Pairwise swap elements of a given linked list

Given a singly linked list, write a function to swap elements pairwise. For example, if the linked list is 1->2->3->4->5 then the function should change it to 2->1->4->3->5, and if the linked list is 1->2->3->4->5->6 then the function should change it to 2->1->4->3->6->5.

METHOD 1 (Iterative)
Start from the head node and traverse the list. While traversing swap data of each node with its next node’s data.

```
/* Program to pairwise swap elements in a given linked list */
#include<stdio.h>
#include<stdlib.h>

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

/*Function to swap two integers at addresses a and b */
void swap(int *a, int *b);

/* Function to pairwise swap elements of a linked list */
void pairWiseSwap(struct node *head)
{
  struct node *temp = head;

   /* Traverse further only if there are at-least two nodes left */
  while (temp != NULL && temp->next != NULL)
  {
    /* Swap data of node with its next node's data */
    swap(&temp->data, &temp->next->data);
```

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

```
    /* Move temp by 2 for the next pair */
    temp = temp->next->next;
  }
}

/* UTILITY FUNCTIONS */
/* Function to swap two integers */
void swap(int *a, int *b)
{
  int temp;
  temp = *a;
  *a = *b;
  *b = temp;
}

/* Function to add a node at the begining of Linked List */
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 nodes in a given linked list */
void printList(struct node *node)
{
  while (node != NULL)
  {
    printf("%d ", node->data);
    node = node->next;
  }
}

/* Druver program to test above function */
int main()
{
  struct node *start = NULL;
```

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

```c
  /* The constructed linked list is:
   1->2->3->4->5 */
  push(&start, 5);
  push(&start, 4);
  push(&start, 3);
  push(&start, 2);
  push(&start, 1);

  printf("\n Linked list before calling  pairWiseSwap() ");
  printList(start);

  pairWiseSwap(start);

  printf("\n Linked list after calling  pairWiseSwap() ");
  printList(start);

  getchar();
  return 0;
}
```

Time complexity: O(n)

METHOD 2 (Recursive)

If there are 2 or more than 2 nodes in Linked List then swap the first two nodes and recursively call for rest of the list.

```c
/* Recursive function to pairwise swap elements of a linked list */
void pairWiseSwap(struct node *head)
{
  /* There must be at-least two nodes in the list */
  if(head != NULL && head->next != NULL)
  {
    /* Swap the node's data with data of next node */
    swap(&head->data, &head->next->data);

    /* Call pairWiseSwap() for rest of the list */
    pairWiseSwap(head->next->next);
  }
}
```

Time complexity: O(n)

The solution provided there swaps data of nodes. If data contains many fields, there will be many swap operations. See this for an implementation that changes links rather than swapping data.

Please write comments if you find any bug in above code/algorithm, 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

2 Tweet 0 0

705 Subscribe

Recent Comments

Abhi You live US or India?

Google (Mountain View) interview · 49 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

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

40 Comments **GeeksforGeeks**

Sort by Newest ▾

Join the discussion…

Atreyee Ray · 22 days ago

the last node can't be read by this method since the temp!=NULL is satisfied b
>next!=NULL isn't satisfied although it is at the end. this one probbaly works pl

```
node *pairwiseswap(node *header)
{
node *temp=header;
int t;
while(temp->next!=NULL && temp!=NULL)
{
t=temp->data;
temp->data=temp->next->data;
temp->next->data=t;
if(temp->next->next!=NULL)
/*the condition temp!=NULL is staisfied for last node but the condition of temp-
exceds the list, thus not knowing what the next adrress containsso segmentat
temp=temp->next->next;
else
temp=temp->next;
}
return header;
}
```

^ | ⌄ • Reply • Share ›

AMIT JAMBOTKAR ➦ Atreyee Ray • 22 days ago

AMIT JAMBOTKAR → Atreyee Ray • 22 days ago

```
while(temp->next!=NULL && temp!=NULL)
```

change this to

```
while( temp!=NULL && temp->next!=NULL)
```

and NO Need of this code

```
if(temp->next->next!=NULL)
/*the condition temp!=NULL is staisfied for last node but the condition
as it exceds the list, thus not knowing what the next adrress containss
temp=temp->next->next;
else
temp=temp->next;
}
```

just

```
temp=temp->next->next;
```

is enough

because for Total number Nodes are ODD temp!=NULL becomes fals
temp->next!=NULL will become false

^ | v • Reply • Share ›

Atreyee Ray → Atreyee Ray • 22 days ago

the problem that i mentioned arises when it is even number

^ | v • Reply • Share ›

Niks → Atreyee Ray • 22 days ago

I am sorry but I dont see a problem with the above code in the
please give an example where it will fail.

Reply • Share ›

VaraKalyan M • 23 days ago

I think , a final step is missed in both the routines.
For ex : 10 ->20 ->30->40->50
After pass 1: 20 ->10 ->30 ->40 ->50.
pass 2: 20 ->10 ->40 ->30 -> 40 ->50.
Here , not only reversing the pair linking it to the previous pair also important. i.

Reply • Share ›

GeeksforGeeks Mod → VaraKalyan M • 23 days ago

Thanks for sharing your thoughts. Could you please provide more deta
program doesn't produce expected output would help.

Reply • Share ›

sijayaraman • 3 months ago

```
void swap_pair(struct node* root)
{
if(root==NULL || root->link==NULL)
return;
struct node* current = root;
struct node* next = current->link;
while(current->link!=NULL)
{
swap(¤t->data,&next->data);
if(current->link == NULL || next->link==NULL)
{
break;
}
current=next->link;
next=current->link;
}
}
```

Reply • Share ›

Himanshu Dagar • 3 months ago

can refer to below link to see code in combined way

http://ideone.com/gcHwba

Reply • Share ›

Himanshu Dagar • 3 months ago

(y)
Simple technique used in recursion (good way to learn recursion)

Reply • Share ›

github • 3 months ago

```
#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node *next;

};

void create_n(struct node **ll,int num)

{

struct node *temp;
```

```
temp=(struct node *)malloc(sizeof(struct node));
```

see more

• Reply • Share ›

Sameer • 4 months ago

http://codingrecipies.blogspot...

http://codingrecipies.blogspot...

• Reply • Share ›

Marsha Donna • 4 months ago

```
void pairwise_swap_ele(struct node **head)
{
int var=0;
struct node *temp=*head;
while(temp!=NULL && temp->link!=NULL)
{
var=temp->data;
temp->data=temp->link->data;
temp->link->data=var;
temp=temp->link->link;
}
}
```

• Reply • Share ›

Amit Kumar • 4 months ago

```
struct Node* add_Recursively(struct Node *itr){

struct Node *prev,*temp;

if(itr == NULL){
```

```
return NULL;

}

prev = add_Recursively(itr->next->next);

temp = itr->next;

itr->next->next = itr;

itr->next=prev;

return temp;

}
```

see more

Reply • Share ›

guest • 4 months ago

Here is the code which swap the nodes rather than data.
Imagine the case where data in each nodes contains many field, rather than s
consider to swap the nodes .

```
void pairWiseSwap(struct node **head)
{
if(*head == NULL || (*head)->next==NULL)
return;

struct node *prev = *head;
struct node *curr = (*head)->next;
*head = curr; // this will be head
struct node *next;
while(curr != NULL)
{
```

```
next = curr->next;
curr->next = prev;
if(next == NULL) { prev->next = NULL; break; }
```

see more

• Reply • Share ›

Guest • 8 months ago

// swapping pairwise using pointers.

```
struct node *pairwise_swap(struct node *head)
{
struct node *temp, *t, *p =NULL;
if(!head || !head->next)
return head;
else
{
temp = head;
while(temp && temp->next)
{
if(!p)
head = temp->next;
else
p->next=temp->next;

t=temp->next;
temp->next=temp->next->next;
t->next =temp;

p =temp;
temp= temp->next;
}
return head;
}
}
```

⌃ | ⌄ • Reply • Share ›

Kush Pandey • 10 months ago

According to me swapping data is not right as the question is to swap the nod
algorithm of reversing k nodes . the program is given below

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

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

/* Reverses the linked list in groups of size k and returns the
   pointer to the new head node. */
struct node *reverse (struct node *head, int k)
{
    struct node* current = head;
```

see more

⌃ | ⌄ • Reply • Share ›

Kaushik • 11 months ago

```
 void swapAlt(struct node** head)
{
        struct node* temp=*head;
        if(temp->next!=NULL)
        if(temp->next!=NULL)
        {
                struct node *a_prev,*a,*b_prev,*b,*b_next;
                a=b=temp;
                a_prev=NULL;
                while(b!=NULL && b->next!=NULL)
                {
                        b=b->next;
                        b_prev=a;
                        b_next=b->next;

                        a->next=b_next;
                        if(a_prev!=NULL)
                        {
                                a_prev->next=b;
```

see more

^ | ⌄ • Reply • Share ›

Sumit • a year ago

This code just exchanges the pointers in place of swapping the values

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

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

void push(struct node **head, int data)
{
  if ((*head) == NULL)
```

```
{
  (*head) = malloc(sizeof(struct node));
  (*head)->next = NULL;
  (*head)->data = data;
}
```

see more

⌃ | ⌄ • Reply • Share ›

Murali S Iyengar ➦ Sumit • 4 months ago

@Sumit
When there are odd number of elements, your code removes last elen

⌃ | ⌄ • Reply • Share ›

Abhishek Mishra • a year ago

What if you can only change the pointers. Say data is huge.

⌃ | ⌄ • Reply • Share ›

Abhishek Mishra • a year ago

What if you can only change the pointers. Say data is huge.

⌃ | ⌄ • Reply • Share ›

anonymous • a year ago

We can solve the question by changing pointers.
Both iterative and recursive methods are applicable.
That would be a very good solution as swapping records could be tedious.

⌃ | ⌄ • Reply • Share ›

anonymous ➦ anonymous • a year ago

In fact,it is a special case of "reverse every k nodes of a linked list" witl

⌃ | ⌄ • Reply • Share ›

whizkid08 • a year ago

```
 void swapPairsRec(struct Node *first)
{
        struct Node *second = first->next;
        int temp;

        if(first == 0 || second == 0)
        return;

        temp = second->data;
        second->data = first->data;
        first->data = temp;

        swapPairsRec(second->next);
}
```

⌃ | ⌄ • Reply • Share ›

Rajdeep • 2 years ago

May be for the cure of cough, I am giving a medicine of TB.
But it can also be solved by algo posted by geeks for

"Reverse a Linked List in groups of given size"

http://www.geeksforgeeks.org/a...
Here the size would be 2

⌃ | ⌄ • Reply • Share ›

KARTHIKEYAN.V.B • 2 years ago

void pairWiseSwap(struct node **head)
{

```
{
struct node *p = *head,*q=(*head)->next,*r=(*head)->next,*prev=NULL;
while(p && q)
{
if(prev) prev->next=q;
p->next=q->next;
q->next=p;
prev=p;
p=p->next;
if(p)
q=p->next;
}
*head=r;
}
```

• Reply • Share ›

SK • 2 years ago

Hi,
Can u help me to achive this using C# code...

how can i do the this in C#...

• Reply • Share ›

SK ➦ SK • 2 years ago

Guys,
Can any one help me....

• Reply • Share ›

FB ➦ SK • 2 years ago

can you not able to change this logic in C#.
shame on you.

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

^ | ⌄ • Reply • Share ›

PsychoCoder • 2 years ago

This is not the value swapping. Basically pairwise pointer swapping.

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

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

node* newNode (int data) {
  node *temp ;
  temp = (node *) malloc (sizeof(node)) ;
  temp->data = data ;
  temp->next = NULL ;
  return temp ;
}
```

see more

^ | ⌄ • Reply • Share ›

Fanendra • 3 years ago

Please find the java code for the same.

```
        public void pairwiseSwap() {
                startNode = pairwiseSwap(this.startNode);
        }
```

```
public Node pairwiseSwap(Node start) {
        Node current = start;
        if (current.next != null) {
                Node next = current.next;
                next.next = current;
                if (current.next.next != null) {
                        current.next =    pairwiseSwap(curren
                } else {
                        current.next = null;
                }
                return next;
        }
        return current;
}
```

⌃ | ⌄ • Reply • Share ›

sharat • 3 years ago

Instead of swapping the just the data, to be more generic, We can use the sol
and use k = 2. This would swap the nodes instead of the data.

⌃ | ⌄ • Reply • Share ›

sharat ➦ sharat • 3 years ago

http://geeksforgeeks.org/?p=80...

⌃ | ⌄ • Reply • Share ›

Sambasiva • 4 years ago

```
list pairwise(list l)
{
        Node *newlist = l->next;
        Node *temp, *prev = NULL;

        if(!newlist) return l;

        while(l && (temp = l->next))
        {
                if(prev)
                        prev->next = temp;
                prev = l;
                l->next = temp->next;
                temp->next = l;
                l = l->next;
        }

        return newlist;
}
```

^ | ˅ • Reply • Share ›

Sambasiva ➦ Sambasiva • 4 years ago

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

struct Node
{
        int data;
        struct Node *next;
        struct Node *arb;
};

typedef struct Node Node;
typedef struct Node Node;
typedef Node* list;

Node *appendNode(list l, int elm);

list intersect(list l1, list l2)
{
```

see more

^ | ⌄ • Reply • Share ›

Sambasiva ➦ Sambasiva • 4 years ago

```
list pairwise(list l)
{
        Node *newlist = l->next;

        if(!newlist) return l;

        Node *p = NULL, *n;

        while( l && (n = l->next))
        {
                if(p)
                        p->next = n;

                p = l;
                l = n->next;
                n->next = p;
        }
        if(p)
          p->next = NULL;
        return newlist;
}
```

Reply • Share ›

Sambasiva Sambasiva • 4 years ago

```
list pairwise(list l)
{
        Node *newlist = l->next;

        if(!newlist) return l;

        Node *p = NULL, *n;

        while( l && (n = l->next))
        {
                if(p)
                        p->next = n;

                p = l;
                l = n->next;
                n->next = p;
        }
        if(p)
          p->next = l;

        return newlist;
}
```

Reply • Share ›

Sam • 4 years ago

Real pointer switching version using prev, cur and temp pointers.

```
public static LinkedList PairwiseSwapPointer(LinkedList head)
       {
           LinkedList cur = head;
           LinkedList prev = null;
           bool bFirst = true;
           while ((null != cur) && (null != cur.Next))
           {
               if (bFirst)
               {
                   head = cur.Next;
                   bFirst = false;
               }

               LinkedList pTemp = cur.Next;
               cur.Next = cur.Next.Next;
               pTemp.Next = cur;
```

see more

^ | ⌄ • Reply • Share ›

g33k ➦ Sam • 3 years ago

I was about the type the same algo after seing people some sort of cra

^ | ⌄ • Reply • Share ›

aravindh ➦ Sam • 4 years ago

Thnx sam..s nt a recursive version much simpler ???

^ | ⌄ • Reply • Share ›

✉ Subscribe D Add Disqus to your site