Write a function to reverse a linked list

Iterative Method
Iterate trough the linked list. In loop, change next to prev, prev to current and current to next.

Implementation of Iterative Method

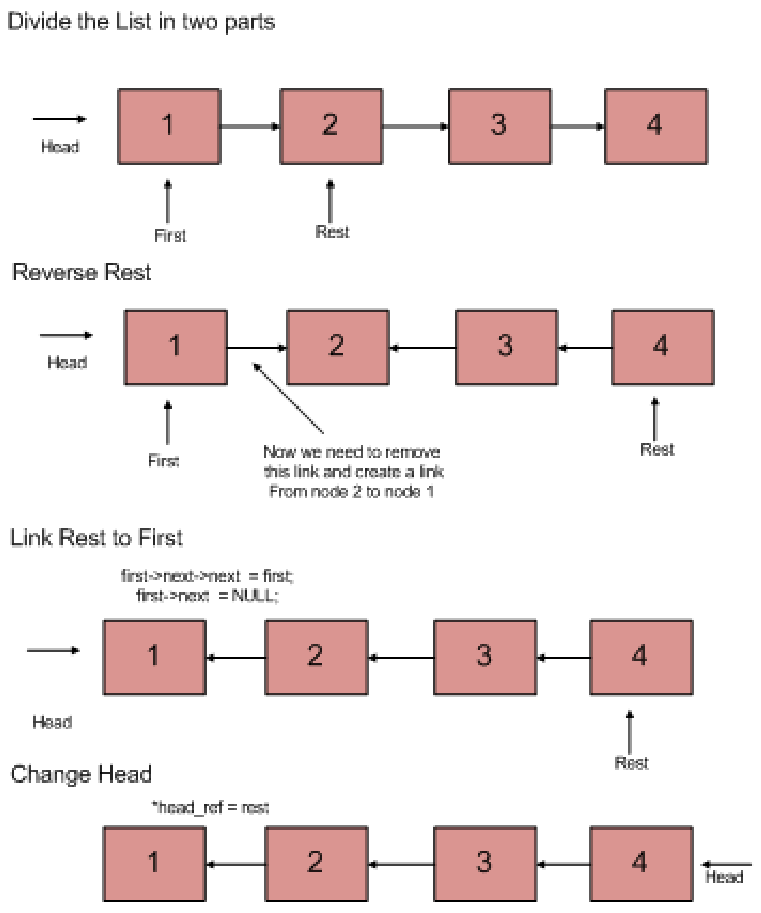
```
#include<stdio.h>
#include<stdlib.h>

/* Link list node */
struct node
```

```
{
    int data;
    struct node* next;
};

/* Function to reverse the linked list */
static void reverse(struct node** head_ref)
{
    struct node* prev   = NULL;
    struct node* current = *head_ref;
    struct node* next;
    while (current != NULL)
    {
        next  = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
    *head_ref = prev;
}

/* Function to push a node */
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;
}

/* Function to print linked list */
void printList(struct node *head)
{
    struct node *temp = head;
    while(temp != NULL)
    {
        printf("%d  ", temp->data);
        temp = temp->next;
    }
}

/* 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);
     push(&head, 85);

     printList(head);
     reverse(&head);
     printf("\n Reversed Linked list \n");
     printList(head);
     getchar();
}
```

Time Complexity: O(n)
Space Complexity: O(1)

Recursive Method:

```
   1) Divide the list in two parts - first node and rest of the linked list.
   2) Call reverse for the rest of the linked list.
   3) Link rest to first.
   4) Fix head pointer
```


Divide the List in two parts
Head
1
2
3
4
First
Rest
Reverse Rest
Head
1
2
3
4
First
Now we need to remove this link and create a link From node 2 to node 1
Rest
Link Rest to First
1
2
3
4
Head
Rest
Change Head
1
2
3
4
Head

```
first->next->next = first;
first->next = NULL;
```

```
*head_ref = rest
```

```
void recursiveReverse(struct node** head_ref)
{
    struct node* first;
    struct node* rest;
    /* empty list */
    if (*head_ref == NULL)
       return;

    /* suppose first = {1, 2, 3}, rest = {2, 3} */
    first = *head_ref;
    rest  = first->next;

    /* List has only one node */
    if (rest == NULL)
       return;

    /* reverse the rest list and put the first element at the end */
    recursiveReverse(&rest);
    first->next->next  = first;

    /* tricky step -- see the diagram */
    first->next  = NULL;

    /* fix the head pointer */
    *head_ref = rest;
}
```

Time Complexity: O(n)
Space Complexity: O(1)

References:
http://cslibrary.stanford.edu/105/LinkedListProblems.pdf

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

Like 41 Tweet 0 +1 6

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

201 Comments GeeksforGeeks Login

Recommend 2 Share Sort by Newest

Join the discussion…

prk • 10 days ago

Since java is pass by value, we need to pass the head pointer at each recursive level. Complete linked list reversal implementation iterative and recursive:

https://ideone.com/YpgEcx

• Reply • Share ›

shellk007 • 13 days ago

```
void reverse_list(node **head)

{

node *head1=NULL,*temp= NULL;

if(*head==NULL)

{

printf("\n List is empty");

return;

}

while((*head)!=NULL)

{

temp= head1;
```

see more

• Reply • Share ›

Nikhil Jindal • 16 days ago

why doesn't the head_ref change in every recursive call?

• Reply • Share ›

thunderWiring • 20 days ago

hey, i used some of the information in this post regarding the recursion and used it for a post in my website: https://thunderwiring.wordpres...

please let me know if you confirm that. Thanks :)

1 • Reply • Share ›

prashant chaudhary • 21 days ago

public class Link {

int a;

Link Next;

public Link(int i){

a=i;

}

}

----

public class LinkList {

Link First = null;

public void insertFirst(int a){

see more

• Reply • Share ›

nyam • a month ago

Method 2:

```
void Reverse(Node** head)
{
assert(head != NULL);
Node* prev = NULL;
Node* next = *head;
while (next != NULL)
{
*head = next;
next = next->next;
(*head)->next = prev;
prev = *head;
}
}
```

• Reply • Share ›

nyam • a month ago

Method 1:

```
void Reverse(Node** head)
```

```
{
assert(head != NULL);

Node* prev = NULL;
Node* next = *head;

while (next != NULL)
{
*head = next;
next = next->next;
(*head)->next = prev;
prev = *head;
}
}
```

• Reply • Share ›

Rohin Mehra • a month ago

Simpler recursive function to reverse the list.

```
struct node* recursiveReverse(struct node *curr, struct node *prev)
{
if(curr == NULL) return prev;
struct node *last_node = recursiveReverse(curr->next, curr);
curr->next = prev;
return last_node;
}
```

http://ideone.com/xzg6ir

• Reply • Share ›

Guest • a month ago

Redundant ==

http://www.geeksforgeeks.org/g...

Please merge both considering comments on them.

• Reply • Share ›

munjal • a month ago

code : http://geekforgeekss.blogspot....

• Reply • Share ›

Praveen Pandey • a month ago

How can *head_ref = rest; set head to last node?? As rest is variable in recursion. Please help.

• Reply • Share ›

Nishant Verma ↗ Praveen Pandey • a month ago

for ease of understanding use reference instead of double pointers ...
which is not available in c

so your method will become something like

```
void recReverse(struct node *&root){
if(root==NULL){
return ;
}

struct node *first =root;
struct node *rest =root->next;

if(rest==NULL){
return ;
}

recReverse(rest);
first->next->next=first;
```

see more

• Reply • Share ›

David • 2 months ago

Can someone explain the first->next->next part?

• Reply • Share ›

abhi011 ↗ David • a month ago

lets say the list is 1->2->3->4->5, now in the beginning first->1 and rest->2, then first->2 and rest->3 and so on as we call the function recursively.

at the end first->4 and rest->5. now first->next points to 5, first->next->next points to Null. we need to make 5 point to 4, so we do first->next->next=first, which makes 5 point 4.

this loop will break and next up is first->3 and rest->4. the above process continues.

3 • Reply • Share ›

Guest • 2 months ago

In method 1
Why does it give an error if I just put the line next = current->next; in the end while initializing it above as follows:

```
static void reverse(struct node** head_ref)
{
```

```
struct node* prev = NULL;
struct node* current = *head_ref;
struct node* next= current->next;
while (current != NULL)
{
current->next = prev;
prev = current;
current = next;
next = current->next;
}
*head_ref = prev;
}
```

• Reply • Share ›

PK • 2 months ago

1) create a new Head pointer
2) Read elements of linked list and add at head of newHead
3) Replace Oldhead= newHead

1 • Reply • Share ›

mehboob • 3 months ago

if we use stact to do above problem.Will that be correct???

1 • Reply • Share ›

Mohammad Zafarul Quadri • 3 months ago

This C code works.

```
------------------------------------------------------------------------------------
void ReverseListUtil(NODE **newHead, NODE *prev, NODE *cur)
{
if(cur == NULL)
{
*newHead = prev;
return;
}
ReverseListUtil(&(*newHead), cur, cur->next);
cur->next = prev;
}

NODE *ReverseList(NODE *head)
{
NODE *newHead = NULL;
ReverseListUtil(&newHead, NULL, head);
return newHead;
}
```

Guest • 3 months ago

C++ Code:

```
void ReverseListUtil(NODE **newHead, NODE *prev, NODE *cur)

{

if(cur == NULL)

{

*newHead = prev;

return;

}

ReverseListUtil(&(*newHead), cur, cur->next);

cur->next = prev;

}
```

see more

Rocky • 3 months ago

The attached document is just enough .
Thanks for sharing it .

Narendra • 3 months ago

How about below code

```
------------------------------------------------------------
List * reverseListUtil(List *t, List **newhead)
{
if(!t)
return NULL;

if(t->next == NULL)
return *newhead=t;

reverseListUtil(t->next, newhead)->next=t;
t->next=NULL;

return t;
}
void reverselist(List **t)
{
reverseListUtil(*t, t);
}
```

• Reply • Share ›

Atul Yadav • 3 months ago

```
class Node
{
int v;
Node next;

Node(int v)
{
this.v=v;
next=null;
}
}

class LinkedList
{
Node s;
public int countNodes(Node s)
{
int c=0;
```

see more

• Reply • Share ›

Guest • 3 months ago

```
class

Node

{

{

int v;

Node

Node

next;
```

Node(

Node(

int v)

see more

^ | ⌄ • Reply • Share ›

Guest • 3 months ago

{{{class

Node

{

{

int v;

Node

Node

next;

Node(

Node(

int v)

see more

^ | ⌄ • Reply • Share ›

popina • 4 months ago

#include <iostream>

using namespace std;

struct Node {

int info;

Node* next;

Node() = default;

Node(int _info) : info(_info), next(nullptr) {}

};

class Queue {

Node* begin = nullptr,* end = nullptr;

public:

see more

1 • Reply • Share ›

Guest • 4 months ago

```
void modify(List **head){
List *stack = null, *new = null;
while(head != null && head->next != null){
new = head->next;
head->next = new->next;
new->next = stack;
stack=new;
}
head->next = stack;

}
```

• Reply • Share ›

harendra • 5 months ago

/* fix the head pointer */
*head_ref = rest;
how it will fix it will change every time please explain anyone

1 • Reply • Share ›

Kislaya Sharma harendra • 4 months ago

It's correct.
Try to understand it with this example for 1->2->3

The recursion goes like

first=1 rest = 2,first =2 rest =3,first =3 rest=NULL
returns back to previous step first =2 rest =3
now first->next->next=first ie. 3->next=2 and
*head_ref=rest ie. head_ref=3
(observe that head_ref in this step was rest in previous step and it was passed by reference hence the changes are reflected in previous step too,hence rest in previous step gets modified)
so now when it goes to previous step ie. first =1 and rest =2

first->next->next=first ie. 2->next=1 and *head_ref=rest(rest was made 3 as it was passed by reference) and hence head_ref = 3 again.

In this way the linked list 1->2->3 becomes 1<-2<-3

18 • Reply • Share ›

Kislaya Sharma • 5 months ago

For all those who think there is a problem with the last step ,they're wrong.
The rest is passed by reference till it reaches the last node and is assigned as head pointer.
After that while coming back the changes made in head are reflected in the rest of previous step in recursion because rest was passed by reference as head.
So in each step going back in recursion the rest value gets transferred and hence it remains same ie. the value head_ref was given in last stage of recursion.
I hope it clears many doubts.

7 • Reply • Share ›

anony Kislaya Sharma • 3 months ago

Is the below condition needed ? since rest will points to null bcoz of this headref will also chnges to null right .
if (rest == NULL) {
*head = first;
return;
}

• Reply • Share ›

Kislaya Sharma anony • 3 months ago

no extra condition is needed.

• Reply • Share ›

coder.girl Kislaya Sharma • 4 months ago

pls explain with example 1->2->3

• Reply • Share ›

Kislaya Sharma coder.girl • 4 months ago

The recursion goes like

first=1 rest = 2,first =2 rest =3,first =3 rest=NULL
returns back to previous step first =2 rest =3
now first->next->next=first ie. 3->next=2 and
*head_ref=rest ie. head_ref=3
(observe that head_ref in this step was rest in previous step and it was passed by reference hence the changes are reflected in previous step too,hence rest in previous step gets modified)
so now when it goes to previous step ie. first =1 and rest =2

first->next->next=first ie. 2->next=1 and *head_ref=rest(rest was made 3 as it was passed by reference) and hence head_ref = 3 again.

In this was the linked list 1->2->3 becomes 1<-2<-3

4 • Reply • Share ›

Ishan • 5 months ago

```
struct node * rr(struct node * head,struct node * prev){

if(head== NULL) return prev;

struct node * x =rr(head->next,head);

head->next = p;

return x;

}

int main(){
struct node *m=NULL;
push(&m,5); push(&m,6); push(&m,7); push(&m,8); push(&m,9);
m=rr(m,NULL);
print(m);
return 0;
}
```

• Reply • Share ›

PJ • 5 months ago

```
struct node* reverse(struct node * curr, struct node * prev)
{
if ( curr == NULL )
{
return prev;
}
struct node * temp = curr->next;
curr->next = prev;
struct node * head = reverse(temp, curr);
return head;
}
```

in main:

```
struct node * newHead = reverse(head->next, head);
head->next = NULL;
```

2 • Reply • Share ›

minion ➦ PJ • 5 months ago

better

• 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

1 • Reply • Share ›

sheetal pathak • 6 months ago

During last step for recursion, the first is 1 and the rest is {2,3,4}. The rest is reversed. As per my understanding, due to *head_ref = rest, the head does not point to 4. Can you please explain?

1 • Reply • Share ›

Kislaya Sharma ➦ sheetal pathak • 5 months ago

For all those who think there is a problem with the last step ,they're wrong.
The rest is passed by reference till it reaches the last node and is assigned as head pointer.
After that while coming back the changes made in head are reflected in the rest of previous step in recursion because rest was passed by reference as head.
So in each step going back in recursion the rest value gets transferred and hence it remains same ie. the value head_ref was given in last stage of recursion.
I hope it clears many doubts

2 • Reply • Share ›

Ansuraj Khadanga ➦ Kislaya Sharma • a month ago

When people understand -"the changes made in head are reflected in the rest of previous step in recursion because rest was passed by reference as head"- this part, the problem can be understood.
Thanks Kislaya for the explanation.

• Reply • Share ›

Ambar Maini • 6 months ago

what is the use of static here

```
static void reverse(struct node** head_ref)
{
struct node* prev = NULL;
struct node* current = *head_ref;
struct node* next;
while (current != NULL)
{
next = current->next;
current->next = prev;
prev = current;
current = next;
}
*head_ref = prev;
}
```

• Reply • Share ›

Ansuraj Khadanga ➦ Ambar Maini • a month ago

Unless we want to reuse the function, in another file, the "static" keyword is not required.
You can go through:
http://www.geeksforgeeks.org/w...

• Reply • Share ›

Charlie ➦ Ambar Maini • 6 months ago

From: http://stackoverflow.com/quest...

We use static here to keep the function local, so any modules (files) outside of the one defined here can't see it.

• Reply • Share ›

Surya ➦ Charlie • 3 months ago

Instead of word "local".....scope"limited" to the file suits better

• Reply • Share ›

AnyOne • 6 months ago

In recursive method, I am unable to understand why only first is getting updated after "first->next->next = first; ", why not rest pointer is getting updated,

1 • Reply • Share ›

prianca__ AnyOne • 6 months ago

1 -> 2 -> 3 -> 4 -> 5

cuz we have to assign *head_ptr = rest ie 5, if we would update it, then due to recusrion, head_ptr would also change

2 • Reply • Share ›

AnyOne AnyOne • 6 months ago

Please reply asap. Thanks in advance

• Reply • Share ›

AnyOne AnyOne • 6 months ago

If rest pointer is not getting updated then how first pointer is getting updated.

• Reply • Share ›

saurav AnyOne • 5 months ago

rest=first->next;

its getting updated.

• Reply • Share ›

Karan Verma • 7 months ago

static void recursiveReverse(struct node **head_ref)

{

struct node *temp = *head_ref;

if(temp->next == NULL)

{

*head_ref = temp;

return;

}

recursiveReverse(&temp->next);

struct node *frw = temp;

```
frw->next = temp;

temp->next = NULL;

}
```

• Reply • Share ›

Load more comments

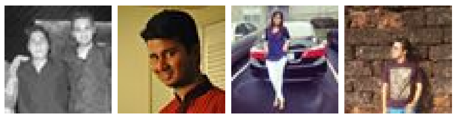
Subscribe Add Disqus to your site Privacy

-

GeeksforGeeks

Like

93,235 people like GeeksforGeeks.



Facebook social plugin

-
-
-
    - 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

Follow @GeeksforGeeks Subscribe

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 · 4 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

-