C DOPE (/blog/)

# C++ : Concatenating two linked lists in C++

May 30, 2017 C++ (/blog/tag/cpp/?tag=cpp) LINKED LIST (/blog/tag/linked-list/?tag=linked-list) DATA STRUCUTRE (/blog/tag/data-strucutre/?tag=data-strucutre) 12536

Become an Author

(/blog/submit-article/)

**Download Our App.**

(https://play.google.com/store/apps/details?id=com.blogsdope&pcampaignid=MKT-Other-global-all-co-prtnr-py-PartBadge-Mar2515-1)

## Previous:

1. Linked lists in C++ (Singly linked list) (https://www.codesdope.com/blog/article/c-linked-lists-in-c-singly-linked-list/)

2. Linked list traversal using while loop and recursion in C++ (https://www.codesdope.com/blog/article/linked-list-traversal-using-loop-and-recursion-in-/)

Make sure that you are familiar with the concepts explained in the post(s) mentioned above before proceeding further.

We will proceed further by taking the linked list we made in the previous post

```
#include <iostream>

using namespace std;

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

class linked_list
{
private:
    node *head,*tail;
public:
    linked_list()
    {
        head = NULL;
        tail = NULL;
    }

    void add_node(int n)
    {
        node *tmp = new node;
        tmp->data = n;
        tmp->next = NULL;

        if(head == NULL)
        {
            head = tmp;
            tail = tmp;
        }
        else
        {
            tail->next = tmp;
            tail = tail->next;
        }
    }
};

int main()
{
    linked_list a;
    a.add_node(1);
    a.add_node(2);
    return 0;
}
```

Concatenating or joining two linked lists is not at all a difficult task. We just need to follow some very simple steps and the steps to join two lists (say ‘a’ and ‘b’) are as follows:

1. Traverse over the linked list ‘a’ until the element next to the node is not NULL.

2. If the element next to the current element is NULL (a->next == NULL) then change the element next to it to 'b' (a->next = b).

That's it. Let's code up these steps.

.

```
void concatenate(struct node *a,struct node *b)
{
    if (a->next == NULL)
        a->next = b;
    else
        concatenate(a->next,b);
}
```

Here, we are traversing over the article using recursion as explained in the article "Linked list traversal using while loop and recursion (https://www.codesdope.com/blog/article/linked-list-traversal-using-loop-and-recursion-in-/)". We are firstly checking if the next node (a->next) is NULL or not. If it is NULL, then we are just changing its value from NULL to 'b' (a->next = b) and if it is not then we are calling the 'concatenate' function again with the next element to traverse over the list.

So, the whole code is:

```
#include <iostream>

using namespace std;

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

class linked_list
{
private:
    node *head,*tail;
public:
    linked_list()
    {
        head = NULL;
        tail = NULL;
    }

    void add_node(int n)
    {
        node *tmp = new node;
        tmp->data = n;
        tmp->next = NULL;

        if(head == NULL)
        {
            head = tmp;
            tail = tmp;
        }
        else
        {
            tail->next = tmp;
            tail = tail->next;
        }
    }

    node* gethead()
    {
        return head;
    }

    static void display(node *head)
    {
        if(head == NULL)
        {
            cout << "NULL" << endl;
        }
        else
        {
            cout << head->data << endl;
            display(head->next);
        }
    }

    static void concatenate(node *a,node *b)
```

```
        {
            if( a != NULL && b!= NULL )
            {
                if (a->next == NULL)
                    a->next = b;
                else
                    concatenate(a->next,b);
            }
            else
            {
                cout << "Either a or b is NULL\n";
            }
        }
};

int main()
{
    linked_list a;
    a.add_node(1);
    a.add_node(2);
    linked_list b;
    b.add_node(3);
    b.add_node(4);
    linked_list::concatenate(a.gethead(),b.gethead());
    linked_list::display(a.gethead());
    return 0;
}
```

## The simpler way

We can use the above method even when we don't have the record of the 'tail'. When we keep the record of the 'tail', we can simply do `a->tail->next = b->head`

## Next:

1. Inserting a new node in a linked list in C++ (https://www.codesdope.com/blog/article/inserting-a-new-node-to-a-linked-list-in-c/)
2. Deletion of a given node from a linked list in C++ (https://www.codesdope.com/blog/article/c-deletion-of-a-given-node-from-a-linked-list-in-c/)

Liked the post?