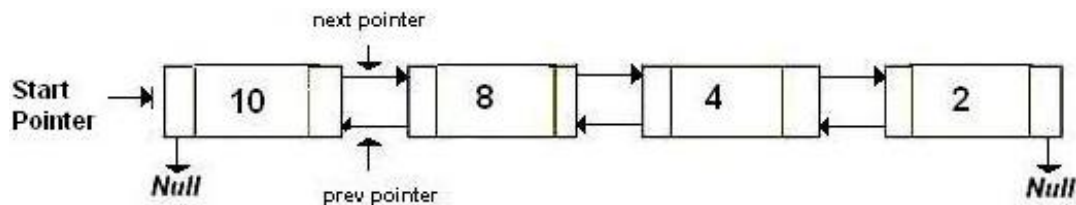


Reverse a Doubly Linked List

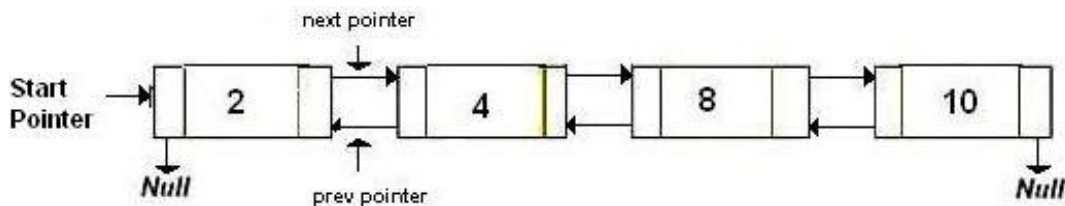
Write a C function to reverse a given Doubly Linked List

See below diagrams for example.

(a) Original Doubly Linked List



(b) Reversed Doubly Linked List



Here is a simple method for reversing a Doubly Linked List. All we need to do is swap prev and next pointers for all nodes, change prev of the head (or start) and change the head pointer in the end.

```
/* Program to reverse a doubly linked list */
#include <stdio.h>
#include <stdlib.h>

/* a node of the doubly linked list */
struct node
{
    int data;
    struct node *next;
    struct node *prev;
```

```
struct node *prev;
};

/* Function to reverse a Doubly Linked List */
void reverse(struct node **head_ref)
{
    struct node *temp = NULL;
    struct node *current = *head_ref;

    /* swap next and prev for all nodes of
    doubly linked list */
    while (current != NULL)
    {
        temp = current->prev;
        current->prev = current->next;
        current->next = temp;
        current = current->prev;
    }

    /* Before changing head, check for the cases like empty
    list and list with only one node */
    if(temp != NULL )
        *head_ref = temp->prev;
}

/* UTILITY FUNCTIONS */
/* Function to insert a node at the begining of the Doubly 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;

    /* since we are adding at the beginning,
    prev is always NULL */
    new_node->prev = NULL;

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

    /* change prev of head node to new node */
    if((*head_ref) != NULL)
        (*head_ref)->prev = new_node ;

    /* move the head to point to the new node */
}
```

```

    (*head_ref)    = new_node;
}

/* Function to print nodes in a given doubly linked list
   This function is same as printList() of singly linked list */
void printList(struct node *node)
{
    while(node!=NULL)
    {
        printf("%d ", node->data);
        node = node->next;
    }
}

/* Driver program to test above functions*/
int main()
{
    /* Start with the empty list */
    struct node* head = NULL;

    /* Let us create a sorted linked list to test the functions
       Created linked list will be 10->8->4->2 */
    push(&head, 2);
    push(&head, 4);
    push(&head, 8);
    push(&head, 10);

    printf("\n Original Linked list ");
    printList(head);

    /* Reverse doubly linked list */
    reverse(&head);

    printf("\n Reversed Linked list ");
    printList(head);

    getchar();
}

```

Time Complexity: $O(n)$

We can also swap data instead of pointers to reverse the Doubly Linked List. [Method used for reversing array](#) can be used to swap data. Swapping data can be costly compared to pointers if size of data item(s) is more.

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



51 Comments Category: Linked Lists

Related Posts:

- Merge two sorted linked lists such that merged list is in reverse order
- Compare two strings represented as linked lists
- Rearrange a given linked list in-place.
- Sort a linked list that is sorted alternating ascending and descending orders?
- Select a Random Node from a Singly Linked List
- Merge Sort for Doubly Linked List
- Point to next higher value node in a linked list with an arbitrary pointer
- Swap nodes in a linked list without swapping data

(Login to Rate and Mark)

2.2 Average Difficulty : **2.2/5.0**
Based on **4** vote(s)

☐

Add to TODO List

☐

Mark as DONE

Like Share 12 people like this.

Writing code in comment? Please use code.geeksforgeeks.org, generate link and share the link here.

@geeksforgeeks, Some rights reserved

[Contact Us!](#)

[About Us!](#)

[Advertise with us!](#)