# Introduction to Singly Linked List (2)

A screenshot of a computer

Description automatically generated

#include <bits/stdc++.h>

using namespace std;

struct Node{

int data;

Node \*next;

Node(int data1, Node \*next1){

data = data1;

next = next1;

}

Node(int data1){

data = data1;

next = nullptr;

}

};

Node \*converttoLL(vector<int> &arr){

Node \*head = new Node(arr[0]);

Node \*mover = head;

for(int i=1; i<arr.size(); i++){

Node \*temp = new Node(arr[i]);

mover->next = temp;

mover = temp;

}

return head;

}

int lenofLL(Node \*head){

Node \*temp = head;

int count=0;

while(temp){

count++;

temp = temp->next;

}

return count;

}

int searchInLinkedList(Node<int> \*head, int k) {

// Write your code here.

Node<int> \*temp = head;

while(temp){

if(temp->data == k) return 1;

temp = temp->next;

}

return 0;

}

int main()

{

//std::cout<<"Hello World";

vector<int> arr={1, 3, 4, 5};

Node \*head = converttoLL(arr);

Node \*temp = head;

//traverse LinkedList

/\*while(temp){

cout << temp->data << " ";

temp = temp->next;

}

cout << head->next;\*/

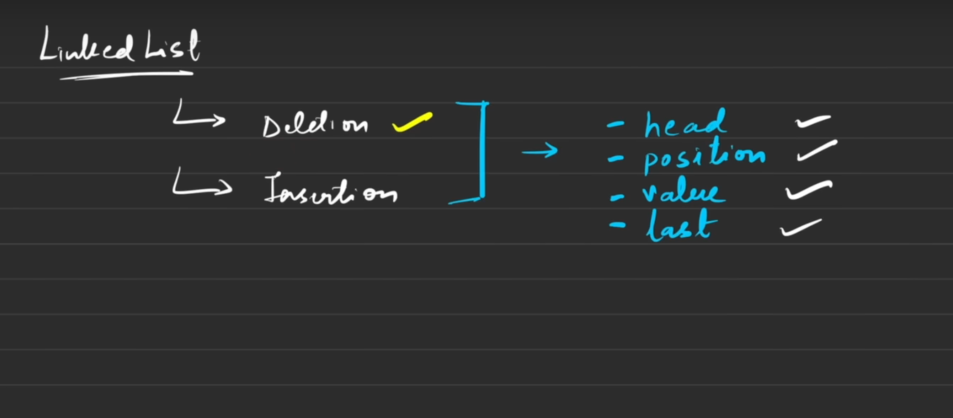
//Length of LinkedList

cout << lenofLL(head);

return 0;

}

# Deletion and Insertion in Singly LL (3)



**Deletion code:**

#include <bits/stdc++.h>

using namespace std;

struct Node{

int data;

Node \*next;

Node(int x){

data=x;

next=nullptr;

}

};

Node \*vtoLL(vector<int> &v){

Node \*head = new Node(v[0]);

Node \*mover = head;

for(int i=1; i<v.size(); i++){

Node \*temp = new Node(v[i]);

mover->next = temp;

mover = temp;

}

return head;

}

void print(Node \*head){

Node \*temp = head;

while(temp){

cout << temp->data << " ";

temp = temp->next;

}

}

//Remove head

Node \*removehead(Node \*head){

if(head == NULL) return head;

Node \*temp = head;

head = head->next;

delete temp;

return head;

}

//Remove tail

Node \*removetail(Node \*head){

if(head==NULL || head->next==NULL) return NULL;

Node \*temp = head;

while(temp->next->next != NULL){

temp = temp->next;

}

delete temp->next;

temp->next=nullptr;

return head;

}

//Remove value from Kth place

Node \*removefromkthposition(Node \*head, int k){

if(head == NULL) return head;

if(k==1){

Node \*temp = head;

head = head->next;

free(temp);

return head;

}

int c=0;

Node \*pre = NULL;

Node \*temp = head;

while(temp != NULL){

c++;

if(c == k){

pre->next = pre->next->next;

free(temp);

break;

}

pre = temp;

temp = temp->next;

}

return head;

}

//Remove Element from LL

Node \*removeelement(Node \*head, int el){

if(head == NULL) return head;

if(head->data == el){

Node \*temp = head;

head = head->next;

free(temp);

return head;

}

Node \*pre=NULL;

Node \*temp=head;

while(temp != NULL){

if(temp->data == el){

pre->next = pre->next->next;

free(temp);

break;

}

pre = temp;

temp = temp->next;

}

return head;

}

int main()

{

vector<int> v={1, 2, 3, 4};

Node \*head = vtoLL(v);

print(head);

cout << "\n";

/\*head = removehead(head);

print(head);

cout << "\n";

head = removetail(head);

head = removefromkthposition(head, 2);\*/

head = removeelement(head, 2);

print(head);

return 0;

}

**Insertion in Linked List code:**

#include <bits/stdc++.h>

using namespace std;

struct Node{

int data;

Node \*next;

Node(int x){

data=x;

next=nullptr;

}

Node(int x, Node \*n){

data=x;

next = n;

}

};

Node \*vtoLL(vector<int> &v){

Node \*head = new Node(v[0]);

Node \*mover = head;

for(int i=1; i<v.size(); i++){

Node \*temp = new Node(v[i]);

mover->next = temp;

mover = temp;

}

return head;

}

void print(Node \*head){

Node \*temp = head;

while(temp){

cout << temp->data << " ";

temp = temp->next;

}

}

//Insert in tail

Node \*insertintail(Node \*head, int val){

if(head == NULL) return new Node(val, head);

Node \*temp = head;

while(temp->next != NULL){

temp = temp->next;

}

Node \*newNode = new Node(val);

temp->next = newNode;

return head;

}

//Insert Value in Kth place

Node \*insertinKposition(Node \*head, int el, int k){

if(head == NULL){

if(k==1) return new Node(el);

else return head;

}

if(k==1) return new Node(el, head);

int c=0;

Node \*temp=head;

while(temp != NULL){

c++;

if(c == (k-1)){

Node \*newNode = new Node(el, temp->next);

temp->next = newNode;

break;

}

temp = temp->next;

}

return head;

}

//insert an Element before a value in Linked List

Node \*insertelementbeforeAvalue(Node \*head, int el, int key){

if(head == NULL) return NULL;

if(head->data == key) return new Node(el, head);

Node \*temp = head;

while(temp->next != NULL){

if(temp->next->data == key){

Node \*newNode = new Node(el, temp->next);

temp->next = newNode;

break;

}

temp = temp->next;

}

return head;

}

int main()

{

vector<int> v={1, 2, 3, 4};

Node \*head = vtoLL(v);

print(head);

cout << "\n";

/\*head = new Node(0, head); //Inserting new node in Linked List

head = insertintail(head, 5);

head = insertinKposition(head, 0, 3);\*/

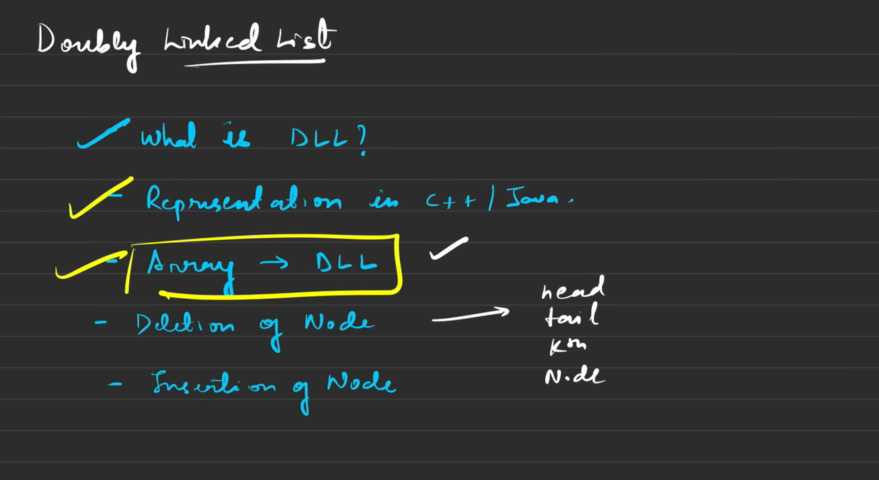
head = insertelementbeforeAvalue(head, 0, 1);

print(head);

return 0;

}

# Doubly Linked List (4)



**Deletion operations code in Doubly Linked List:**

A screen shot of a black board

Description automatically generated

#include <bits/stdc++.h>

using namespace std;

struct Node{

int data;

Node \*next;

Node \*back;

Node(int x){

data=x;

next=nullptr;

back = nullptr;

}

Node(int x, Node \*n, Node \*b){

data=x;

next = n;

back = b;

}

};

Node \*vtoDLL(vector<int> &v){

Node \*head = new Node(v[0]);

Node \*prev = head;

for(int i=1; i<v.size(); i++){

Node \*temp = new Node(v[i], nullptr, prev);

prev->next = temp;

prev = temp;

}

return head;

}

void print(Node \*head){

Node \*temp = head;

while(temp != NULL){

cout << temp->data << " ";

temp = temp->next;

}

}

//Deletion of Head in Dubly Linked List

Node \*deletehead(Node \*head){

if(head==NULL | head->next==NULL) return NULL;

Node \*prev = head;

head = head->next;

head->back = nullptr;

prev->next = nullptr;

delete prev;

return head;

}

//Deletion of tail in Doubly likenlist

Node \*deletetail(Node \*head){

if(head==NULL || head->next==NULL) return NULL;

Node \*tail = head;

while(tail->next != NULL) tail = tail->next;

Node \*newtail = tail->back;

newtail->next = nullptr;

tail->back = nullptr;

delete tail;

return head;

}

//Deleting kth element from Doubly Linked List

Node \*deleteKthelement(Node \*head, int k){

if(head == NULL) return NULL;

int c=0;

Node \*kthnode = head;

while(kthnode != NULL){

c++;

if(c==k) break;

kthnode = kthnode->next;

}

//if(k>c) return head;

Node \*prev = kthnode->back;

Node \*front = kthnode->next;

if(prev==NULL && front==NULL) return NULL;

else if(prev == NULL) return deletehead(head);

else if(front == NULL) return deletetail(head);

prev->next = front;

front->back = prev;

kthnode->back=nullptr;

kthnode->next=nullptr;

delete kthnode;

return head;

}

//Delete node with reference

void deleteNode(Node \*temp){

Node \*prev=temp->back;

Node \*front=temp->next;

if(front == NULL){

prev->next = temp->back = nullptr;

free(temp);

return;

}

prev->next = front;

front->back = prev;

temp->next = temp->back = nullptr;

free(temp);

}

int main()

{

vector<int> v={1, 2, 3, 4};

Node \*head = vtoDLL(v);

print(head);

cout << "\n";

//head = deletehead(head);

//head = deletetail(head);

//head = deleteKthelement(head, 2);

deleteNode(head->next);

print(head);

return 0;

}