• Reply • Share ›

Anand ➚ neo • a year ago

@neo: This is not tail recursion. go through the following link to better understand the tail recursion.

http://c2.com/cgi/wiki?TailRec

http://c2.com/cgi/wiki?FallRe...

• Reply • Share ›

sanjeen fsdjfu • a year ago

I am not sure about the requirement.But on my opinion it must print nth from the last element inserted into the linked list,but this code prints nth from first.

• Reply • Share ›

Guest • a year ago

I am not sure,this code is for printing nth from last,but it is printing nth from first.

• Reply • Share ›

Underground • 2 years ago

Above program would break if the SLL contains loop !

• Reply • Share ›

Guest • 2 years ago

Shouldn't count be initialized to 1 in Method 2 ????

• Reply • Share ›

hemanthreddy • 2 years ago

n value less than zero case is not handled here

• Reply • Share ›

rahulcynosure • 2 years ago

```
void printnthfromlast(struct Node * head,int n)
{
struct Node * temp = head;
int count=0;
while(head)
{
if(count>=n)
temp=temp->next;
head=head->next;
count++;
}
if(count>=n)
printf("%d",temp->data);
}
```

1 • Reply • Share ›

neelabh ➚ rahulcynosure • 2 years ago

Explanation of above code: In your coding first you initialize temp=head and count=0. in while loop you are incrementing head node until meet to last node. in the while loop

there is if condition in if condition will increment temp pointer until count>=n means it will maintain n node distance between head and temp pointer. after reaching last node temp will indicate the Nth node from last node.
Very nice solution given by you.

```
/* Paste your code here (You may delete these lines if not writing code) */
```

1 • Reply • Share ›

Kumar • 2 years ago

In Method 2: second while loop it should be
while(ref_ptr -> next != NULL)

• Reply • Share ›

Load more comments

- Interview Experiences
- Advanced Data Structures
- Dynamic Programming
- Greedy Algorithms
- Backtracking
- Pattern Searching
- Divide & Conquer
- Mathematical Algorithms
- Recursion
- Geometric Algorithms

Popular Posts

- All permutations of a given string
- Memory Layout of C Programs
- Understanding “extern” keyword in C
- Median of two sorted arrays
- Tree traversal without recursion and without stack!
- Structure Member Alignment, Padding and Data Packing
- Intersection point of two Linked Lists
- Lowest Common Ancestor in a BST.
- Check if a binary tree is BST or not
- Sorted Linked List to Balanced BST

Recent Comments

- lebron

  since the array size is 5, it takes constant...

  K’th Smallest/Largest Element in Unsorted Array | Set 3 (Worst Case Linear Time) · 3 hours ago

- lebron

  merge sort

  K’th Smallest/Largest Element in Unsorted Array | Set 3 (Worst Case Linear Time) · 3 hours ago

- Shubham Sharma

  You saved my time :)

  Searching for Patterns | Set 2 (KMP Algorithm) · 3 hours ago

- Prakhar

  Why so many LOCs, if I'm not wrong (please...

  Largest Sum Contiguous Subarray · 3 hours ago

- Aayush Gupta

  For R4 Q3, Another solution would be to use a...

  Amazon Interview Experience | Set 168 · 5 hours ago

- EigenHarsha

  For Power Of 2, We Simply Doing.. var1 =

  Practo Interview Experience | Set 2 (Off-Campus) · 5 hours ago