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)
```


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

```
      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

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

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);
```


Recent Comments

Abhi You live US or India?

Google (Mountain View) interview · 47 minutes ago

Aman Hi, Why arent we checking for conditions...

Write a C program to Delete a Tree. · 1 hour ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 1 hour ago

Sanjay Agarwal bool tree::Root_to_leaf_path_given_sum(tree...

Root to leaf path sum equal to a given number · 1 hour ago

GOPI GOPINATH @admin Highlight this sentence "We can easily...

Count trailing zeroes in factorial of a number · 1 hour ago

newCoder3006 If the array contains negative numbers also. We...

Find subarray with given sum · 2 hours ago

```
  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 Tpoics:

- 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
- Delete N nodes after M nodes of a linked list
- Design a stack with operations on middle element
- Swap Kth node from beginning with Kth node from end in a Linked List

28 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

kinshuk chandra • 10 days ago

I really like the 2nd approach. But there is a recursive solution to this as well as 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 ›

Arvind • a month ago

seems second method gives nth node from beginning instead of end. am i wr

• Reply • Share ›

VeridisQuo → Arvind • a month ago

the program is correct

Note: the function push adds an element to the beginning of the linked l

• Reply • Share ›

AMIT JAMBOTKAR → VeridisQuo • a month ago

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

• Reply • Share ›

Saket Pandey • a month 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 • 6 months ago

Recursive solution

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;
}
}
```

1 • Reply • Share ›

deepuanand → neo • 14 days 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 • 14 days ago

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

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

• Reply • Share ›

sanjeen fsdjfu • 8 months ago

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

• Reply • Share ›

Guest • 8 months ago

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

• Reply • Share ›

Underground • 8 months ago

Above program would break if the SLL contains loop !

• Reply • Share ›

Guest • 8 months ago

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

• Reply • Share ›

hemanthreddy • 9 months ago

n value less than zero case is not handled here

• Reply • Share ›

rahulcynosure • a year ago

```
void printnthfromlast(struct Node * head,int n)
{
struct Node * temp = head;
int count=0;
while(head)
{
if(count>=n)
temp=temp->next;
```

temp=temp->next;
head=head->next;
count++;
}
if(count>=n)
printf("%d",temp->data);
}

1 • Reply • Share ›

neelabh ➦ rahulcynosure • a year ago

Explanation of above code: In your coding first you initialize temp=heac
in while loop you are incrementing head node until meet to last node. ir
condition will increment temp pointer until count>=n means it will maint
temp pointer. after reaching last node temp will indicate the Nth node fr
Very nice solution given by you.

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

1 • Reply • Share ›

Kumar • 2 years ago

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

• Reply • Share ›

Mohinder ➦ Kumar • 8 months ago

Only if count is 1.

• Reply • Share ›

Kartik ➦ Kumar • 2 years ago

Please take a closer look at the program. The condition is correct.

• Reply • Share ›

GeeksforGeeks • 3 years ago

@Anuj Bansal & Agniswar: Thanks for inputs. We have added method 1 to the

• Reply • Share ›

Nishant Kumar → GeeksforGeeks • 2 years ago

Time complexity should be O(n^2) for recursive method. Am i right ?

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

• Reply • Share ›

Kartik → Nishant Kumar • 2 years ago

It is a simple O(n) program.

• Reply • Share ›

Anuj Bansal • 3 years ago

One recursive solution could be as follows:

```
void Nth(NODEPTR node, int n) {
   static int i =0;
   if(node == NULL)
      return;
   Nth(node->next, n);
   if(++i == n)
      printf("%d", node->info);
}
```

1 • Reply • Share ›

zaid → Anuj Bansal • 2 years ago

@anuj bansal:what is the time complexity for the above code??????

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

• Reply • Share ›

Nishant Kumar → zaid • 2 years ago

Time complexity should be O(n^2) for recursive method. Am i ri

```
/* Paste your code here (You may delete these lines if r
```

• Reply • Share ›

Yogesh Batra → Nishant Kumar • 2 years ago

No, Time complexity is Linear. It traverses the list twice.
O(2n)~O(n)

```
/* Paste your code here (You may delete these lin
```

• Reply • Share ›

kartik → Anuj Bansal • 3 years ago

@Anuj Bansal: Thanks for sharing the recursive solution. This solution then gets the nth node.

• Reply • Share ›

Agniswar • 3 years ago

I think we can solve this problem in this way too-
Nth node from the end essentially means (len-N)th node from the front where l
So,we simply need one pointer to find the (len-N)th node instead of the 2 point
tell me if i am wrong !

• Reply • Share ›

kartik → Agniswar • 3 years ago

@Agniswar: Generally, we only have header node of a given Linked list the length first by traversing the complete list and then get the (len - N)t

• Reply • Share ›

Subscribe Add Disqus to your site